Optimizing Quality for Probabilistic Skyline Computation and Probabilistic Similarity Search

Xiaoye Miao, Yunjun Gao, *Member, IEEE,* Linlin Zhou, Wei Wang, Qing Li, *Senior Member, IEEE*

Abstract—Probabilistic queries have been extensively explored to provide answers with confidence, in order to support the real-life applications struggling with uncertain data, such as sensor networks and data integration. However, the uncertainty of data may propagate, and thus, the results returned by probabilistic queries contain much noise, which *degrades* query quality significantly. In this paper, we propose an efficient optimization framework, termed as QueryClean, for both probabilistic skyline computation and probabilistic similarity search. The goal of QueryClean is to optimize query quality via selecting a group of uncertain objects to clean under limited resource available, where a joint-entropy based quality function is leveraged. We develop an efficient structure called ASI to index the possible result sets of probabilistic queries, which helps to avoid many times of probabilistic query evaluations over a large number of the possible worlds for quality computation. Moreover, we present *exact* and *approximate* algorithms for the optimization problem, using two newly presented heuristics. Considerable experimental results on both real and synthetic data sets demonstrate the efficiency and scalability of our proposed framework QueryClean.

Index Terms—Probabilistic Skyline Query, Probabilistic Similarity Query, Query Quality, Optimization Algorithms

1 Introduction

Uncertain data exists in many real-life applications due to a variety of reasons, e.g., the noise in sensor inputs or errors in wireless transmission [1], missing or incorrect values in data integration [2], etc. Consequently, the query processing on uncertain data has received much attention from database community, such as probabilistic skyline computation [3], [4], [5], probabilistic nearest neighbor search [6], [7], [8], [9], [10], probabilistic top-k query [11], [12], [13], and so forth.

A probabilistic query returns, from an uncertain database, the objects with non-zero probabilities to be the query result. Hence, the uncertainty of the data objects propagates to the query results, even though users usually expect to obtain correct and accurate results. Accordingly, it is difficult for the users to identify good data objects and make correct decisions from the answer/result sets with much noise, especially for the data set with high uncertainty. Thus, the probabilistic query has poor quality, resulting in poor decisions. Furthermore, critical decisions based on poor-quality data have very serious consequences[1]. As reported by Gartner [14], poor data quality is a primary reason for 40% of all business initiatives failing to achieve their targeted benefits, and data quality affects overall labor productivity by as much as a 20%.

- *X. Miao, Y. Gao, and L. Zhou are with the College of Computer Science, Zhejiang University, 38 Zheda Road, Hangzhou 310027, China. E-mail: {miaoxy, gaoyj, zlinlin}@zju.edu.cn.*
- *Y. Gao is with the Key Laboratory of Big Data Intelligent Computing of Zhejiang Province, Zhejiang University, Hangzhou 310027, China.*
- *W. Wang is with the School of Computer Science and Engineering, The University of New South Wales, Sydney, 2052, Australian. E-mail: weiw@cse.unsw.edu.au.*
- *X. Miao and Q. Li are with the Department of Computer Science, City University of Hong Kong, Hong Kong, China. E-mail: {xiaomiao, itqli}@cityu.edu.hk.*

1. Available at http://competitiveanalytics.com/data-cleansing/.

It is well-known that data cleaning is an effective way to improve data quality [15]. Nevertheless, in most cases, data cleaning is a labor-intensive, time-consuming, and expensive process, and cleaning all the data is usually neither cost-justified nor practical [15]. Therefore, it is infeasible to clean all data objects due to limited resources available. As a consequence, complementary to the fruitful work upon probabilistic models and queries, in this paper, we aim to improve the quality of probabilistic query results via making full use of limited budget to find the beneficial objects (to clean) for quality improvement.

Example 1. **A motivating example**. Assume that Bob wants to buy a used car from four candidate cars o_1, o_2, o_3, and o_4 shown in Table 1(a), where every car is represented by an uncertain object with several normalized price $(P.)$ and mileage $(M.)$ pairs (i.e., tuples) from different data sources (e.g., web sites). Without loss of generality, the reliability of each data source refers to the probability of the corresponding tuple. Table 1(b) depicts the price and mileage of every car in a 2D space.

A P-skyline query can be employed to recommend the favorite cars for Bob, which retrieves the cars that have the non-zero probabilities to be not dominated by any other cars. Here, a car o dominates another car o' iff o is not worse than o' in both attributes of price and mileage, and is better than o' in at least one attribute of price and mileage. As an example, the probability of t_1 being a P-skyline tuple, denoted as $\Pr_s(t_1)$, is equal to the reliability of t_1 (i.e., $Prob.$) multiplying the probability of t_1 being not dominated by any tuple of other objects. Since o_2's tuple t_4 and all tuples of o_3 and o_4 do not dominate t_1, $\Pr_s(t_1) = \Pr(t_1) \times \Pr(t_4) \times 1 \times 1 = 0.08$. Similarly, $\Pr_s(t_2) = 0.7$ and $\Pr_s(t_3) = 0.2$. Thus, the uncertain object o_1 is a P-skyline object with the probability $\Pr_s(o_1) = \Pr_s(t_1) + \Pr_s(t_2) + \Pr_s(t_3) = 0.98$. Likewise, one can calculate $\Pr_s(o_2) = 1$, $\Pr_s(o_3) = 0.48$, and $\Pr_s(o_4) = 0$.

When Bob verifies the exact price and the mileage of the *Jeep* car by contacting its owner, Bob makes sure that tuple

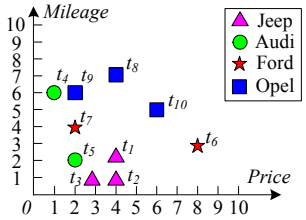
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2805824, IEEE Transactions on Knowledge and Data Engineering

TABLE 1: Illustration of a Real $CarDB$ Dataset

(a) The original dataset

ID	Car	Tuple (P., M.)	Prob.
o_1	Jeep	t_1 (4, 2)	0.1
		t_2 (4, 1)	0.7
		t_3 (3, 1)	0.2
o_2	Audi	t_4 (1, 6)	0.8
		t_5 (2, 2)	0.2
o_3	Ford	t_6 (8, 3)	0.4
		t_7 (2, 4)	0.6
o_4	Opel	t_8 (4, 7)	0.3
		t_9 (2, 6)	0.2
		t_{10} (6, 5)	0.5

(b) The dataset in a 2D space



(c) The dataset after cleaning o_1

ID	Car	Tuple (P., M.)	Prob.
o_1	Jeep	t_2 (4, 1)	1.0
o_2	Audi	t_4 (1, 6)	0.8
		t_5 (2, 2)	0.2
o_3	Ford	t_6 (8, 3)	0.4
		t_7 (2, 4)	0.6
o_4	Opel	t_8 (4, 7)	0.3
		t_9 (2, 6)	0.2
		t_{10} (6, 5)	0.5

t_2 is the exact information having 100% reliability (as listed in Table 1(c)). At this time, we can get the new probabilities $\Pr'_s(o_1) = 1$, $\Pr'_s(o_2) = 1$, $\Pr'_s(o_3) = 0.48$, and $\Pr'_s(o_4) = 0$. Hence, compared with the former query results, the query results after verifying/cleaning o_1 have $\Pr'_s(o_1) = 1$, which is more certainty than $\Pr_s(o_1) = 0.98$. It indicates that, the uncertainty of query results decreases after cleaning objects.

In this paper, we aim at optimizing the quality of probabilistic skyline (P-skyline) query and similarity search including probabilistic k nearest neighbor (P-kNN) query and probabilistic range (P-range) query. Existing strategies only focus on simple queries such as max query [16], region query [16], and PT-k query [17]. Since optimization methods are *query-dependent*, existing strategies cannot efficiently support the quality optimization problem of the probabilistic skyline query and probabilistic similarity search.

As a result, in this paper, we present an efficient optimization framework, termed as QueryClean, to choose the most beneficial uncertain objects to clean for improving the quality, where a joint-entropy based quality function (denoted as κ) is leveraged. There are two main operations in QueryClean, i.e., *quality computation* and *object selection*. Quality computation is to derive the expected query quality for each chosen object set to clean. Object selection aims to obtain a set of chosen objects with the maximum expected quality under limited cleaning budget.

On one hand, for each chosen object set to clean, an intuitive method to calculate the joint-entropy based expected quality is to collect the probabilities of all the result sets, which needs many times of probabilistic query evaluations on all possible worlds. It is costly, even infeasible, let alone that *multiple* expected quality computations are needed to find the set of objects to clean with the maximum expected quality in object selection problem. Therefore, *efficient expected quality computation* is the first challenge we have to solve. Towards this, we develop an efficient structure, called ASI, which indexes all the possible query result sets. The remarkable attraction of ASI lies in, the expected quality for any chosen object set (to clean) can be calculated directly based on the structure of ASI, instead of performing multiple probabilistic queries over all the possible worlds. In other words, one ASI structure can serve each quality computation if there is no update on the data set or the query parameter (e.g., k in P-kNN search).

On the other hand, in terms of the object selection, it is a combinational problem to find the object set to clean with the maximum expected quality. We have to evaluate the combinational number of chosen object sets to clean. Hence, the second challenge is, *within a large search space, to find the optimal object set to clean that has the maximum expected quality*. In view of this, we identify a much smaller candidate object set, and thus shrink the search space significantly. Moreover, for a probabilistic query ϕ (not limit on probabilistic skyline and similarity queries), the expected quality function $\mathbb{E}[\kappa(\phi|O \subseteq \mathcal{S})]$ of choosing an object set O (included in a dataset $\mathcal{S}$) to clean is *monotonic*. In other words, $\mathbb{E}[\kappa(\phi|O_1)] \leq \mathbb{E}[\kappa(\phi|O_2)]$ if $O_1 \subseteq O_2 \subseteq \mathcal{S}$. Hence, using the monotonic property, once we find an object set having no exceeding cost to cleaning budget, it is unnecessary to evaluate all its subsets. Furthermore, we present an *exact* algorithm as well as a greedy algorithm and a sampling-based algorithm, in order to accelerate object selection. To sum up, the key contributions of this paper are as follows:

- We propose an efficient framework QueryClean to optimize the quality of probabilistic skyline computation and probabilistic similarity search.
- We present a novel index structure, namely, ASI, based on which an efficient algorithm, called RrB, is developed for quality computation.
- In addition to an exact algorithm, we propose a greedy algorithm and a sampling-based algorithm for object selection, where two effective heuristics are utilized. Also, we analyze the complexities of our proposed algorithms.
- Extensive experiments using both real and synthetic data sets demonstrate the performance of QueryClean.

The rest of the paper is organized as follows. Section 2 formalizes the problem. Then, we introduce QueryClean framework in Section 3. Efficient algorithms are elaborated for quality computation and object selection in Section 4 and Section 5, respectively. The experimental results and our findings are reported in Section 6. Finally, Section 7 reviews related work, and Section 8 concludes the paper with some directions for future work.

2 Preliminaries

In this section, we first introduce the uncertain data model. Then, we define the (expected) quality function, and formalize our problem studied in this paper. Table 2 lists the symbols used frequently throughout the paper.

Given an uncertain dataset $\mathcal{S} = \{o_1, o_2, \cdots\}$, o_i is an uncertain object. An uncertain object could be conceptually described by a probability density function (pdf) f in a data space D. Generally, $f(o) \geq 0$ for any object o in the data space D, and $\int_{o \in D} f(o)do = 1$. In fact, the probability density function of an uncertain object is often unavailable explicitly. Instead, a set of samples are drawn or collected in the hope of approximating the probability density function [5]. Thus, without loss of generality, in this paper, one uncertain object o_i is denoted as a set of multiple deterministic objects, called o_i's tuples, each of which has an

TABLE 2: Symbols and Description

Notation	Description
$o_i \in \mathcal{S}$	the i-th uncertain object in a dataset $\mathcal{S}$
$\lvert\mathcal{S}\rvert$	the number of the objects in $\mathcal{S}$
$o_i^t \in o_i$	a tuple w.r.t. an object o_i
$\Pr(o_i^t)$	the existence probability of a tuple o_i^t
I	the average number of tuples for an uncertain object
$c(o_i)$	the cost of cleaning an uncertain object $o_i \in \mathcal{S}$
$\phi(q;\mathcal{S})$	a probabilistic query on $\mathcal{S}$ with a query object q
$\Pr_s(o_i)$	the probability of an object o_i being a query answer
r	a tuple-based answer set (which contains a set of tuples) for a query $\phi(q;\mathcal{S})$
$\Pr(r)$	the probability of r being a tuple-based answer set
$\kappa(\phi(q;\mathcal{S}))$	the original quality of a query ϕ w.r.t. q and $\mathcal{S}$
$\kappa(\phi\lvert\{o_i^t\})$	the quality of a query ϕ after the uncertain object set $\{o_i\}$ is cleaned as the tuple set $\{o_i^t\}$
R	an object-based answer set (including a set of uncertain objects) for a query $\phi(q;\mathcal{S})$
$\Pr(R)$	the probability of R being an object-based answer set
$\mathbb{E}[\kappa(\phi\lvert\{o_i\})]$	the expected quality of choosing the object set $\{o_i\}$ to clean
$\mathcal{O} \subseteq \mathcal{S}$	a candidate object set for object selection

existence probability. Note that, the tuple's probability can be inferred by machine learning models, e.g., Bayesian network [18].

Definition 1. **(Possible world)**. A possible world $w = \{o_1^{t_1}, o_2^{t_2}, \cdots\}$ is a set of the instances containing *one and only one* tuple $o_i^{t_i}$ from every uncertain object $o_i \in \mathcal{S}$. Let $\mathcal{W}$ be the set of all the possible worlds. If uncertain objects are supposed to be independent between each other, the probability of a possible world w can be derived by Eq. 1.

$$\Pr(w \in \mathcal{W}) = \prod_{o_i^{t_i} \in w} \Pr(o_i^{t_i}) \tag{1}$$

Example 2. For the sample dataset with four uncertain objects $\{o_1, o_2, o_3, o_4\}$ in Table 1(a), there are 3, 2, 2, and 3 tuples w.r.t. o_1, o_2, o_3, and o_4, respectively. Thus, there are in total 36 possible worlds. Among them, there is one possible world $w_1 = \{t_1, t_4, t_6, t_8\}$ with each tuple corresponding to one uncertain object exactly. The probability of w_1, i.e., $\Pr(w_1)$, is $\Pr(t_1) \times \Pr(t_4) \times \Pr(t_6) \times \Pr(t_8)$ = 0.0096.

In this paper, we focus on the kind of general probabilistic queries *without a probability threshold*. In other words, the objects with non-zero probabilities (instead of a larger probability than a certain probability threshold) being result objects are returned by the query. Given an uncertain dataset $\mathcal{S}$, a probabilistic skyline (P-skyline) query finds all the objects that are not dominated by any other object with non-zero probabilities. Here, an object o dominates another object o' iff o is not worse than o' in all attributes, and is better than o' in at least one attribute. By contrast, a probabilistic k nearest neighbor (P-kNN) query retrieves k objects from an uncertain dataset $\mathcal{S}$ closest to a given query point q with non-zero probabilities. A probabilistic range (P-range) query returns all the objects that are within θ distance to the query point q with non-zero probabilities. In particular, the skyline query serves the application scenarios such as multi-criteria decision making and location-based services, while similarity search including P-kNN and P-range queries is a powerful tool in many research areas (e.g., data mining and pattern recognition). Note that, the expected similarity based on the Euclidean distance is employed in P-kNN query and P-range retrieval.

An intuitive way to define the quality of a probabilistic query is to measure the uncertainty of the query result set. Based on the joint-entropy function widely used in information theory, we define the quality of a probabilistic query in Definition 2, where a query result set with fewer uncertainty has a higher quality. It is worth mentioning that, Shannon's entropy has been applied to the uncertain data in many previous work such as [19], [20].

Definition 2. **(Quality of a probabilistic query)**. Given an uncertain dataset $\mathcal{S}$ and a query object q (if it exists), the quality of a probabilistic query ϕ w.r.t. q and $\mathcal{S}$, denoted as $\kappa(\phi(q;\mathcal{S}))$, is defined based on the joint-entropy of a group of variables $X = \langle x_1, x_2, \cdots, x_{|\mathcal{S}|}\rangle$ with each variable x_i corresponding to an event whether the uncertain object o_i is an answer object for $\phi(q;\mathcal{S})$.

$$\kappa(\phi(q;\mathcal{S})) = \sum_X \Pr(X)\log_2\Pr(X) \tag{2}$$

Definition 2 leverages the negative of joint-entropy function to define query quality. Note that, $\log 0$ is defined as 0 in Definition 2. The variable x_i is a 0-1 variable indicating whether an uncertain object o_i is an answer object. If o_i is an answer object, x_i = 1; otherwise, x_i = 0. Thus, the variable vector $X = \langle x_1, x_2, \cdots, x_{|\mathcal{S}|}\rangle$ is a 0-1 vector, and $\Pr(X)$ is actually the probability of the object-based answer set encoded by the 0-1 vector. For instance, if $X = \langle 1, 0, 1, 0\rangle$ w.r.t. our sample dataset in Table 1(a), $\Pr(X)$ is the probability of the object set $\{o_1, o_3\}$ being an object-based answer set.

Although Eq. 2 is easy to understand via referring to the joint-entropy function, the summation of $\Pr(X)\log_2\Pr(X)$ over all the possible assignments of the vector variable X is costly. In total, there are $2^{|\mathcal{S}|}$ specified value vectors w.r.t. the vector X we have to consider for quality computation. For ease of quality computation, we transform the quality function in Eq. 2 into Eq. 3, where the set Ω contains all possible query result sets R.

$$\kappa(\phi(q;\mathcal{S})) = \sum_{R\in\Omega} \Pr(R)\log_2\Pr(R) \tag{3}$$

Eq. 3 benefits the quality computation due to the relatively small space of the answer sets. This is because, for any practical query, the query result set is expected to have a much smaller size than the whole dataset. In addition, for simplifying representation, let R/r be the object/tuple-based answer set, and $\Pr(r)$ be the probability of set r to be a tuple-based answer set. Then, every object-based result set R_i contains a group of tuple-based result sets r_{ij}. The probability of set R_i to be an object-based answer set, denote as $\Pr(R_i)$, is the summation of all the probabilities $\Pr(r_{ij})$s.

Example 3. For the P-skyline query over our sample dataset, all the query result sets are shown in Table 3. The set $R_1(=\{o_1, o_2\})$ is an object-based answer set for the P-skyline query as well as $R_2(=\{o_1, o_2, o_3\})$ and $R_3(=\{o_2\})$. Set R_1 consists of five tuple-based answer sets, i.e., $r_{11}, r_{12}, r_{13}, r_{14}$, and r_{15}. The tuple-based answer set r_{11} contains two tuples t_1 and t_4 w.r.t. objects o_1 and o_2, respectively. Therefore, the probability $\Pr(R_1) = \sum_{i=1}^{5}\Pr(r_{1i})$ = 0.5. Similarly, $\Pr(R_2)$ = 0.48 and $\Pr(R_3)$ = 0.02. Consequently, based on Eq. 3, we can calculate the query quality, i.e., $\kappa(skyline(\mathcal{S}))$ = $\Pr(R_1)\log_2\Pr(R_1)$ + $\Pr(R_2)\log_2\Pr(R_2)$ + $\Pr(R_3)\log_2\Pr(R_3)$ = $0.5\log_2 0.5 + 0.48\log_2 0.48 + 0.02\log_2 0.02 = -1.121$.

It is noteworthy that, the quality is a *negative* value. It makes sense that, the more certain of the query result, the larger the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2805824, IEEE Transactions on Knowledge and Data Engineering

TABLE 3: The P-skyline Query Result Sets

R	r	Pr(r)	R	r	Pr(r)
$R_1 : \{o_1, o_2\}$	$r_{11} : \{t_1, t_4\}$	0.032	$R_2 : \{o_1, o_2, o_3\}$	$r_{21} : \{t_1, t_4, t_7\}$	0.048
	$r_{12} : \{t_2, t_4\}$	0.224		$r_{22} : \{t_2, t_4, t_7\}$	0.336
	$r_{13} : \{t_2, t_5\}$	0.140		$r_{23} : \{t_3, t_4, t_7\}$	0.096
	$r_{14} : \{t_3, t_4\}$	0.064	$R_3 : \{o_2\}$	$r_{31} : \{t_5\}$	0.020
	$r_{15} : \{t_3, t_5\}$	0.040			

quality value. The biggest quality value is *zero*, according to the properties of entropy function.

According to Eq. 3, we aim to minimize the uncertainty of the object-based answer sets, which is motivated by the origin of query processing (to find the objects satisfying some constraints). Instead, the PWS-quality [16], [17] considers the probabilities of tuple-based answer sets. For instance, assume that there are only two different tuple-based answer sets $\{t_1, t_4\}$ and $\{t_2, t_4\}$ (w.r.t. one same object-based answer set $\{o_1, o_2\}$) for a query. Our quality function thinks there is a deterministic query answer set $\{o_1, o_2\}$, while PWS-quality deems there exists uncertainty in query results for the two different tuple-based answer sets. Thus, the techniques used in [16], [17] cannot address our problem due to different optimization objectives.

In addition to the quality functions mentioned earlier, there is a more general query quality, denoted by $\mathcal{G}$. It is defined as the summation of the negative entropy of every uncertain object being an answer object. Let $\Pr(o)$ denote the probability of the object o being an answer object, the query quality $\mathcal{G}$ is equal to $\sum_{o \in \mathcal{S}}(\Pr(o) \log \Pr(o) + (1 - \Pr(o)) \log(1 - \Pr(o)))$. Note that, the quality $\mathcal{G}$ concerns about each independent object, irrespective of the object/tuple-based answer set (which is the basis of the aforementioned two quality functions). We are exploring the quality optimization methods using the general query quality function in our ongoing work.

The expected quality of choosing a group of uncertain objects to clean is stated in Definition 3, based on which one can decide which group of objects is chosen to clean such that the expected query quality is maximized.

Definition 3. **(Expected quality of a probabilistic query)**. For an uncertain dataset $\mathcal{S}$ and a query object q (if it exists), the expected quality of a probabilistic query ϕ when a set O_c of uncertain objects is chosen to clean, denoted by $\mathbb{E}[\kappa(\phi(q; \mathcal{S})|O_c)]$ (abbreviated as $\mathbb{E}[\kappa(\phi|O_c)]$), is defined in Eq. 4. Here, $\kappa(\phi|T_c)$ denotes the quality of query ϕ if the objects in O_c are cleaned as the tuple set T_c, and $\mathcal{T}$ is supposed to contain all the possible tuple sets that O_c could be cleaned as.

$$\mathbb{E}[\kappa(\phi|O_c)] = \sum_{T_c \in \mathcal{T}} \Pr(T_c)\kappa(\phi|T_c) \tag{4}$$

In this paper, we consider data cleaning as a user-provided module, where a variety of cleaning techniques can be employed, such as rule-based approaches [21], data cleaning tools [22], [23], and crowdsourcing [24], [25]. Without loss of generality, for an uncertain object o, our cleaning module simply *determines* o as one of its tuples o^t at the probability of $\Pr(o^t)$, which incurs cleaning cost, denoted by $c(o)$. Note that, there are several points on the cleaning model worth exploring in depth. (i) An uncertain object is cleaned as the point that is derived by a weighted function based on its tuples or by a certain distribution. (ii) In the situations of data integration and data fusion, cleaning one uncertain object helps to reduce the uncertainty of its correlated objects (having the tuples from the same data source). How to improve query quality under this model is an essential and interesting problem.

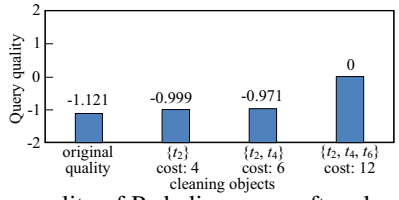


Fig. 1: The quality of P-skyline query after cleaning objects

(iii) For the case of *data noise* [26], the data returned by the cleaning model has a certain error probability. One could minimize the error probability via majority voting strategy. As a result, the problem becomes how to minimize the number of consultations with cleaning agents about cleaning uncertain objects, in order to get the correct cleaned objects having high probability. Another promising method is to strategically incentive the cleaning agents (e.g., crowd workers) with extra bonus for high-quality answers.

Example 4. Take the dataset in Table 1(a) as an example, for the P-skyline query, the quality is plotted in Figure 1 after some objects have been cleaned. Figure 1 also shows the cleaning cost for every set of clean objects, where it is assumed that cleaning the objects o_1, o_2, o_3, and o_4 needs 4, 2, 6, and 3 cents, respectively.

As calculated previously, the original quality is -1.121. After the object o_1 is cleaned as t_2 at the cost of 4 cents, $\kappa(skyline|\{t_2\}) = 0.52 \log_2 0.52 + 0.48 \log_2 0.48 = -0.999$, in which the object-based answer set $R_1 = \{r_{12}, r_{13}\}$ with $\Pr(R_1) = 0.52$, $R_2 = \{r_{22}\}$ with $\Pr(R_2) = 0.48$, and R_3 disappears since t_5 does not exist any longer. For another case, after the objects o_1 and o_2 are cleaned as t_2 and t_4 respectively at the cost of 6 cents, we can get $\kappa(skyline|\{t_2, t_4\}) = -0.971$, where $R_1 = \{r_{12}\}$ with $\Pr(R_1) = 0.4$, $R_2 = \{r_{22}\}$ with $\Pr(R_2) = 0.6$, and R_3 disappears. In addition, after the objects o_1, o_2, and o_3 are cleaned as t_2, t_4, and t_6 respectively at the cost of 12 cents, we can derive that the quality $\kappa(skyline|\{t_2, t_4, t_6\}) = 0$. It is observed that, the quality improves with the enlarging cleaning set, while the cleaning cost is increasing.

Definition 4. **(Our problem)**. Given an uncertain dataset $\mathcal{S}$ and a cleaning budget B, the goal of the problem studied in this paper is to find out a set $O^\star \subseteq \mathcal{S}$ of uncertain objects to clean, such that the expected quality of a probabilistic query ϕ is maximized. Formally,

$$O^\star = \arg_{O_c} \max\{\mathbb{E}[\kappa(\phi|O_c)] \mid O_c \subseteq \mathcal{S}, \sum_{o \in O_c} c(o) \leq B\} \tag{5}$$

It is worthwhile to explain that, computing query quality is at the least as hard as the probabilistic query (without a probability threshold), which has a relatively high complexity due to the exponential number of the possible worlds one has to consider. In addition, it is well-known that entropy is a submodular function [27]. The maximization of a general submodular function, which is usually NP-hard[2], is essentially a sub-problem of our optimization problem. Therefore, quality computation and object selection are two big challenges in our problem, as to be further explained in the next section.

3 Optimization Framework

In this section, we briefly overview our proposed optimization framework QueryClean, in order to solve our problem stated in

2. https://en.wikipedia.org/wiki/Submodular_set_function.

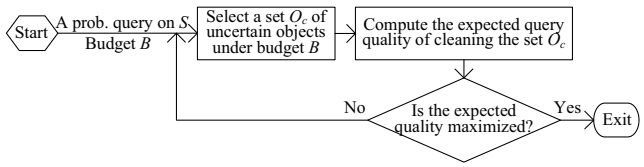


Fig. 2: The flowchart of QueryClean

Definition 4.

Figure 2 depicts the general flowchart of QueryClean framework. Given a cleaning budget B, for a probabilistic query over an uncertain dataset $\mathcal{S}$, QueryClean iteratively selects a group of uncertain objects (stored in a set O_c) for cleaning *with limited resource B available*. QueryClean terminates until the maximum expected query quality of cleaning the set O_c is achieved. Without loss of generality, for ease of understanding, we focus on P-skyline query in the main body of this paper. Note that, QueryClean is able to handle quality optimization on other probabilistic spatial queries. The extension to probabilistic similarity search including P-kNN query and P-range query will be detailed in Section 4.4.

For further understanding, Algorithm 1 shows the pseudo-code of QueryClean framework. For a probabilistic query ϕ, it aims to find the chosen object set $O^\star$ with the maximum expected quality of ϕ, denoted as $\kappa^\star$. Initially, QueryClean sets the maximum expected quality $\kappa^\star$ to the minimal value $-\infty$. Then, for each chosen object set O_c (to clean) satisfying $\sum_{o\in O_c} c(o) \leq B$, QueryClean computes the expected quality of ϕ, i.e., $\mathbb{E}[\kappa(\phi|O_c)]$, by invoking *Expected Quality Computation* (EQC) function (line 3). If $\mathbb{E}[\kappa(\phi|O_c)]$ is larger than $\kappa^\star$, then $\kappa^\star$ is updated to $\mathbb{E}[\kappa(\phi|O_c)]$, and $O^\star$ is updated to O_c (lines 4-5). Finally, QueryClean returns $O^\star$ and $\kappa^\star$, and stops (line 6). Note that, EQC function computes expected quality via Eq. 4, as to be detailed in Section 4.1.

From QueryClean, we can identify that, for each chosen object set O_c to clean, the expected quality computation using EQC function is the dominant computation cost. Hence, QueryClean is of complexity $O(n \cdot \beta)$, in which n is the number of the possible chosen object sets O_c (to clean) satisfying $\sum_{o\in O_c} c(o) \leq B$, and β denotes the average cost of expected quality computation. Thus, we aim to minimize the processing costs in terms of both n and β, which refers to *object selection* and *quality computation*, respectively.

On one hand, for the expected quality computation cost β, it grows exponentially with the dataset cardinality and the average number of tuples w.r.t. an uncertain object. Thus, for a large data set with high uncertainty, β is rather large, and hence, how to efficiently compute expected quality is a key challenge we should address. The goal of Section 4 (to be presented) is to accelerate quality computation.

On the other hand, one can see that the object selection problem in Eq. 5 is *non-linear* due to the nature of Shannon entropy. In fact, the optimization problem is to find out a set of the objects from $\mathcal{S}$ to clean with the complexity $O(2^{|\mathcal{S}|})$. Thus, reducing the number of object combinations for evaluation is one difficult yet essential task we should tackle. To this end, Section 5 (to be presented later) devotes to reducing the number of possible object sets to clean, in order to improve object selection efficiency.

Algorithm 1: QueryClean Framework

Input: an uncertain dataset $\mathcal{S}$, a resource budget B, a query ϕ
Output: the object set $O^\star$ with maximum expected quality $\kappa^\star$

1: $\kappa^\star \longleftarrow -\infty$
2: **foreach** chosen object set $O_c \subseteq \mathcal{S}$ with $\sum_{o\in O_c} c(o) \leq B$ **do**
3: $\quad \mathbb{E}[\kappa(\phi|O_c)] \longleftarrow \mathsf{EQC}(\mathcal{S}, \phi, O_c)$
4: $\quad$ **if** $\mathbb{E}[\kappa(\phi|O_c)] > \kappa^\star$ **then**
5: $\quad\quad \kappa^\star \longleftarrow \mathbb{E}[\kappa(\phi|O_c)]$, $O^\star \longleftarrow O_c$
6: **return** $O^\star$ and $\kappa^\star$
7: **Function**: $\mathsf{EQC}(\mathcal{S}, \phi, O_c)$ `// compute φ's quality`
8: $\mathbb{E}[\kappa(\phi|O_c)] \longleftarrow 0$
9: **foreach** possible clean tuple set T_c in chosen object set O_c **do**
10: $\quad \Pr(T_c) \longleftarrow \Pi_{t\in T_c} \Pr(t)$
11: $\quad \kappa(\phi|T_c) \longleftarrow \mathsf{Quality}(\mathcal{S}, \phi, T_c)$ `// using Eq. 3`
12: $\quad \mathbb{E}[\kappa(\phi|O_c)] \longleftarrow \mathbb{E}[\kappa(\phi|O_c)] + \Pr(T_c) \cdot \kappa(\phi|T_c)$ `// using Eq. 4`
13: **return** $\mathbb{E}[\kappa(\phi|O_c)]$

4 Quality Computation

In this section, we first detail the EQC method for expected quality computation, and then, we propose an effective *answer-set based indexing structure*, called ASI. Using ASI, we present an efficient algorithm RrB for quality computation.

4.1 EQC Method

EQC function implements expected quality computation in a brute-force way based on Eq. 4 (lines 7-13 of Algorithm 1). Specifically, EQC invokes Quality function to compute the query quality after O_c is cleaned as each possible clean tuple set T_c. Here, Quality gets the probabilities of the answer sets via evaluating all the possible worlds. Then, EQC summarizes these qualities w.r.t. all the clean tuple sets to derive the expected quality, based on Eq. 4.

Let I be the average number of tuples w.r.t. one uncertain object, the time complexity of EQC is $O(I^{|O_c|} \cdot I^{|\mathcal{S}|} \cdot \alpha)$ for a specified uncertain dataset $\mathcal{S}$, in which $|O_c|$ is the number of chosen objects O_c to clean, $I^{|\mathcal{S}|}$ is the number of possible worlds, and α represents the cost of performing a query on a possible world. Without loss of generality, α is usually seemed as a constant. Given a chosen object set O_c to clean, it needs $I^{|O_c|}$ times of quality computations to derive the expected quality, since there are $I^{|O_c|}$ possible clean tuple sets O_c can be cleaned as. As an example, for the case of $O_c = \{o_1\}$ over the sample dataset in Table 1(a), the expected quality $\mathbb{E}[\kappa(skyline|\{o_1\})]$ (= $\Pr(t_1) \times \kappa(skyline|\{t_1\})$ + $\Pr(t_2) \times \kappa(skyline|\{t_2\})$ + $\Pr(t_3) \times \kappa(skyline|\{t_3\})$) needs three times quality computations, since o_1 has three possible tuples that could be cleaned as, i.e., I = 3, $|O_c|$ = 1.

The biggest challenge to calculate the expected query quality of choosing O_c (to clean) is the large number (i.e., $I^{|O_c|}$) of quality computations, which increases exponentially with the number of the objects in O_c. Moreover, every quality computation has to conduct a probabilistic query evaluation on $I^{|\mathcal{S}|}$ possible worlds for getting the probabilities $\Pr(R)$, in order to derive the quality.

4.2 ASI Structure

In this subsection, we present the *answer-set based indexing structure*, namely, ASI, which indexes all the possible answer sets for a probabilistic query ϕ. Instead of evaluating a large number (i.e., $I^{|O_c|}$, as discussed earlier) of probabilistic queries over all the possible worlds for quality computation, ASI supports to *directly* derive the probability $\Pr(R)$ of every object-based answer set

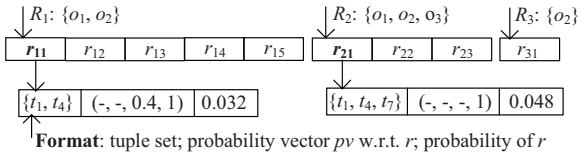


Fig. 3: The ASI structure on the dataset in Table 1(a)

for any chosen object set O_c to clean, and thus, it improves the efficiency of quality computation.

In fact, ASI is a hash table with the object-based answer sets as keys, which stores together all the tuple-based answer sets r_{ij} belonging to the object-based answer set R_i. To be more specific, for every tuple-based answer set r_{ij}, ASI stores the tuples contained in it and the probability of r_{ij}, i.e., $\Pr(r_{ij})$. In addition, it also stores a probability vector, denoted as $\vec{pv}(r_{ij})$, to facilitate the update of $\Pr(r_{ij})$. Each element in $\vec{pv}(r_{ij})$, denoted as $\vec{pv}(r_{ij})[o]$, is an associated probability corresponding to $o \in (\mathcal{S} - R_i)$, indicating the possibility of r_{ij} being a tuple-based answer set in terms of o, regardless of other objects. The probability of a tuple-based answer set is stated below.

Definition 5. **(Tuple-based answer set probability)**. Given a tuple-based answer set $r \in R$, for an uncertain dataset $\mathcal{S}$ and a probabilistic query ϕ, the probability of r being a tuple-based result set, denoted as $\Pr(r)$, is calculated by Eq. 6. Here, $\vec{pv}(r)[o_2]$ is the summation of the existence probabilities of all the tuples of o_2 such that r is an answer set.

$$\Pr(r) = \Pi_{o_1^t \in r} \Pr(o_1^t) \cdot \Pi_{o_2 \in (\mathcal{S}-R)} \vec{pv}(r)[o_2] \quad (6)$$

Example 5. Take our sample dataset in Table 1(a) as an example, its ASI structure for P-skyline query is illustrated in Figure 3. In total there are three object-based answer sets, i.e., $R_1 = \{o_1, o_2\}$, $R_2 = \{o_1, o_2, o_3\}$, and $R_3 = \{o_2\}$. ASI stores three elements for each tuple-based answer set. As an example, for the answer set $r_{11}(= \{t_1, t_4\})$, ASI stores $\{t_1, t_4\}$; the associated probability vector $\vec{pv}(r_{11}) = (-, -, 0.4, 1)$, where the corresponding objects to r_{11} are useless for probability update and hence are omitted (denoted by " $-$ " in the vector); and the probability of r_{11}, i.e., $\Pr(r_{11})$. In particular, $\vec{pv}(r_{11})[o_3]$ = $\Pr(t_6)$ =0.4, since only the tuple t_6 of object o_3 is dominated by t_1 from r_{11} in the P-skyline query. As all the tuples t_8, t_9, and t_{10} of object o_4 are dominated by t_1 or t_4 from r_{11}, $\vec{pv}(r_{11})[o_4]$ = 1. In addition, based on Eq. 6, we derive the probability of r_{11}, i.e, $\Pr(r_{11}) = \Pr(t_1) \times \Pr(t_4) \times \vec{pv}(r_{11})[o_3] \times \vec{pv}(r_{11})[o_4] = 0.1 \times 0.8 \times 0.4 \times 1 = 0.032$, as shown in Figure 3.

It is woth mentioning that, there are at most $(2^{|\mathcal{S}|} - 1)$ object sets that are possible to be chosen to clean, each of which needs to calculate the expected quality, as discussed in Section 3. Instead of evaluating many probabilistic queries, we can derive those query qualities via directly updating the probabilities of all the answer sets, using ASI structure. This is motivated by Lemma 1. Furthermore, the specific probability update strategy for answer sets is stated in Lemma 2.

Lemma 1. Let ω/Ω be the collection of tuple/object-based answer sets of the query over the original dataset $\mathcal{S}$ (stored in ASI), and $\omega(O_c)/\ \Omega(O_c)$ be the collection of tuple/object-based answer sets for the query when the objects in $O_c \subseteq \mathcal{S}$ are chosen to clean. It is confirmed that $\omega(O_c) \subseteq \omega$, $\Omega(O_c) \subseteq \Omega$.

The proof is straightforward via referring to the cleaning model, under which the possible worlds are disappearing when more and more objects are cleaned as one of their tuples. Hence, the query result set over original data set is a superset of the result set when some objects are cleaned. Lemma 1 guarantees the correctness of ASI structure for quality computation. It indicates that, except for the answer sets stored in ASI, there is no new answer set appeared when objects are cleaned. Thus, we only need to update the probabilities $\Pr(R)$ w.r.t. the object-based answer sets R stored in ASI correctly and efficiently for quality computation.

Lemma 2. **(Probability update strategy for answer sets)**. Given a clean tuple set T_c and a tuple-based answer set $r \in R$ with the probability $\Pr(r)$, if r exists (i.e., $\Pr(r) \neq 0$), $\Pr(r)$ can be updated by Eq. 7, where $\Pr^{\mathcal{A}}(r)$ is the probability of r stored in ASI structure, and $\vec{pv}(r)[o_j]$ is the probability of r being an answer set in terms of o_j.

$$\Pr(r) = \frac{\Pr^{\mathcal{A}}(r)}{\Pi_{o_i^{t_1} \in (T_c \cap r)} \Pr(o_i^{t_1}) \cdot \Pi_{o_j^{t_2} \in (T_c - r)} \vec{pv}(r)[o_j]} \quad (7)$$

Proof. For a tuple $o^t \in T_c$, it means that the object o is cleaned as o^t, and thus, the existence probability of the tuple o^t becomes one after cleaning. Based on Definition 5, $\Pr(r)$ is calculated by multiplying two parts w.r.t. two groups of tuples/objects. Hence, the update of $\Pr(r)$ also refers to two groups of tuples, i.e., $T_c \cap r$ and $(T_c - r)$. Specifically, if tuple $o^t \in T_c$ belongs to r, the updated $\Pr(r)$ equals the original probability $\Pr^{\mathcal{A}}(r)$ divided by the existence probability $\Pr(o^t)$. If tuple $o^t \in T_c$ is not included in r, it indicates that o^t helps r to be an answer set. Thus, the updated $\Pr(r)$ is obtained by $\Pr^{\mathcal{A}}(r)/\vec{pv}(r)[o]$, according to Definition 5. Therefore, $\Pr(r)$ is updated by Eq. 7. □

We would like to highlight that, ASI structure is static and constant during the whole procedure of object selection for cleaning, and it can be reused for multiple (expected) quality computations. In the worst case, it takes $O(I^{|\mathcal{S}|} \cdot |\mathcal{S}|^2)$ time in getting all the tuple-based answer sets (collected in a set ω) for P-skyline query. Here, $I^{|\mathcal{S}|}$ is the number of possible worlds. The time complexity of performing a query on every possible world is $O(|\mathcal{S}|^2)$. Thus, the construction of ASI needs the complexity of $O(I^{|\mathcal{S}|} \cdot |\mathcal{S}|^2)$. Furthermore, the space complexity of ASI is $O(|\omega|)$. In general, using Heuristic 1 (to be explained later), the candidate data set size is much smaller than $|\mathcal{S}|$, which makes it possible to construct ASI efficiently on large data sets.

In our implementation, for accelerating ASI construction, we utilize several efficient structures, heuristics, and strategies. As an example, three of the techniques for ASI construction in P-skyline queries are summarized as follows.

(i) *MBR pruning*: The minimum bounding rectangle (MBR) of every uncertain object is utilized to prune the objects having zero-probability to be query answers, since those objects are useless for quality improvement (as proved in Heuristic 1 later). Hence, the data scale drops significantly, and thus, the number of possible worlds is exponentially minimized for ASI construction.

(ii) *Possible world access strategy*: We sort the unpruned objects by the bottom-left coordinates so that only the former object is possible to have the ability to dominate all tuples of an object ranked behind. Then, we traverse the possible worlds (to compute skyline objects) in a similar depth-first search (DFS) traversal with the first ranked object as root node. This evaluation strategy can help to further skip the evaluation of some possible worlds for ASI construction.

(iii) *Multiple-layer hash*: It needs too much time to decide whether a new coming tuple-based answer set has been stored

previously in ASI structure. Therefore, we index all the tuple-based answer sets (achieved from the evaluated possible worlds) by a hash table. It inserts a new tuple-based answer set r into ASI via three keys, i.e., the number of tuples, the identifer of the first object, and the identifer of the first tuple in the tuple-based answer set consecutively. The set r is finally inserted into ASI only when there is no r stored in ASI previously. Otherwise, only the probability of r stored in ASI is updated by adding the probability of the new r. In other words, each new coming answer set is processed via three hash operations. Hence, it reduces the complexity to $O(|\omega|)$ from $O(|\omega|^2)$, where $|\omega|$ is the number of all tuple-based answer sets. In addition, it is important to note that, using this technique, ASI structure can be updated easily, if the query answer sets change.

In addition, we would like to explain the reason behind we do not employ the layered range tree [28]. Different from finding the probabilistic k-skyline sets supported by the layered range tree [28], the answer set in our work has no constant size. Hence, there are a much larger number of answer sets we have to derive than the number of the probabilistic k-skyline sets in [28], and thereby we cannot skip the evaluation of each possible world. As a result, the layered range tree is not able to benefit/accelerate the probability computation of the tuple-based answer set (stored in ASI index).

Furthermore, there is a heuristic approach for ASI construction. It evaluates a fraction of possible worlds, instead of traversing all the possible worlds, and ASI is constructed only based on the obtained tuple-based answer sets from the evaluated possible worlds. This method is promising, and worth further improving in our future work, as confirmed by our experimental evaluation in Section 6.1.

4.3 RrB Algorithm

In this section, we propose an efficient algorithm for *quality computation*, namely, RrB, With the support of ASI structure, we have an improved version of EQC method, called IEQC, which uses RrB to calculate *expected* quality.

Algorithm 2 depicts the pseudo-code of RrB, which takes as inputs an uncertain dataset $\mathcal{S}$, a probabilistic query ϕ, a clean tuple set T_c w.r.t. a chosen object set O_c to clean (i.e., O_c is cleaned as a clean tuple set T_c), and an ASI structure $\mathcal{A}$, and outputs the query quality $\kappa(\phi|T_c)$. First, RrB sets quality κ as $-\infty$ (line 1). Then, it visits every object-based answer set R_i from ASI structure $\mathcal{A}$ to update the probability of $\Pr(R_i)$ via the summation of the updated probabilities $\Pr(r_{ij})$ for quality computation (lines 2-8). Since the possible worlds are disappearing when objects are cleaned, some tuple-based answer sets r may disappear (i.e., $\Pr(r) = 0$). Hence, it is sufficient to update $\Pr(r)$ if r exists. As a result, prior to computing $\Pr(r)$, RrB firstly verifies the existence of r_{ij} using two conditions (line 5). Only both of the conditions hold, r_{ij} is still an answer set.

Specifically, when satisfying the first condition $|T_c \cap r_{ij}| = |O_c \cap R_i|$, it prunes the case of r_{ij}, in which there exists some object $o \in O_c \cap R_i$, but o is cleaned as its tuple o^t ($\in T_c$) that is not included in set r_{ij}. In other words, only when every object in $O_c \cap R_i$ is cleaned as the tuple in r_{ij}, the set r_{ij} may still exist. The second condition is that, r_{ij} dominates all the tuples in $(T_c - r_{ij})$ for P-skyline query. When the second condition holds, it makes sure that any clean tuple in $(T_c - r_{ij})$ cannot be a query answer and thus guarantees that r_{ij} is still a tuple-based answer set. For example, for an object $o \in O_c$ but $o \notin R_i$, after o is cleaned as its tuple o^t ($\in T_c$), if o^t is not dominated by any tuple in r_{ij}, then r_{ij} disappears (i.e., it is not a tuple-based answer set any more), and hence, $\Pr(r_{ij}) = 0$. For each r_{ij} existed, RrB updates $\Pr(r_{ij})$ based on Eq. 7 (line 6). After evaluating all the object-based answer sets (collected in the set Ω), the quality $\kappa(\phi|T_c)$ is derived and returned (lines 8-9).

Algorithm 2: RrB Algorithm

Input: an uncertain dataset $\mathcal{S}$, a query ϕ, a clean tuple set T_c w.r.t. a chosen object set O_c to clean, an ASI index $\mathcal{A}$
Output: the quality $\kappa(\phi|T_c)$

1: $\kappa(\phi|T_c) \longleftarrow -\infty$
2: **foreach** possible answer object set $R_i \in \Omega$ **do**
3: $\quad \Pr(R_i) \longleftarrow 0$
4: $\quad$ **foreach** possible answer tuple set $r_{ij} \in R_i$ **do**
5: $\quad\quad$ **if** $|T_c \cap r_{ij}| = |O_c \cap R_i|$ **and** for $\forall o_1^{t_1} \in T_c - r_{ij}$, $\exists o'^{t'} \in r_{ij}$ dominates tuple $o_1^{t_1}$ for P-skyline **then**
6: $\quad\quad\quad \Pr(r_{ij}) \leftarrow \frac{\Pr^{\mathcal{A}}(r_{ij})}{\prod_{o_2^{t_2} \in T_c \cap r_{ij}} \Pr(o_2^{t_2}) \cdot \prod_{o_1^{t_1} \in T_c - r_{ij}} \vec{pv}(r_{ij})[o_1]}$ // using Eq. 7
7: $\quad\quad\quad \Pr(R_i) \longleftarrow \Pr(R_i) + \Pr(r_{ij})$
8: $\quad \kappa(\phi|T_c) \longleftarrow \kappa(\phi|T_c) + \Pr(R_i) \log_2 \Pr(R_i)$ // using Eq. 4
9: **return** $\kappa(\phi|T_c)$

Example 6. For the dataset in Table 1(a), we use RrB to calculate the quality $\kappa(skyline|\{t_2\})$ after O_c (= $\{o_1\}$) is cleaned as the clean tuple set T_c (= $\{t_2\}$). The corresponding ASI structure is depicted in Figure 3.

To begin with, RrB visits the object-based answer set R_1. For R_1 (= $\{o_1, o_2\}$ shown in Figure 3), RrB evaluates its tuple-based answer sets r_{11}, r_{12}, r_{13}, r_{14}, and r_{15} in order. For r_{11} (= $\{t_1, t_4\}$), as $|T_c \cap r_{ij}| \neq |O_c \cap R_i|$, r_{11} disappears (i.e., $\Pr(r_{11}) = 0$) for the clean tuple set $T_c = \{t_2\}$. For r_{12} (= $\{t_2, t_4\}$), since the condition $|T_c \cap r_{ij}| = |O_c \cap R_i|$ holds, we can derive $\Pr(r_{12}) = \frac{\Pr^{\mathcal{A}}(r_{12})}{\Pr(o_1^{t_2})} = \frac{0.224}{0.7} = 0.32$. Similarly, as $r_{13} = \{t_2, t_5\}$, we can derive $\Pr(r_{13}) = \frac{\Pr^{\mathcal{A}}(r_{13})}{\Pr(o_1^{t_2})} = \frac{0.14}{0.7} = 0.2$. In addition, the tuple-based answer sets $r_{14} = \{t_3, t_4\}$ and $r_{15} = \{t_3, t_5\}$ disappear. Thus, $\Pr(R_1) = \Pr(r_{12}) + \Pr(r_{13}) = 0.52$. Similarly, we can get $\Pr(R_2) = \Pr(r_{22}) = 0.48$, and $\Pr(R_3) = 0$. Finally, the quality after the object o_1 is cleaned as its tuple t_2, i.e., $\kappa(skyline|\{t_2\}) = 0.52 \log_2 0.52 + 0.48 \log_2 0.48 = -0.999$.

Let ω be the collection of all tuple-based answer sets, and O_c be the chosen object set to clean. The time complexity of RrB algorithm is $O(|\omega| \cdot |O_c|)$, since it needs $O(|O_c|)$ time to check whether a tuple-based answer exists. Hence, the improved expected quality computation algorithm IEQC (using RrB) takes $O(I^{|O_c|} \cdot |\omega| \cdot |O_c|)$ time, in which I is the average number of tuples w.r.t. an uncertain object.

4.4 Extension to Probabilistic Similarity Search

We would like to point out that, for (expected) query quality computation, ASI index and RrB algorithm can directly tackle P-kNN search and P-range query.

On one hand, the presented *MBR pruning* and *multiple-layer hash* in Section 4.2 for ASI construction is also applicable to P-kNN search and P-range query. On the other hand, at line 5 of

Algorithm 2 (i.e., RrB), in addition to the first condition $|T_c \cap r_{ij}| = |O_c \cap R_i|$, for P-$k$NN query, the second condition is that, there is no tuple in $(T_c - r_{ij})$ being k nearest tuples to the query object (i.e., $d(o_1^{t_1}, q) \geq d^k(r_{ij}, q)$). For P-range query, the second condition is that, there is no tuple in $(T_c - r_{ij})$ within θ away from the query object (i.e., $d(o_1^{t_1}, q) > \theta$). Using these modifications, RrB could tackle probabilistic similarity search.

In addition, it is important to note that, object selection is independent of query type. Consequently, as long as the expected quality computation allows for different probabilistic queries, QueryClean can deal with the quality optimization problem for those queries.

5 Improvement on QueryClean

The goal of QueryClean is to select a group of uncertain objects for cleaning such that the expected query quality is maximized with limited resource. It is exactly the responsibility of object selection algorithms. In this section, we present efficient heuristics and algorithms for optimizing object selection.

5.1 Pruning Heuristics

Heuristic 1. For an uncertain object o in an uncertain dataset $\mathcal{S}$, if the probability of o being an answer object, denoted as $\Pr_s(o)$, is zero, cleaning the object o cannot improve the query quality. The object set $\mathcal{O}$ defined in Eq. 8 forms a candidate set for object selection.

$$\mathcal{O} = \{o \in \mathcal{S} \mid \Pr_s(o) \neq 0\}. \tag{8}$$

Heuristic 1 suggests that, the objects with non-zero probabilities being answer objects form a smaller candidate set for object selection. An intuitive explanation for Heuristic 1 is that, the query result set does not change after cleaning the object o if $\Pr_s(o) = 0$, and thus, the uncertainty of answer set is not minimized. Take our sample dataset in Table 1(a) as an example. For the object o_4, we know that $\Pr_s(o_4) = 0$. After o_4 is cleaned as t_8, the quality $\kappa(skyline|\{t_8\}) = \sum_{i=1}^{3} \Pr(R_i) \log_2 \Pr(R_i) = 0.5 \log_2 0.5 + 0.48 \log_2 0.48 + 0.02 \log_2 0.02 = -1.121$, which equals to the quality in the original dataset, i.e., $\kappa(skyline(\mathcal{S}))$, as calculated in Example 3. Note that, the candidate object set $\mathcal{O}$ is $\{o_1, o_2, o_3\}$ over our sample dataset.

Moreover, we can collect those candidate objects from the object-based answer sets R with the support of ASI structure (proposed in Section 4), i.e., $\mathcal{O} = \cup_{R \in \Omega} R$. Also, the number of the object sets chosen to clean significantly drops from $(2^{|\mathcal{S}|} - 1)$ to $(2^{|\cup_{R \in \Omega} R|} - 1)$, which contributes to the efficiency enhancement of object selection problem.

However, $(2^{|\cup_{R \in \Omega} R|} - 1)$ may be still a large number if there are many object-based answer sets and/or each pair of object-based answer sets are very different (share few objects). Hence, in order to further boost performance, we proceed to exploit the important property of quality function. Heuristic 2 reveals the monotonic property of the expected quality function.

Heuristic 2. Given two uncertain object sets O_1, $O_2 \subseteq \mathcal{O}$, if O_2 is a subset of O_1, the expected quality of cleaning O_1 is not smaller than that of cleaning O_2, i.e., $\mathbb{E}[\kappa(\phi|O_1)] \geq \mathbb{E}[\kappa(\phi|O_2)]$ if $O_2 \subseteq O_1 \subseteq \mathcal{O}$.

Proof. Without loss of generality, we assume that $O_1 = \{o_1, o_2\}$ and $O_2 = \{o_1\}$ with each object o_i having two tuples $o_i^{t_1}$ and $o_i^{t_2}$. Then, according to the expected quality function in Eq. 4, we have

$$\begin{aligned}
\mathbb{E}[\kappa(\phi|O_1)] &= \Pr(o_1^{t_1}) \Pr(o_2^{t_1}) \kappa(\phi|\{o_1^{t_1}, o_2^{t_1}\}) \\
&\quad + \Pr(o_1^{t_1}) \Pr(o_2^{t_2}) \kappa(\phi|\{o_1^{t_1}, o_2^{t_2}\}) \\
&\quad + \Pr(o_1^{t_2}) \Pr(o_2^{t_1}) \kappa(\phi|\{o_1^{t_2}, o_2^{t_1}\}) \\
&\quad + \Pr(o_1^{t_2}) \Pr(o_2^{t_2}) \kappa(\phi|\{o_1^{t_2}, o_2^{t_2}\}), \\
\mathbb{E}[\kappa(\phi|O_2)] &= \Pr(o_1^{t_1}) \kappa(\phi|\{o_1^{t_1}\}) + \Pr(o_1^{t_2}) \kappa(\phi|\{o_1^{t_2}\}).
\end{aligned}$$

Let we define a function $M(o_1^{t_i})$ as $\Pr(o_2^{t_1})\kappa(\phi|\ \{o_1^{t_i}, o_2^{t_1}\})$ + $\Pr(o_2^{t_2})\kappa(\phi|\ \{o_1^{t_i},\ o_2^{t_2}\})$, i = 1, 2. Thus, we can get that,

$$\begin{aligned}
\mathbb{E}[\kappa(\phi|O_1)] - \mathbb{E}[\kappa(\phi|O_2)] &= \Pr(o_1^{t_1})(M(o_1^{t_1}) - \kappa(\phi|\{o_1^{t_1}\})) \\
&\quad + \Pr(o_1^{t_2})(M(o_1^{t_2}) - \kappa(\phi|\{o_1^{t_2}\}))
\end{aligned}$$

Based on the monotonicity of entropy function H, we can know that $H(A) \leq H(B)$ for every set $A \subseteq B$ [27]. Hence, we have $\kappa(\phi|\{o_1^{t_i},\ o_2^{t_1}\}) \geq \kappa(\phi|\{o_1^{t_i}\})$ and $\kappa(\phi|\{o_1^{t_i}, o_2^{t_2}\}) \geq \kappa(\phi|\{o_1^{t_i}\})$. As $0 < \Pr(o_2^{t_i}) \leq 1$, $i = 1, 2$, and $\Pr(o_2^{t_1}) + \Pr(o_2^{t_2}) = 1$, we can get $M(o_1^{t_i}) \geq \kappa(\phi|\{o_1^{t_i}\}), i = 1, 2$. Thus, $\mathbb{E}[\kappa(\phi|O_1)] \geq \mathbb{E}[\kappa(\phi|O_2)]$. The proof completes. □

Heuristic 2 echoes the observation in Example 4 that, query quality does not drop with an enlarging cleaning object set, as plotted in Figure 1. For instance, assume that the expected quality $\mathbb{E}[\kappa(\phi|\{o_a, o_b, o_c\})]$ has been derived. If the chosen object set $\{o_a, o_b, o_c\}$ to clean satisfies the budget constraint, it is unnecessary to calculate the expected qualities of choosing the object sets $\{o_a, o_b\}$, $\{o_a, o_c\}$, $\{o_b, o_c\}$, $\{o_a\}$, $\{o_b\}$, and $\{o_c\}$ to clean. This is because, the expected quality (i.e., $\mathbb{E}[\kappa(\phi|\{o_a, o_b, o_c\})]$) of choosing the set $\{o_a, o_b, o_c\}$ to clean is guaranteed to be not smaller than the expected quality of choosing one of its six subsets to clean, according to Heuristic 2.

Algorithm 3: Branch and Bound Algorithm (B&B)

Input: an uncertain dataset $\mathcal{S}$, a resource budget B, a query ϕ, an ASI structure $\mathcal{A}$
Output: the object set $O^\star$ with maximum expected quality $\kappa^\star$

```
/* IEQC is an improved expected quality computation function using RrB algorithm. */
κ* ⟵ −∞
O ⟵ ⋃_{R∈Ω} R                                   // Heuristic 1
Push(H, O)                                      // for Heuristic 2
while H is not empty do
    O ⟵ Pop(H)
    if O has not been visited previously then
        mark O as visited
        if Σ_{o∈O} c(o) ≤ B then
            E[κ(φ|O)] ⟵ IEQC(S, φ, O, A)
            if E[κ(φ|O)] > κ* then
                κ* ⟵ E[κ(φ|O)], O* ⟵ O
        else
            foreach O's unvisited and unpruned subset O_i
            with |O_i| = |O| − 1 do
                Push(H, O_i)
return O* and κ*
```

5.2 Algorithms

B&B **algorithm**. We propose *Branch and Bound* (B&B) algorithm for object selection problem based on Heuristic 1 and

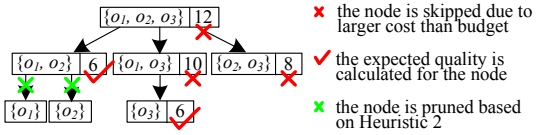


Fig. 4: The BB-tree on the dataset in Table 1(a)

Heuristic 2, where RrB algorithm is employed to calculate quality. The idea of B&B is to partition the feasible set, i.e., $\mathcal{O}$, into smaller subsets, and then to eliminate its subsets from further consideration if the cleaning cost of the current chosen object set to clean is within the budget.

Algorithm 3 presents the pseudo-code of B&B algorithm, which is an improved version of QueryClean framework in Algorithm 1. First of all, B&B pushes the whole candidate object set $\mathcal{O}$ into a stack H, where the stack H is used to maintain the *branch and bound tree* (BB-tree). Then, it pops entries of H until H is empty. When an unvisited entry O is popped, if its cleaning cost is not larger than the budget B, the expected quality $\mathbb{E}[\kappa(\phi|O)]$ of choosing O to clean is calculated, and meanwhile, the maximum expected quality $\kappa^\star$ and the corresponding chosen object set (to clean) $O^\star$ is updated if $\mathbb{E}[\kappa(\phi|O)] > \kappa^\star$ (lines 8-11). Otherwise, if the cleaning cost is larger than the budget B, all the unvisited and unpruned (by Heuristic 2) subsets of O with cardinality $|O|-1$ are pushed into H (lines 12-14). Finally, B&B returns the optimal solution $O^\star$ and $\kappa^\star$, and terminates (line 15). Note that, the correctness of B&B is guaranteed by Heuristics 1 and 2, as explained/proved in Section 5.1.

Example 7. For the P-skyline query on the sample dataset in Table 1(a), we perform B&B algorithm, where the cleaning budget B = 7, and the costs of cleaning o_1, o_2, o_3, and o_4 are 4, 2, 6, and 3, respectively.

Initially, B&B derives the candidate object set $\mathcal{O} = \{o_1, o_2, o_3\}$ for object selection according to Heuristic 1. Figure 4 shows the BB-tree on the dataset in Table 1(a). First, B&B pops the candidate object set $\mathcal{O} = \{o_1, o_2, o_3\}$ from H. The cleaning cost $c(o_1) + c(o_2) + c(o_3)$ (= 12) is larger than budget (= 7). Hence, its subsets with cardinality $(|\mathcal{O}| - 1)$ (= 2) are pushed into H, after which $H = \{\{o_1, o_2\}, \{o_1, o_3\}, \{o_2, o_3\}\}$. Next, entry $\{o_1, o_2\}$ are popped from H. As its cleaning cost is smaller than budget, B&B computes the expected quality of choosing $\{o_1, o_2\}$ to clean, i.e., $\mathbb{E}[\kappa(skyline|\{o_1, o_2\})]$, based on RrB algorithm. Then, it prunes all the subsets of $\{o_1, o_2\}$ using Heuristic 2.

In the following, entry $\{o_1, o_3\}$ is popped. Its cleaning cost is larger than budget. Thus, the unvisited subsets of $\{o_1, o_3\}$ but not included in the evaluated entry $\{o_1, o_2\}$ are pushed into H. Thereafter, $H = \{\{o_3\}, \{o_2, o_3\}\}$. Hence, entry $\{o_3\}$ is popped, and $\mathbb{E}[\kappa(skyline|\{o_3\})]$ is calculated. Next, entry $\{o_2, o_3\}$ is skipped by B&B due to its larger cleaning cost than budget. Meanwhile, for the two subsets of $\{o_2, o_3\}$, $\{o_2\}$ is pruned by $\{o_1, o_2\}$, and $\{o_3\}$ has been evaluated. Thus, there is no entry added to H, and H is empty. Finally, B&B stops by delivering the optimal solution with $\max(\mathbb{E}[\kappa(skyline|\{o_1, o_2\})], \mathbb{E}[\kappa(skyline|\{o_3\})])$.

Greedy **algorithm**. As analyzed in Section 2, finding an optimal solution for this optimization problem has high complexity. To this end, we develop Greedy algorithm to handle the object selection problem, on top of the candidate set $\mathcal{O}$ derived by Heuristic 1. The algorithm selects one cleaning object $o^\star$ from $\mathcal{O}$ with the maximum unit-cost expected quality every time until the budget is used up. The solution returned by Greedy is at a performance guarantee $(1 - 1/e)$, as proved in [29].

Algorithm 4: Greedy Algorithm

Input: an uncertain dataset $\mathcal{S}$, a resource budget B, a query predicate ϕ, an ASI structure $\mathcal{A}$
Output: the object set $O^\star$ with maximum expected quality $\kappa^\star$

1: $\kappa^\star \longleftarrow -\infty$
2: $\mathcal{O} \longleftarrow \bigcup_{R\in\Omega} R$ // Heuristic 1
3: **while** $\mathcal{O} \neq \varnothing$ **do**
4: $\quad \kappa_c \longleftarrow -\infty$, flag = *false*
5: $\quad$ **foreach** object $o_i \in \mathcal{O}$ with $\sum_{o\in O^\star} c(o) + c(o_i) \leq B$ **do**
6: $\quad\quad$ flag= *true*
7: $\quad\quad \mathbb{E}[\kappa, (O^\star + \{o_i\})] \longleftarrow$ IEQC$(\mathcal{S}, \phi, O^\star + \{o_i\}, \mathcal{A})$
8: $\quad\quad$ **if** $\frac{\mathbb{E}[\kappa,(O^\star+\{o_i\})]}{\sum_{o\in O^\star} c(o)+c(o_i)} > \kappa_c$ **then**
9: $\quad\quad\quad \kappa_c \longleftarrow \frac{\mathbb{E}[\kappa,(O^\star+\{o_i\})]}{\sum_{o\in O^\star} c(o)+c(o_i)}$, $o^\star \longleftarrow o_i$
10: $\quad$ **if** flag = *false* **then**
11: $\quad\quad$ **break**
12: $\quad \mathcal{O} \longleftarrow \mathcal{O} - \{o^\star\}$
13: $\quad$ **if** $\mathbb{E}[\kappa, (O^\star + \{o^\star\})] > \kappa^\star$ **then**
14: $\quad\quad \kappa^\star \longleftarrow \mathbb{E}[\kappa, (O^\star + \{o^\star\})]$, $O^\star \longleftarrow O^\star + \{o^\star\}$
15: **return** $O^\star$ and $\kappa^\star$

The pseudo-code of Greedy algorithm is depicted in Algorithm 4. For each object o in a candidate set $\mathcal{O}$, if the accumulated cleaning cost of the selected objects in $O^\star$ plus the cleaning cost of o is not larger than the budget, IEQC function is invoked to calculate the expected quality of choosing $(O^\star + \{o\})$ to clean using RrB algorithm. Then, the object $o^\star$ is updated as o_i if choosing $(O^\star + \{o_i\})$ to clean has a larger unit-cost expected quality than before, and the maximal unit-cost expected quality is updated accordingly (lines 5-9). Otherwise, if there is no object satisfying the accumulated cleaning cost of the selected objects in $O^\star$ plus the cleaning cost of o is not larger than the budget, Greedy exits the object selection, and terminates shortly (lines 10-11). If the expected quality when adding $o^\star$ into set $O^\star$ is larger than $\kappa^\star$, then $\kappa^\star$ is updated, and $o^\star$ is inserted into $O^\star$ (lines 13-14). In the end, Greedy returns the chosen object set $O^\star$ and the corresponding expected quality $\kappa^\star$ (line 15).

HSample **algorithm**. We also present a novel sampling algorithm to tackle the object selection problem. Specifically, the sampling technique is to capture a group of chosen object sets to clean from the power set of $\mathcal{O}$, and to return the sampled object set having the maximum expected quality, with the result accuracy guaranteed by Lemma 3.

Lemma 3. **(Accuracy guarantee)**. Let $Y_1, \cdots, Y_m \in \mathcal{Y}$ be a set of m (≥ 1) independent random variables denoting possible chosen object sets to clean (i.e., $Y_i \subseteq \mathcal{O}$ for $i \in [1, m]$), κ be the original quality of $\phi(q; \mathcal{S})$, and $f(y_1, \cdots, y_i, \cdots, y_m)$ be the maximum function of expected quality when choosing object sets $y_1, \cdots, y_m$ to clean respectively (i.e., $f(y_1, \cdots, y_m) = \max_{i\in[1,m]} \mathbb{E}[\kappa(\phi|y_i)]$). It is confirmed that

$$|f(y_1, \cdots, y_i, \cdots, y_m) - f(y_1, \cdots, y_i', \cdots, y_m)| \leq -\kappa,$$

for all $i \in [1, m]$ and $\forall\, y_1, \cdots, y_m, y_i' \in \mathcal{Y}$. Let $f(S)$ denote $f(Y_1, \cdots, Y_m)$, then, for all $\epsilon > 0$, the inequality below holds.

$$\Pr[|f(S) - \mathbb{E}[f(S)]| \geq \epsilon] \leq 2e^{\frac{-2\epsilon^2}{m\kappa^2}} \quad (9)$$

Proof. First, we prove the inequality $|f(y_1, \cdots, y_i, \cdots, y_m) - f(y_1, \cdots, y_i', \cdots, y_m)| \leq -\kappa$ holds. According to the mono-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2018.2805824, IEEE Transactions on Knowledge and Data Engineering

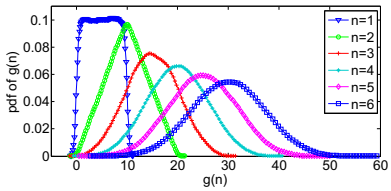


Fig. 5: The cost function $g(n)$ of cleaning n objects

tonic property of our expected quality (proved in Heuristic 2), we can conclude that $\mathbb{E}[\kappa(\phi|y_i)] \geq \mathbb{E}[\kappa(\phi)] = \kappa$ for all $i \in [1, m]$, where y_i is a chosen object set to clean. Since $f(y_1, \cdots, y_m)$ = $\max_{i\in[1,m]} \mathbb{E}[\kappa(\phi|y_i)]$, we can derive that $f(y_1, \cdots, y_m) \geq \mathbb{E}[\kappa(\phi|y_i)]$ for all $i \in [1, m]$. Thus, we have $f(y_1, \cdots, y_m) \geq \kappa$, meaning that κ is the lower bound of $f(y_1, \cdots, y_m)$. On the other hand, it is known that, the highest score of expected quality is zero based on the properties of entropy function, i.e., $f(y_1, \cdots, y_m) \leq 0$. Hence, function $f(y_1, \cdots, y_m)$ is bounded by $[\kappa, 0]$, and thus $|f(y_1, \cdots, y_i, \cdots, y_m) - f(y_1, \cdots, y_i', \cdots, y_m)| \leq -\kappa$. As a result, Eq. 9 can be derived based on McDiarmid's inequality [30]. The proof completes. □

Moreover, if we partition the solution space (i.e., the $(2^{|\mathcal{O}|}-1)$ possible chosen object sets to clean) into $|\mathcal{O}|$ clusters (denoted as $\Delta_1, \Delta_2, \cdots, \Delta_{|\mathcal{O}|}$) such that every partition Δ_i satisfies that $\Delta_i = \{O \subseteq \mathcal{O} \mid |O| = i\}$, we find that the optimal solution can be captured in a much smaller sample space. In other words, we could capture samples only within several Δ_is, instead of the whole space within all the Δ_is. It is observed by analyzing the property of a cost function, termed as $g(n)$, in the factor of the number n of chosen objects to clean.

To be more specific, if h_o denotes the cost function of cleaning an uncertain object o (e.g., the monetary cost), the total cost function $g(n)$ of cleaning n uncertain objects (stored in an object set $\mathcal{S}_c$) is equal to $\sum_{o\in\mathcal{S}_c} h_o$. Figure 5 illustrates the probability density function (pdf) of $g(n)$ when $n = 1, 2, 3, 4, 5$, and 6 respectively, where h_o is supposed to follow *Uniform* distribution within the range $[1, 10]$. It is observed that, with the growth of n, $g(n)$ tends to *Gaussian* distribution with mean $n \cdot \mu$. It is worth noting that, other practical distributions for the cleaning cost function (i.e., h_o) are welcomed to utilize. Whatever the distribution of h_o is, the accumulated cost function g is able to support object selection.

As an example, suppose budget $B = 16$, we can capture solutions (i.e., the chosen object sets to clean) from the clusters near to $\Delta_{\frac{B}{\mu}}$ including $\Delta_2, \Delta_3, \Delta_4$, and Δ_5 (w.r.t. $g(2)$, $g(3)$, $g(4)$, and $g(5)$) by discarding clusters Δ_1 and Δ_6 (w.r.t. $g(1)$ and $g(6)$) for object selection problem. The motivation is straightforward, because $g(1)$ has a very low probability (near to zero) to cover the budget value 16, as shown in Figure 5. So does $g(6)$. Hence, we can conclude that, clusters Δ_1 and Δ_6 contain the optimal solution with a very low confidence. Generally speaking, if i of a partition Δ_i is much larger than $\lfloor\frac{B}{\mu}\rfloor$, the object selection solutions from Δ_i cannot satisfy the budget constraint. If i is much smaller than $\lfloor\frac{B}{\mu}\rfloor$, the object selection solutions from Δ_i may reserve some budget, indicating that more objects can be chosen to clean in order to further improve the quality based on Heuristic 2. Thus, it is most likely that the cluster Δ_i with i near to $\frac{B}{\mu}$ do contain the optimal solution.

As a result, we propose Histogram Sample (HSample) algorithm with its pseudo-code depicted in Algorithm 5. Specifically, HSample first gets the candidate object set $\mathcal{O}$. Then, it captures m samples from clusters $\Delta_{\lfloor\frac{B}{\mu}\rfloor-2}$, $\Delta_{\lfloor\frac{B}{\mu}\rfloor-1}$, $\Delta_{\lfloor\frac{B}{\mu}\rfloor}$, $\Delta_{\lfloor\frac{B}{\mu}\rfloor+1}$, and $\Delta_{\lfloor\frac{B}{\mu}\rfloor+2}$ (if they exist) (line 3). Next, the sampled object set O with the maximum expected quality is returned by HSample.

Algorithm 5: HSample Algorithm

Input: an uncertain dataset $\mathcal{S}$, a resource budget B, a query ϕ, an ASI structure $\mathcal{A}$, a sample size m, the average cleaning cost μ

Output: the objects $O^\star$ with the maximum expected quality $\kappa^\star$

1: $\kappa^\star \longleftarrow -\infty$
2: $\mathcal{O} \longleftarrow \bigcup_{R\in\Omega} R$ // Heuristic 1
3: sample m object sets to clean from clusters $\Delta_{\lfloor\frac{B}{\mu}\rfloor-2}$, $\Delta_{\lfloor\frac{B}{\mu}\rfloor-1}$, $\Delta_{\lfloor\frac{B}{\mu}\rfloor}$, $\Delta_{\lfloor\frac{B}{\mu}\rfloor+1}$, and $\Delta_{\lfloor\frac{B}{\mu}\rfloor+2}$
4: **foreach** sampled object set O satisfying $\sum_{o\in O} c(o) \leq B$ **do**
5: $\quad \mathbb{E}[\kappa(\phi|O)] \longleftarrow$ IEQC$(\mathcal{S}, \phi, O, \mathcal{A})$
6: $\quad$ **if** $\mathbb{E}[\kappa(\phi|O)] > \kappa^\star$ **then**
7: $\quad\quad \kappa^\star \longleftarrow \mathbb{E}[\kappa(\phi|O)]$, $O^\star \longleftarrow O$
8: **return** $O^\star$ and $\kappa^\star$

Time complexity analysis. Let ω be the collection of the tuple-based answer sets, B be the cleaning budget, and μ be the average cost value for cleaning an uncertain object. Then, the expected quality computation cost $\beta = I^{\lfloor\frac{B}{\mu}\rfloor}\cdot|\omega|\cdot\lfloor\frac{B}{\mu}\rfloor$, as discussed in Section 4. Therefore, the exact B&B algorithm takes $O(2^{|\mathcal{O}|}\cdot\beta)$ time in finding the optimal solution, if $\mathcal{O}$ is the candidate set for the object selection derived based on Heuristic 1. Greedy has the complexity of $O(|\mathcal{O}|^2 \cdot \beta)$ time, and the time complexity of HSample is $O(m \cdot \beta)$ if m samples are captured.

6 Experimental Evaluation

In this section, we present a comprehensive experimental evaluation. All algorithms were implemented in C++, and all experiments were conducted on an Intel Core i7 Duo 3.60GHz PC with 16GB RAM, running Microsoft Windows 7 Professional Edition.

In our experiments, we employ both real and synthetic data sets. For real data sets, the used cars data set $CarDB$ [4] and the Forest Covertype data set $Forest$ are utilized. $CarDB$ refers to P-skyline query, and $Forest$ corresponds to P-kNN and P-range queries, considering the characteristics of the datasets and the queries. $CarDB$ is gathered from a used car website over a 10-month period (from 2006/1 to 2006/11) with 45,311 cars. We use *two* attributes *price* and *mileage* for every car. $Forest$ is originally obtained from US Geological Survey (USGS) and USFS data. We utilize 250,000 records with *ten* attributes. Note that, the used car information in $CarDB$ is uncertain in data integration, and the records in $Forest$ contains uncertainty due to transmission failure or inconsistent values received from several sensors. We generate uncertain objects by simulating 1-5 tuples randomly with probabilities under Uniform distribution, following the methodology in [3], [5], [31]. In addition, the tuple probabilities could be inferred by machine learning techniques, such as Bayesian network [18]. For synthetic data sets, we first create the centers of all uncertain objects o_t using the benchmark data generator proposed in [32]. Then, in order to generate an uncertain object o, we follow the methodology in [3], [5], [31], and determine the tuples within the region centered at the location o_t under Uniform and Gaussian distributions.

In our experiments, we explore several factors, as summarized in Table 4 where the default values are shown in bold. In order to model the cleaning problem, we create a cleaning cost for each

TABLE 4: Parameter Ranges and Default Values

Parameter	Range
the number $\|\mathcal{S}\|$ of objects	50K, **100K**, 200K, 400K
budget B	10, 15, **20**, 25
sample size m	10, 50, **100**, 200, 500

TABLE 5: ASI Construction Cost on *CarDB*

Cardinality	**10K**	**20K**	**30K**	**45311**
$\|\mathcal{O}\|$	10	13	15	23
$\|\omega\|$	320	1296	1680	55024
Space size (KB)	86.416	358.632	460.452	25461.232
Constr. time (s)	0.022	0.098	0.794	429.752

uncertain object, which is an integer and uniformly distributed in the range of [1, 10]. In addition, for P-kNN and P-range queries, we report the average performance of 50 queries with query objects captured from dataset randomly, where k is 10 and range threshold is 10 by default, and the Euclidean distance is used. It is noteworthy that, there is no existing algorithm that could directly solve the problem studied in this paper.

6.1 Evaluation on ASI Construction

In this subsection, we study the performance of ASI construction using the real CarDB data set.

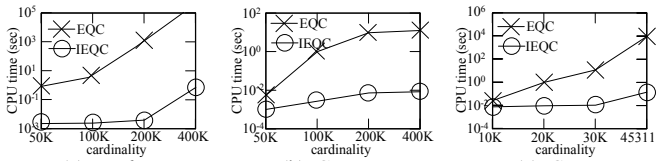
First, the experimental results when dataset cardinality varies are reported in Table 5. $|\mathcal{O}|$ is the size of the candidate set pruned by Heuristic 1, which is closely related to the size of the possible worlds we should evaluate for ASI construction. $|\omega|$ is the number of achieved tuple-based answer sets. The construction time (in the unit of seconds) consists of the processing time of obtaining the tuple-based answer sets and constructing ASI based on those tuple-based answer sets. As analyzed in Section 4.2, the time ascends exponentially with the increasing of dataset cardinality $|\mathcal{S}|$ (as well as the candidate set size $|\mathcal{O}|$). This is because, the set of the qualified objects included in $\mathcal{O}$ (that cannot be pruned) becomes larger with the growth of $|\mathcal{S}|$. Thus, the possible worlds we need to evaluate for ASI construction incur much more overhead. In addition, one can observe that, the space cost of ASI (in the unit of Kbytes) grows approximately linearly with $|\omega|$, which is consistent with the analysis in Section 4.2.

Second, in order to further verify the effect of *heuristic* ASI index on the performance of QueryClean, in which ASI is constructed only based on the obtained tuple-based answer sets from a fraction of possible worlds (instead of all possible worlds). Table 6 shows the corresponding experimental results when changing the number of the evaluated possible worlds (i.e., #PWs) on *CarDB*, where the results on the last column correspond to the exact ASI index. Let $O_e^\star$ ($O_h^\star$) be the selected uncertain object set for cleaning by B&B derived based on exact (heuristic) ASI index. The Jaccard index in the table is equal to $\frac{O_e^\star \cap O_h^\star}{O_e^\star \cup O_h^\star}$.

As expected, with the increasing of evaluated possible worlds, the number of tuple-based answer sets, the space size, the construction time of ASI, and the execution time of B&B for selecting objects grow linearly. The accuracy (i.e., Jaccard index) ascends due to more evaluated possible worlds. The candidate set $\mathcal{O}$ pruned by Heuristic 1 has a constant size, since it is only dependent of the dataset, not related to the evaluated possible worlds. Note that, compared with the exact algorithm, it needs less time and space for the heuristic method to construct ASI index. Moreover, our framework QueryClean (i.e., selecting objects via B&B algorithm) on the top of the exact and heuristic ASI index has comparable performance, when evaluating more and more possible worlds. As an example, when #PWs = 75,000,000, the Jaccard index is 1.0. Hence, in this case, it is sufficient to evaluate 75,000,000 possible worlds for object selection, instead of all possible worlds in much extra overhead. We plan to investigate how to further improve the heuristic method for ASI construction in our future work.

TABLE 6: Heuristic Approach Study for ASI Construction

#PWs	**1M**	**25M**	**50M**	**75M**	**100M**	**Exact**
$\|\mathcal{O}\|$	23	23	23	23	23	23
$\|\omega\|$	952	9532	18975	27513	35342	55024
Space size (KB)	384.944	3845.148	7871.224	11660.384	15396.608	25461.232
Constr. time (s)	2.514	61.293	124.610	184.673	250.255	429.752
B&B time (s)	18.275	162.000	378.130	736.840	1107.500	2012.700
Jaccard index	0.400	0.400	0.400	1.000	1.000	1.000



(a) *Uniform* (b) *Gaussian* (c) *CarDB*

Fig. 6: Evaluation on quality computation techniques

6.2 Efficiency of Quality Computation

This set of experiments compares the performance of expected quality computation *with* and *without* optimizations (i.e., ASI with pruning strategies). Specifically, EQC (mentioned in Section 4.1) corresponds to the one *without* optimizations, and IEQC is the one *with* optimizations, which employs RrB in Algorithm 2 using ASI with pruning strategies. In order to compare the two algorithms, we conduct 100 expected quality computations in which 100 uncertain objects (to clean) are randomly selected from the datasets, and report the average CPU time.

Figure 6 plots the experimental results of expected quality computation w.r.t. P-skyline queries when changing the dataset cardinality $|\mathcal{S}|$. It is observed that, IEQC outperforms EQC in several orders of magnitude. The good performance of IEQC contributes to RrB algorithm, which utilizes ASI structure to save the large cost of evaluating many probabilistic queries over a large number of possible worlds. Besides, the computation cost increases with the growth of $|\mathcal{S}|$, especially for EQC. This is because, with the increasing of $|\mathcal{S}|$, the number of possible worlds grows exponentially, incurring more overhead for EQC. Due to its remarkable performance, we employ IEQC for expected quality computation in the rest of experiments.

6.3 Performance of QueryClean

In this subsection, we evaluate the performance of our framework QueryClean (i.e., object selection algorithms). The execution time and the query quality are reported for each algorithm, where the query quality on the original data set is given for comparison.

We first explore the influence of budget B on the performance of object selection algorithms. The experimental results are depicted in Figures 7(a) through 7(e) for P-skyline queries, P-kNN search, and P-range retrieval. It is observed that, B&B is inferior to Greedy and HSample in terms of time cost in most cases. For execution time and quality, Greedy is slightly better than HSample (where sample size is 100) in some cases. With the growth of budget B, the query quality improves gradually, and the time cost becomes larger accordingly. The reason is that, as budget B turns larger, there is enough resource to choose more uncertain objects to clean, resulting in higher query quality yet more time cost. Note that, for every dataset, with the support of a constant

TABLE 7: Average precision of Greedy and HSample

	Uniform (P-skyline)	Gaussian (P-skyline)	CarDB (P-skyline)	Forest (P-kNN)	Forest (P-range)
Greedy	0.585	1.000	0.958	0.982	0.916
HSample	0.534	0.750	0.880	0.611	0.924

TABLE 8: Pruning Rate of Heuristic 2

	Uniform	Gaussian	CarDB	Forest
P-skyline	0.090	0.012	0.001	Not applicable
P-kNN	0.130	0.118	Not applicable	0.129
P-range	0.240	0.331	Not applicable	0.528

ASI index, it needs at most 10 seconds for the two approximate algorithms to select the objects for cleaning over the dataset of 100,000 uncertain objects.

We report the average precision of two approximate algorithms in Table 7. It is defined as the average ratio of the quality derived by Greedy or HSample to the quality from the exact B&B algorithm. We can observe that, the average precision of both Greedy and HSample is not lower than 0.5 (even up to 1.0) for each query. In addition, Table 8 lists the pruning rate of Heuristic 2 used in B&B algorithm for $B = 20$. The pruning rate is defined as the ratio of the number of the object sets that do not need the expected quality computation (i.e., pruned by Heuristic 2) to the total number (i.e., $2^{|\mathcal{O}|} - 1$) of the possible object sets to clean. Note that, Heuristic 2 is effective for all cases.

Then, we investigate the effect of standard deviation of Gaussian distribution on the performance of QueryClean. Figure 7(f) plots the corresponding experimental results with the standard deviation σ varying from 250 to 1000 (500 by default), where the distribution mean μ is 5000. One can observe that, the algorithms are not very sensitive to the standard deviation. As expected, the exact B&B algorithm achieves the highest query quality at the most time cost, and Greedy performs slightly better than HSample in all cases.

Next, we inspect the impact of sample size m on the performance of HSample algorithm, with the corresponding results illustrated in Figure 8. It is observed that, the query quality achieved by HSample is significantly higher than the original query quality. In addition, the query quality and the time cost increase with the growth of m. This is because, when more samples are captured, it incurs more overhead for HSample in order to find out the optimal solution, while it obtains the higher expected quality.

In summary, B&B algorithm can achieve the optimal solution with the relatively higher processing cost. As contrast, Greedy and HSample get comparable near-optimal solutions with much lower costs. Moreover, for the scenarios with a high requirement for efficiency, HSample is a better choice, via tuning the sample size m to balance the tradeoff between efficiency and quality.

6.4 Case Study

To further confirm the effectiveness of QueryClean, we extract a group of used cars[3]. For every car, the price and mileage values from different car owners form an uncertain object. Table 9 lists the original P-skyline objects of the 20 cars having non-zero probabilities. It also gives the P-skyline objects after cleaning the selected objects as one of its tuples randomly, under resource budget B being 10 and 20 respectively. Note that, the objects chosen by QueryClean for cleaning are denoted as "$*$" ("$\star$") when the budget B is 10 (20). For illustration purpose, those 20 cars are plotted in Figure 9 with the chosen objects highlighted in red color.

3. Available at http://www.carsdb.com/en/listings/used_cars.

TABLE 9: The P-skyline of a Real Used Car Dataset

Object	Original P-skyline	P-skyline (B = 10)	P-skyline (B = 20)
$*\star$ Nissan AD Van (NAV) 2010	✓ 0.222		
$*\star$ Toyota Passo (TP) 2010	✓ 0.417	✓ 0.429	
Honda Fit (HF) 2010	✓ 0.500	✓ 0.571	✓ 0.429
Mitsubishi RVR (MR) 2010	✓ 0.004	✓ 0.008	
$\star$ Suzuki Wagon R (SWR) 2010	✓ 0.623	✓ 0.690	✓ 1.000
Toyota Harrier (TH) 2000			
Honda Accord (HA) 2011	✓ 0.125	✓ 0.250	
Toyota Axio (TA) 2012	✓ 0.103	✓ 0.210	
Honda Insight (HA) 2010			
Mitsubishi Colt Plus (MCP) 2011	✓ 0.377	✓ 0.595	
Toyota Alphard (TAl) 2002			
$*\star$ Surf SSRX (SS) 2002	✓ 1.000	✓ 1.000	✓ 1.000
Toyota Kluger (TK1) 2001			
Toyota Kluger (TK4) 2004			
Toyota Wish (TW) 2003	✓ 0.054	✓ 0.107	
Mazda Familia (MF) 2010	✓ 0.102		
Mitsubishi Minicab Truck (MMT) 2007	✓ 1.000	✓ 1.000	✓ 1.000
Toyota Mark X (TMX) 2004	✓ 0.001	✓ 0.004	
Honda CR-V (HCV) 2005			
Toyota RAV4 (TR) 2006			
Size of P-skyline set	13	11	4
Quality of P-skyline query	-6.093	-5.317	-0.985

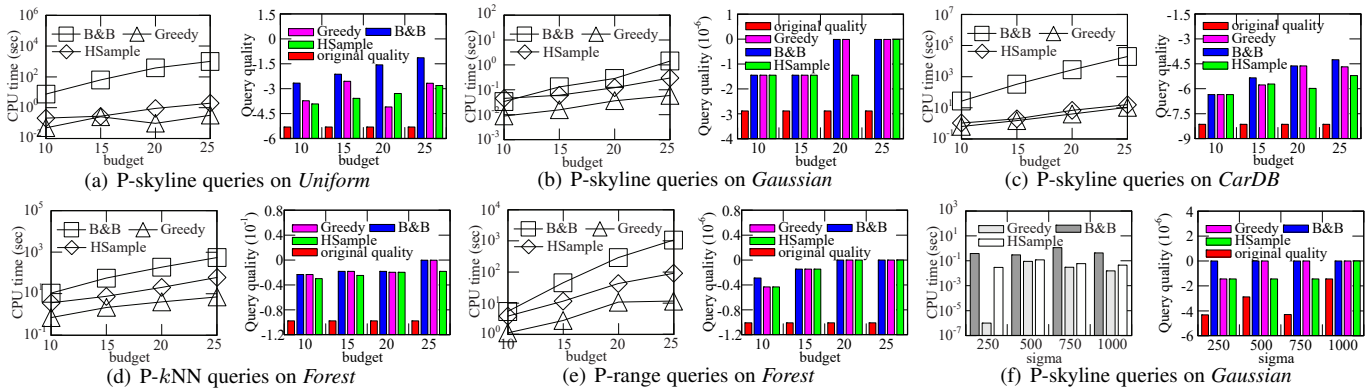
It is observed that, the original P-skyline objects cover a large part of the whole dataset (i.e., 13/20), and many of them have low probabilities, indicating more noise. When budget B equals 10, the P-skyline size decreases to 11. When $B = 20$, the P-skyline size drops to 4, with high probabilities being P-skyline objects. It means that, the uncertainty of the result set reduces significantly after cleaning objects. As depicted in Table 9, the quality after cleaning objects is higher than the quality without cleaning objects. Moreover, the larger the budget, the higher the quality. Hence, QueryClean helps to minimize the query result uncertainty (and hence improve query quality), which cannot be achieved by existing algorithms of probabilistic queries.

7 Related Work

Querying uncertain data. There are mainly two types of probabilistic spatial queries, i.e., (i) *probabilistic preference queries*, such as P-skyline query [3], [4], [5] and probabilistic top-k query [11], [12], [33], [31], [13]; and (ii) *probabilistic similarity queries*, e.g., probabilistic threshold nearest neighbor search [6], [7], [8], [9], [10]. Based on existing probabilistic query processing approaches, it is difficult for users to acquire useful insights, and make correct decisions from the answer sets with much noise. Thus, we aim to improve probabilistic query quality, different from the above efforts on probabilistic query efficiency.

Moreover, we focus on the general probabilistic query *without* a probability threshold, which is much harder than the one *with* a probability threshold [34]. As a result, the techniques for the probabilistic queries with probability thresholds cannot facilitate our work in this paper, such as UP-index [35], PS-index [36], the probabilistic matrix index (PMI) [37], the confident-based similarity function [38], the pruning rules based on probabilistic signature (connectivity) and feature [39], etc.

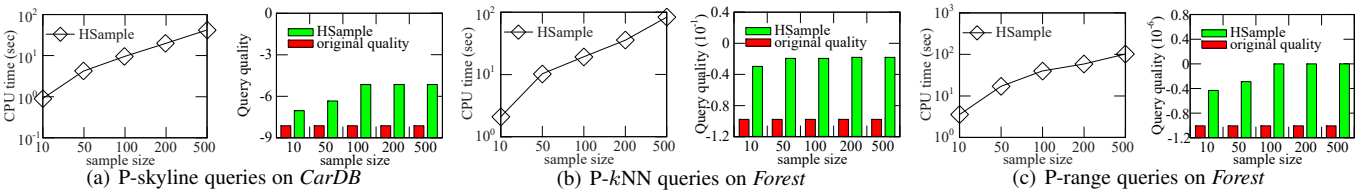
Quality optimizations for probabilistic queries. Cheng et al. [16] explore region and max queries under a limited amount of resources, in order to achieve the best improvement in the quality of query results. They define a quality metric (PWS-quality) to evaluate the uncertainty of all possible query results. Then, Cheng et al. [40] present cleaning processes based on sampling strategies. In [17], they extend the method in [16] to support probabilistic top-k queries. In contrast, Lin et al. [41] introduce data acquisition



(a) P-skyline queries on *Uniform* (b) P-skyline queries on *Gaussian* (c) P-skyline queries on *CarDB*

(d) P-kNN queries on *Forest* (e) P-range queries on *Forest* (f) P-skyline queries on *Gaussian*

Fig. 7: Performance evaluation on QueryClean



(a) P-skyline queries on *CarDB* (b) P-kNN queries on *Forest* (c) P-range queries on *Forest*

Fig. 8: Evaluation on HSample algorithm vs. m

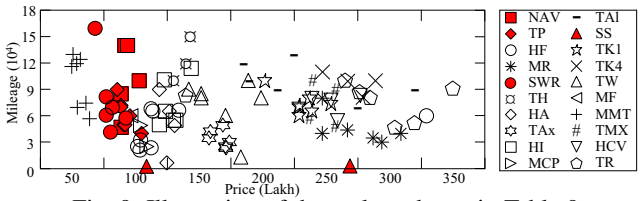


Fig. 9: Illustration of the real used cars in Table 9

to improve the quality of the result for P-kNN search. They define the quality of P-kNN query result as the probability of the most probable P-kNN answer, in which a larger quality leads to a more definite P-kNN query result, and can offer more insights. Xu et al. [42] study the best deterministic representation of data so as to continue using existing end-applications that take only deterministic input. In addition, some cleaning studies concentrate on entity resolution (ER) problem for SQL-like queries [2], [24], [43], [44], although their focus is different from our framework QueryClean that aims to seek the beneficial uncertain objects to clean with the maximum expected query quality.

It is noteworthy that, the existing approaches cannot benefit the quality optimization of probabilistic skyline and similarity queries via choosing uncertain objects to clean. This is because, query quality optimization is query-dependent, and the skyline query is more difficult to handle than the region/max query, the top-k query, and SQL-like queries, even in deterministic databases.

8 Conclusions

In this paper, we propose a novel optimization framework, namely, QueryClean, to improve the quality of probabilistic skyline and similarity queries by selecting a group of uncertain objects to clean. We boost the efficiency of QueryClean from two aspects, i.e., accelerating quality computation and optimizing object selection. We develop an efficient RrB algorithm using an effective structure, i.e., ASI, which has the capability of directly calculating the expected query quality of choosing a set of objects to clean, instead of processing probabilistic queries multiple times. In addition to one exact B&B algorithm, we present Greedy and HSample algorithms with two effective pruning heuristics to tackle object selection problem. Extensive experiments on both real and synthetic data sets demonstrate the performance of QueryClean. In the future, we intend to study how to further improve query quality over uncertain databases, e.g., using crowdsourcing techniques.

Acknowledgments. This work was supported in part by the 973 Program of China under Grant No. 2015CB352502, NSFC Grants under No. 61522208 and 61379033, and the NSFC-Zhejiang Joint Fund under Grant No. U1609217. Wei Wang was supported by ARC DP 170103710, D2D CRC Grants DC25002 and DC25003. Yunjun Gao is the corresponding author of this work.

References

[1] C. C. Aggarwal and P. S. Yu, "A survey of uncertain data algorithms and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 5, pp. 609–623, 2009.

[2] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo, "A sample-and-clean framework for fast and accurate query processing on dirty data," in *SIGMOD*, pp. 469–480, 2014.

[3] M. J. Atallah and Y. Qi, "Computing all skyline probabilities for uncertain data," in *PODS*, pp. 279–287, 2009.

[4] X. Liu, D.-N. Yang, M. Ye, and W.-C. Lee, "U-skyline: A new skyline query for uncertain databases," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 945–960, 2013.

[5] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *VLDB*, pp. 15–26, 2007.

[6] T. Bernecker, T. Emrich, H.-P. Kriegel, N. Mamoulis, M. Renz, and A. Züfle, "A novel probabilistic pruning approach to speed up similarity queries in uncertain databases," in *ICDE*, pp. 339–350, 2011.

[7] J. Chen, R. Cheng, M. Mokbel, and C.-Y. Chow, "Scalable processing of snapshot and continuous nearest-neighbor queries over one-dimensional uncertain data," *VLDB J.*, vol. 18, no. 5, pp. 1219–1240, 2009.

[8] R. Cheng, J. Chen, M. Mokbel, and C.-Y. Chow, "Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data," in *ICDE*, pp. 973–982, 2008.

[9] X. Xie, R. Cheng, M. L. Yiu, L. Sun, and J. Chen, "UV-diagram: A voronoi diagram for uncertain spatial databases," *VLDB J.*, vol. 22, no. 3, pp. 319–344, 2013.

[10] P. Zhang, R. Cheng, N. Mamoulis, M. Renz, A. Zufle, Y. Tang, and T. Emrich, "Voronoi-based nearest neighbor search for multi dimensional uncertain databases," in *ICDE*, pp. 158–169, 2013.

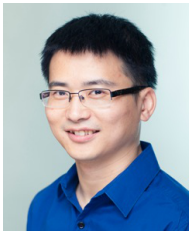
[11] M. Hua, J. Pei, W. Zhang, and X. Lin, "Efficiently answering probabilistic threshold top-k queries on uncertain data.," in *ICDE*, pp. 1403–1405, 2008.

[12] M. Hua, J. Pei, W. Zhang, and X. Lin, "Ranking queries on uncertain data: A probabilistic threshold approach," in *SIGMOD*, pp. 673–686, 2008.
[13] M. A. Soliman, I. F. Ilyas, and K. Chen-Chuan Chang, "Top-k query processing in uncertain databases," in *ICDE*, pp. 896–905, 2007.
[14] T. Friedman and M. Smith, *Measuring the business value of data quality*. Gartner, 2011.
[15] S. Adelman, L. T. Moss, and M. Abai, *Data Strategy*. Addison-Wesley, 2005.
[16] R. Cheng, J. Chen, and X. Xie, "Cleaning uncertain data with quality guarantees," in *VLDB*, pp. 722–735, 2008.
[17] L. Mo, R. Cheng, X. Li, D. W. Cheung, and X. S. Yang, "Cleaning uncertain data for top-k queries," in *ICDE*, pp. 134–145, 2013.
[18] S. De, Y. Hu, M. V. Vamsikrishna, Y. Chen, and S. Kambhampati, "BayesWipe: A scalable probabilistic framework for cleaning bigdata," *arXiv preprint arXiv:1506.08908*, 2015.
[19] Y. Yang, N. Meneghetti, R. Fehling, Z. H. Liu, and O. Kennedy, "Lenses: An on-demand approach to ETL," *PVLDB*, vol. 8, no. 12, pp. 1578–1589, 2015.
[20] E. Ciceri, P. Fraternali, D. Martinenghi, and M. Tagliasacchi, "Crowdsourcing for top-k query processing over uncertain data," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 1, pp. 41–53, 2016.
[21] P. C. Arocena, B. Glavic, G. Mecca, R. J. Miller, P. Papotti, and D. Santoro, "Messing up with BART: Error generation for evaluating data-cleaning algorithms," *PVLDB*, vol. 9, no. 2, pp. 36–47, 2015.
[22] Z. Khayyat, I. F. Ilyas, A. Jindal, S. Madden, M. Ouzzani, P. Papotti, J.-A. Quiané-Ruiz, N. Tang, and S. Yin, "BigDansing: A system for big data cleansing," in *SIGMOD*, pp. 1215–1230, 2015.
[23] X. Wang, X. L. Dong, and A. Meliou, "Data X-Ray: A diagnostic tool for data errors," in *SIGMOD*, pp. 1231–1245, 2015.
[24] M. Bergman, T. Milo, S. Novgorodov, and W.-C. Tan, "Query-oriented data cleaning with oracles," in *SIGMOD*, pp. 1199–1214, 2015.
[25] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye, "KATARA: A data cleaning system powered by knowledge bases and crowdsourcing," in *SIGMOD*, pp. 1247–1261, 2015.
[26] B. Groz and T. Milo, "Skyline queries with noisy comparisons," in *PODS*, pp. 185–198, ACM, 2015.
[27] S. Fujishige, *Submodular functions and optimization*, vol. 58. Elsevier, 2005.
[28] J. Liu, H. Zhang, L. Xiong, H. Li, and J. Luo, "Finding probabilistic k-skyline sets on uncertain data," in *CIKM*, pp. 1511–1520, ACM, 2015.
[29] A. Krause and C. Guestrin, "A note on the budgeted maximization of submodular functions," 2005.
[30] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2012.
[31] M. A. Soliman, I. F. Ilyas, and K. C.-C. Chang, "Probabilistic top-k and ranking-aggregate queries," *ACM Trans. Database Syst.*, vol. 33, no. 3, p. 13, 2008.
[32] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," in *ICDE*, pp. 421–430, 2001.
[33] C. Re, N. Dalvi, and D. Suciu, "Efficient top-k query evaluation on probabilistic data," in *ICDE*, pp. 886–895, 2007.
[34] X. Zhou, K. Li, G. Xiao, Y. Zhou, and K. Li, "Top k favorite probabilistic products queries," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 10, pp. 2808–2821, 2016.
[35] F. Angiulli and F. Fassetti, "Indexing uncertain data in general metric spaces," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 9, pp. 1640–1657, 2012.
[36] Y. Tong, X. Zhang, C. C. Cao, and L. Chen, "Efficient probabilistic supergraph search over large uncertain graphs," in *CIKM*, pp. 809–818, ACM, 2014.
[37] Y. Yuan, G. Wang, L. Chen, and H. Wang, "Graph similarity search on large uncertain graph databases," *VLDB J.*, vol. 24, no. 2, pp. 271–296, 2015.
[38] M. Gao, C. Jin, W. Wang, X. Lin, and A. Zhou, "Similarity query processing for probabilistic sets," in *ICDE*, pp. 913–924, IEEE, 2013.
[39] W. Zhang, X. Lin, Y. Zhang, K. Zhu, and G. Zhu, "Efficient probabilistic supergraph search," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 4, pp. 965–978, 2016.
[40] R. Cheng, E. Lo, X. S. Yang, M.-H. Luk, X. Li, and X. Xie, "Explore or exploit?: Effective strategies for disambiguating large databases," *PVLDB*, vol. 3, no. 1-2, pp. 815–825, 2010.
[41] Y.-C. Lin, D.-N. Yang, H.-H. Shuai, and M.-S. Chen, "Data acquisition for probabilistic nearest-neighbor query," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 2, pp. 410–427, 2015.
[42] J. Xu, D. V. Kalashnikov, and S. Mehrotra, "Query aware determinization of uncertain objects," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 1, pp. 207–221, 2015.
[43] H. Altwaijry, D. V. Kalashnikov, and S. Mehrotra, "Query-driven approach to entity resolution," *PVLDB*, vol. 6, no. 14, pp. 1846–1857, 2013.
[44] H. Altwaijry, S. Mehrotra, and D. V. Kalashnikov, "QuERy: A framework for integrating entity resolution with query processing," *PVLDB*, vol. 9, no. 3, pp. 120–131, 2015.

Xiaoye Miao received the PhD degree in computer science from Zhejiang University, China, in 2017. She received the BS degree in computer science from Xi'an Jiaotong University, China, in 2012. She is currently working as a postdoctoral fellow at City University of Hong Kong. Her research interests include uncertain/incomplete databases, graph databases, data cleaning, and data pricing.

Yunjun Gao received the PhD degree in computer science from Zhejiang University, China, in 2008. He is currently a Professor in the College of Computer Science, Zhejiang University, China. His research interests include spatial and spatio-temporal databases, metric and incomplete/uncertain data management, spatio-textual data processing, and database usability. He is a member of the ACM and the IEEE, and a senior member of the CCF.

Linlin Zhou received the BS degree in software engineering from Southeast University, China, in 2014. She received the MS degree in the College of Computer Science, Zhejiang University, China, in 2017. Her research interests include database usability and data cleaning.



Wei Wang received the PhD degree in computer science from the Hong Kong University of Science and Technology, in 2004. He is currently a Professor in the School of Computer Science and Engineering, University of New South Wales, Australia. His research interests include similarity search, knowledge graphs, and databases.

Qing Li is a Professor at the Department of Computer Science, City University of Hong Kong where he joined as a faculty member since Sept 1998. Prior to that, he has taught at the Hong Kong Polytechnic University, the Hong Kong University of Science and Technology and the Australian National University (Canberra, Australia). Prof. Li has served as a consultant to Microsoft Research Asia (Beijing, China), Motorola Global Computing and Telecommunications Division (Tianjin Regional Operations Center), and the Division of Information Technology, Commonwealth Scientific and Industrial Research Organization (CSIRO) in Australia. He is/was a Guest Professor of the University of Science and Technology of China (USTC) and Zhong Shan (Sun Yat-Sen) University, a Visiting Professor of the Institute of Computing Technology (Knowledge Grid), Chinese Academy of Science (Beijing, China), and an Adjunct Professor of the Huazhong University of Science and Technology and the Hunan University (Changsha, China) where he got his BEng.degree from the Department of Computer Science in 1982. He is also a Guest Professor (Software Technology) of the Zhejiang University (Hangzhou, China) – the leading university of the Zhejiang province where he was born. He is currently a Fellow of IET (formerly IEE), a Senior Member of IEEE, a member of ACM-SIGMOD and IEEE Technical Committee on Data Engineering. He is the Chairperson of the Hong Kong Web Society, and also served/is serving as an executive committee (EXCO) member of IEEE-Hong Kong Computer Chapter and ACM Hong Kong Chapter.