Computing the Most-likely Restricted Skyline on Uncertain Datasets

Xiangyu Gao
Harbin Institute of Technology
Harbin, China
gaoxy@hit.edu.cn

Jianzhong Li
Shenzhen Institute of Advanced Technology
Chinese Academy of Sciences
Shenzhen, China
lijzh@hit.edu.cn

Dongjing Miao
Harbin Institute of Technology
Harbin, China
miaodongjing@hit.edu.cn

ABSTRACT

Due to the widespread application of restricted rskyline queries and the rapid increase in the amount of uncertain data, supporting restricted skyline queries on uncertain data has been recently studied. Existing work focused on computing the probability of a tuple appearing in the restricted rskyline result. However, it is observed that tuples with high probability under this definition may be mutually $\mathcal{F}$-dominated, which affects the diversity of selected tuples for further decisions. To address this issue, in this paper, we study the problem of finding a set of incomparable tuples that has the highest probability as the restricted rskyline result. We prove this problem is NP-hard and can not be approximated within $2^{-O(n^{1-\delta})}$ in polynomial time for any $\delta > 0$, unless P = NP. To solve this problem efficiently, we first design a series of data reduction rules to reduce the input data size. Then, we propose two exact algorithms with different searching strategies and pruning strategies. We also propose a local-search based approximation algorithm to further improve the time efficiency at the expense of a little loss in solution quality. Our extensive experiments on real-world and synthetic datasets confirm the effectiveness and efficiency of our proposal.

PVLDB Reference Format:
Xiangyu Gao, Jianzhong Li, and Dongjing Miao. Computing the Most-likely Restricted Skyline
on Uncertain Datasets. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

1 INTRODUCTION

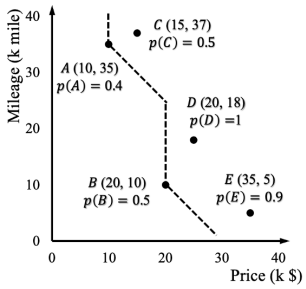
Uncertain data is inherent in many applications such as market analysis, environmental surveillance, and bioinformatics [3]. The reasons for uncertain data in these applications include but are not limited to equipment accuracy, delayed data updates, unreliable data transmission, etc. Due to the importance of these applications and the rapid increase in the amount of uncertain data, a great deal of attention has been devoted to query processing on uncertain datasets [2, 4–7, 10, 16–18, 21, 23, 24].

Selecting a small yet representative subset of tuples for supporting decision-making is a vital research topic in database society. To fulfill this task, restricted skyline (rskyline) queries [11] are proposed for identifying a subset of tuples that preserves the best score with respect to any scoring function of interest. Formally speaking, given a set of monotone scoring functions $\mathcal{F}$, the rskyline query retrieves a set of tuples that are not $\mathcal{F}$-dominated by any other tuples, where a tuple t is said to $\mathcal{F}$-dominate another tuple s if t scores better than s under any function in $\mathcal{F}$.

Supporting rskyline queries on uncertain data is crucial for many real-world applications. For instance, a job seeker has different probabilities of getting jobs with varying salary and job security from different companies. The hiring probability is determined by the seeker's education level, jobs' acceptance rates, etc. Given the requirement that if the job security reduces by one level, then the salary must increase by at least 1,000 dollars, conducting rskyline queries with $\mathcal{F} = \{\omega_{Sa}\text{Salary} + \omega_{Se}\text{Security} \mid \omega_{Sa} \geq 1000 \cdot \omega_{Se}\}$ can help the seeker to identify jobs with high probabilities of being most satisfying. Also, second-hand products on the internet may not always be available duo to untimely data updates. The available probability can be estimated based on the dealer's credit-ranking or the dealer's product volume. Then, performing rskyline queries with a preference for cheaper products will help to select products more likely to have a price advantage.

Recently, Gao et al. first investigated how to conduct rskyline queries on uncertain datasets [16]. They adopted possible worlds semantics [1] and studied the problem of computing the probability that a tuple belongs to the rskyline. However, we observe that tuples with high probability (e.g., top-k) identified by their methods may be mutually $\mathcal{F}$-dominated and some attractive tuples are ignored. This affects the diversity of selected tuples for further decisions. As an illustrative example, Figure 1 (a) shows an uncertain dataset of five used cars with varying prices and mileages. Let $\mathcal{F} = \{\omega_P\text{Price} + \omega_M\text{Milege} \mid \omega_P \geq \omega_M\}$, all possible worlds and their rskylines are listed in Figure 1 (b). As defined in [16], the probability of a car being in the rskyline is the sum of the existence probabilities of possible worlds within which it belongs to the rskyline. According to the results listed in Figure 1 (c), the top-2 cars B and D. Whereas, D is $\mathcal{F}$-dominated by B. In other word, B and D never constitute an rskyline in any possible world. Thus, no matter how a shopper further sets up the relationship between ω_P and ω_M (e.g., $\omega_P = \omega_M$ or $\omega_P = 10 \cdot \omega_M$), B is the only choice. Moreover, since incomparable tuples do not affect each other's probability of being in the rskyline,

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX



(a) Used cars.

Possible world	Prob.	RSKY
{A, B, C, D, E}	0.09	{A, B}
{A, B, C, D}	0.01	{A, B}
{A, B, D, E}	0.09	{A, B}
{A, C, D, E}	0.09	{A, D, E}
{B, C, D, E}	0.135	{B, C}
{A, B, D}	0.01	{A, B}
{A, C, D}	0.01	{A, D}
{A, D, E}	0.09	{A, D, E}
{B, C, D}	0.015	{B, C}
{B, D, E}	0.135	{B}
{C, D, E}	0.135	{C, D, E}
{A, D}	0.01	{A, D}
{B, D}	0.015	{B}
{C, D}	0.015	{C, D}
{D, E}	0.135	{D, E}
{D}	0.015	{D}

(b) Possible worlds, existence probabilities, rskylines.

CarID	Rsky. Prob.
A	0.4
B	0.5
C	0.3
D	0.5
E	0.45

(c) Probability of a car appearing in the rskyline.

RSKY	Rsky. Prob.
{A, B}	0.2
{A, D, E}	0.18
{B, C}	0.15
{B}	0.15
{C, D, E}	0.135
{D, E}	0.135
{C, D}	0.015
{D}	0.015
{A, D}	0.02

(d) Probability of a set of incomparable cars being the rskyline.

Figure 1: An uncertain dataset of used cars and $\mathcal{F} = \{\omega_P \text{Price} + \omega_M \text{Mileage} \mid \omega_P \geq \omega_M\}$.

both A and C with price advantages are excluded from the result. However, the probability that neither is available is merely 0.3. To cover at least one of them, a straightforward method is to specify a larger k. But this will include more tuples that are $\mathcal{F}$-dominated by other tuples, such as E in this example.

Motivated by these issues, in this paper, we study the problem of finding a valid rskyline set, i.e., a set of tuples that are not $\mathcal{F}$-dominated by each other, with the maximum probability of being the rskyline result. We refer to such set as the most-likely rskyline and refer to this problem as the most-likely rskyline problem. Similarly, the probability of a valid rskyline set as the rskyline result is defined as the sum of the existence probabilities of possible worlds within which it is the rskyline result. Figure 1 (d) lists all valid rskyline sets along with their probabilities, and the most-likely rskyline is $\{A, B\}$. Unlike the top-k results, the most-likely rskyline always consists of incomparable tuples. It is easy to verify that A serves as a better choice than B when $\omega_P \geq 2.5\omega_M$. Moreover, under this definition, incomparable tuples form different rskyline results with different probabilities, which provides a global insight for rskyline results across all possible worlds.

To our knowledge, no work has been done to solve the most-likely rskyline problem to date. The challenges in solving this problem come from two aspects. First, the introduction of uncertainty makes there might be an exponential number of valid rskyline sets. Thus, it is time costly to compute rskyline of each possible world with efficient methods proposed for traditional rskyline queries [11]. Second, the involvement of $\mathcal{F}$ makes the most-likely rskyline problem much harder than a special case of it, i.e., the most-likely skyline problem [4, 21], which aims to find a set of non-dominated tuples with the maximum probability as the skyline result. Recall that rskyline was proved to be equivalent to skyline when $\mathcal{F}$ consists of all monotone scoring functions [11]. It was proved that when $d \geq 3$, the most-likely skyline problem is NP-hard [21] and can not be approximated within $2^{-O(n^{1-\delta})}$ in polynomial time for any $\delta > 0$, unless P = NP [4]. In this paper, we prove that these hardness results hold for the most-likely rskyline problem even when $d = 2$.

Given these hardness results, we first design a series of data reduction rules to reduce the input size as much as possible. This will improve the practical performance of all proposed algorithms. We then focus on designing exact algorithms for the most-likely rskyline problem. We propose a baseline algorithm DSA by adapting the dynamic programming based search algorithm proposed for the most-likely skyline problem [21]. Due to the arbitrariness of set $\mathcal{F}$, the adaption is non-trivial. We apply different strategies to deal with different types of $\mathcal{F}$. However, it is observed that DSA is inefficient since its breadth-first search strategy and loose pruning condition force it to store a large set of candidates in memory. Therefore, we propose a new branch-and-bound algorithm called BBA, which performs a depth-first search on the candidate space. We also develop more effective pruning strategies based on properties of reduced dataset to further speed up the search. Experimental results show that BBA outperforms DSA by around two orders of magnitude. Although efficient for small-to-medium sized uncertain datasets, exponential time complexity prevents BBA from handling uncertain datasets of enormous size. Thus, we propose a local search based approximation algorithm LSA at the expense of a little loss in solution quality. The main intuition is to first find an approximate result and then improve its quality through some local search operations. Although without theoretical accuracy guarantee, experimental results show that LSA achieves an approximation ratio less than 1.5 in most cases.

To sum up, our major contributions are summarized as follows.

- We formalize the most-likely rskyline problem. We prove the problem is NP-hard and can not be approximated within $2^{-O(n^{1-\delta})}$ in polynomial time for any $\delta > 0$, unless P = NP.
- We design a series of data reduction rules, which significantly reduce the input data size.
- We propose two exact algorithms DSA and BBA and a local search based approximation algorithm LSA.
- We conduct a comprehensive experimental study on both real and synthetic datasets to demonstrate the efficiency and scalability of the proposed algorithms.

The rest of this paper is organized as follows. We formalize the most-likely rskyline problem and investigate its problem hardness in section 2. We propose a series of data reduction rules in section 3. We propose two exact algorithms and a local search based approximation algorithm in section 4 and section 5, respectively. We report the experimental results in section 6 and review the related work in section 7. Finally, we conclude the paper in section 8.

2 PROBLEM DEFINITION AND HARDNESS

In this section, we first review the restricted skyline query, then formally define the most-likely restricted skyline problem and investigate its problem complexity.

2.1 Restricted Skyline

We consider a d-dimensional dataset D consisting of n tuples. Each tuple $t \in D$ has d numeric attributes, denoted by $t = (t[1], \cdots, t[d])$. Under the assumption that lower values are preferred than higher ones, the dominance and skyline are defined as follows.

Definition 2.1 (Dominance and Skyline). Given a dataset D, a tuple t dominates another tuple s, denoted as $t \prec s$, if $\forall 1 \leq d \leq d$, $t[i] \leq s[i]$ and $\exists j \in [1, d]$, $t[j] < s[j]$. The skyline of D, denoted as $\text{SKY}(D)$, is a subset of tuples that are not dominated by any other tuples in D, i.e.,

$$\text{SKY}(D) = \{t \in D \mid \forall s \in D, s \not\prec t\}.$$

Given a scoring function $f : \mathbb{R}^d \to \mathbb{R}^+$, the value $f(t[1], \cdots, t[d])$ is called the score of tuple t, also written as $f(t)$. And f is said to be is *monotone* if for any tuple t, s it holds that $f(t) \leq f(s)$ if $\forall 1 \leq i \leq d, t[i] \leq s[i]$. By considering a set of scoring functions $\mathcal{F}$, the $\mathcal{F}$-dominance and restricted skyline are defined by extending the dominance and skyline introduced in Definition 2.1.

Definition 2.2 ($\mathcal{F}$-dominance and Restricted Skyline). Given a d-dimensional dataset D and a set of monotone scoring functions $\mathcal{F}$, a tuple t $\mathcal{F}$-dominates another tuple s, denoted as $t \prec_{\mathcal{F}} s$, if $\forall f \in \mathcal{F}, f(t) \leq f(s)$. The restricted skyline (rskyline) of D with respect to $\mathcal{F}$, denoted as $\text{RSKY}(D, \mathcal{F})$, is a subset of tuples that are not $\mathcal{F}$-dominated by any other tuples in D, i.e.,

$$\text{RSKY}(D, \mathcal{F}) = \{t \in D \mid \forall s \in D, s \not\prec_{\mathcal{F}} t\}.$$

By abuse of notation, we also define the $\mathcal{F}$-dominance relationships between a tuple and a set of tuples. We use $t \prec_{\mathcal{F}} S$ to denote that t $\mathcal{F}$-dominates every tuple in S, and $S \prec_{\mathcal{F}} t$ to denote that there exists an tuple in S that $\mathcal{F}$-dominates t.

2.2 Uncertain Restricted Skyline

Similar to previous related work [4, 21], we denote an uncertain dataset by $\mathcal{D} = (T, p)$, where T is a set of tuples and $p : T \to (0, 1]$ is a function that determines the existence probability of a tuple in the dataset. A possible world $D \subseteq T$ of an uncertain dataset $\mathcal{D}$ is a deterministic dataset which is a possible outcome of the random variables representing the tuples in T. Moreover, we assume that the uncertainty variables of different tuples are independent of one another. Thus, the probability of sampling the possible world D from the random variables representing the uncertain dataset $\mathcal{D}$ can be mathematically quantified as

$$\Pr(D) = \prod_{t \in D} p(t) \cdot \prod_{t \notin D} (1 - p(t)).$$

We use $D \sqsubseteq \mathcal{D}$ to denote that a dataset D is a possible world of $\mathcal{D}$. It is easy to verify that the total number of such possible world of $\mathcal{D}$ is $2^{|T|}$ and $\sum_{D \sqsubseteq \mathcal{D}} \Pr(D) = 1$. Under the possible world semantics, the probability that a candidate set S is the rskyline of an uncertain dataset $\mathcal{D}$ with respect to $\mathcal{F}$ is defined as follows.

Definition 2.3 (RSkyline Probability). Given an uncertain dataset $\mathcal{D} = (T, p)$ and a set of monotone scoring functions $\mathcal{F}$, the rskyline probability of a candidate rskyline S is defined as the probability that S is the rskyline of the sampled realization of $\mathcal{D}$ with respect to $\mathcal{F}$, i.e.,

$$\Pr_{\text{rsky}}(S) = \sum_{D \sqsubseteq \mathcal{D}} \Pr(D) \cdot \mathbf{1}(S = \text{RSKY}(D, \mathcal{F})), \tag{1}$$

where $\mathbf{1}(\cdot)$ is the indicator function.

Note that S is the rskyline of a possible world D of $\mathcal{D}$ with respect to $\mathcal{F}$ if and only if all tuples in S appear in D and all tuples not $\mathcal{F}$-dominated by any tuple in S do not exist. Therefore, the rskyline probability of S can be equivalently represented as

$$\Pr_{\text{rsky}}(S) = \prod_{t \in S} p(t) \cdot \prod_{t \in T \setminus S, S \not\prec_{\mathcal{F}} t} (1 - p(t)). \tag{2}$$

Now, we are ready to formally introduce the most-likely rskyline problem studied in this paper.

Definition 2.4 (The Most-likely RSkyline Problem). Given an uncertain dataset $\mathcal{D} = (T, p)$ and a set of monotone scoring functions $\mathcal{F}$, the goal of the most-likely rskyline problem is to find a subset of tuples $S^* \subseteq T$ such that rskyline probability $\Pr_{\text{rsky}}(S^*)$ is maximized, i.e.,

$$\text{MRSKY}(\mathcal{D}, \mathcal{F}) = S^* = \arg\max_{S \subseteq T} \Pr_{\text{rsky}}(S).$$

2.3 Problem Hardness

Next, we discuss the hardness of the most-likely rskyline problem.

Theorem 2.5. *The most-likely rskyline problem is* NP-hard.

Proof. Given an uncertain dataset $\mathcal{D} = (T, p)$, the most-likely skyline problem [4, 21] aims to find a subset S with the largest skyline probability $\Pr_{\text{sky}}(S)$, where

$$\Pr_{\text{sky}}(S) = \sum_{D \sqsubseteq \mathcal{D}} \Pr(D) \cdot \mathbf{1}(S = \text{SKY}(D)) = \prod_{t \in S} p(t) \cdot \prod_{t \in T \setminus S, S \not\prec t} (1 - p(t)).$$

And it is proved that the most-likely skyline problem is NP-hard when $d \geq 3$ [4, 21]. Since the most-likely skyline problem is a special case of the problem studied in this paper (by letting $\mathcal{F}$ be the set of all monotone scoring functions), it is straightforward to derive that the most-likely rskyline problem is also NP-hard when $d \geq 3$. Thus, in what follows, we further prove that the most-likely rskyline problem is still NP-hard when $d = 2$ by establishing a polynomial-time reduction from the most-likely skyline problem with $d = 3$. Given a 3-dimensional uncertain dataset $\mathcal{D}_{\text{sky}} = (T_{\text{sky}} = \{t_1, \cdots, t_n\}, p_{\text{sky}})$, we construct a 2-dimensional uncertain dataset $\mathcal{D}_{\text{rsky}} = (T_{\text{rsky}}, p_{\text{rsky}})$ and a set of monotone scoring functions $\mathcal{F} = \{f_1(\cdot), f_2(\cdot), f_3(\cdot)\}$ as follows. Let $T_{\text{rsky}} = \{x_i = (\frac{i}{n+1}, \frac{n+1-i}{n+1}) \mid i \in \{1, \cdots, n\}\}$ and $p_{\text{rsky}}(x_i) = p_{\text{sky}}(t_i)$. Then, for each $i \in \{1, 2, 3\}$, add a piece-wise function $f_i(x)$ into $\mathcal{F}$, where

$$f_i(x) = \begin{cases} t_j[i] & \text{if } \exists 1 \leq j \leq n, x = x_j, \\ \min_{1 \leq j \leq n} t_j[i] & \text{else if } x[2] \leq 1 - x[1], \\ \max_{1 \leq j \leq n} t_j[i] & \text{otherwise.} \end{cases}$$

Function $f_i(x)$ is monotone since for any x_1 and x_2 if $x_1[1] \leq x_2[1]$ and $x_1[2] \leq x_2[2]$, then $f_i(x_1) \leq f_i(x_2)$. Thus, we obtain a valid input for the most-likely rskyline problem. According to the definition of $\mathcal{F}$-dominance, we know that for any two tuples $x_p, x_q \in T_{\text{rsky}}$, $x_p \prec_{\mathcal{F}} x_q$ if and only if $f_i(x_p) = t_p[i] \leq f_i(x_q) = t_q[i]$ holds for all $1 \leq i \leq 3$, i.e., $t_p \prec t_q$. Therefore, for each

possible world $D_{\text{sky}} = \{t_i \mid i \in I, I \subseteq \{1, \cdots, n\}\} \sqsubseteq \mathcal{D}_{\text{sky}}$, set $S_{\text{sky}} = \{t_j \mid j \in J, J \subseteq I\}$ is the skyline of D_{sky} if and only if $S_{\text{rsky}} = \{x_j \mid j \in J\}$ is the rskyline of $D_{\text{rsky}} = \{x_i \mid i \in I\} \sqsubseteq \mathcal{D}_{\text{rsky}}$ with respect to $\mathcal{F}$. That is $\Pr_{\text{sky}}(S_{\text{sky}}) = \Pr_{\text{rsky}}(S_{\text{rsky}})$. Thus, finding a subset S_{sky} with largest $\Pr_{\text{sky}}(S_{\text{sky}})$ on $\mathcal{D}_{\text{sky}}$ is equivalent to finding a subset S_{rsky} with largest $\Pr_{\text{rsky}}(S_{\text{rsky}})$ on $\mathcal{D}_{\text{rsky}}$ under the scoring functions $\mathcal{F}$. This completes the proof. □

A common approach for dealing with NP-hard problems is to efficiently find approximation results. Note that the goal of the most-likely rskyline problem is to maximize the rskyline probability. The objective function can be equivalently rewritten as follows,

$$\Pr_{\text{rsky}}(S) \propto f(S) = \log \Pr_{\text{rsky}}(S) = \sum_{t \in S} \log p(t) + \sum_{t \in T \setminus S, S \not\prec_{\mathcal{F}} t} \log(1 - p(t))$$

We show that function $f(\cdot)$ falls into the worst class of non-submodular functions, i.e., it is even not γ-weakly submodular with $\gamma > 0$ [9].

Lemma 2.6. *$f(\cdot)$ is not submodular.*

Proof. A submodular function $g(\cdot)$ should satisfy that for every $X, Y \subseteq T$ with $X \subseteq Y$ and every $t \in T \setminus Y$, $g(X \cup \{t\}) - g(X) \geq g(Y \cup \{t\}) - g(Y)$. We use a counterexample to show that $f(\cdot)$ disobeys this rule. The input uncertain dataset is $\mathcal{D} = (T, p)$, where $T = \{a = (0, 1), b = (1, 0), c = (1.5, 0.5)\}$ and $p(a) = p(b) = 0.5$, $p(c) = 0.1$, and $\mathcal{F}$ including all monotone scoring functions. Let $X = \{a\}$, $Y = \{a, c\}$, and $t = b$, we have

$$\begin{cases} f(X) = \log p(a) + \log(1 - p(b)) + \log(1 - p(c)) = \log \frac{9}{40}, \\ f(Y) = \log p(a) + \log p(c) + \log(1 - p(b)) = \log \frac{1}{40}, \\ f(X \cup \{t\}) = f(Y \cup \{t\}) = \log p(a) + \log p(b) = \log \frac{1}{4}. \end{cases}$$

Then, it is derived

$$f(X \cup \{t\}) - f(X) - f(Y \cup \{t\}) + f(Y) = \log \frac{1}{9} < 0.$$

This completes the proof. □

Lemma 2.7. *$f(\cdot)$ is not γ-weakly submodular with a positive γ.*

Proof. A function $g(\cdot)$ is γ-weakly submodular if $\exists \gamma > 0$, s.t. $\sum_{t \in Y \setminus X}[g(X \cup \{t\}) - g(X)] \geq \gamma \cdot [g(X \cup Y) - g(X)]$, $\forall X, Y \subseteq T$. We use a counterexample to show that $f(\cdot)$ disobeys this rule. The input uncertain dataset is $\mathcal{D} = (T, p)$, where $T = \{a = (0, 1), b = (1, 0), c = (1.5, 0.5)\}$ and $p(a) = 1/2$, $p(b) = 4/7$, $p(c) = 1/3$, and $\mathcal{F}$ including all monotone scoring functions. Let $X = \{a\}$ and $Y = \{a, b, c\}$, we have

$$\begin{cases} f(X) = \log p(a) + \log(1 - p(b)) + \log(1 - p(c)) = \log \frac{1}{7} \\ f(X \cup \{b\}) = f(X \cup Y) = \log p(a) + \log p(b) = \log \frac{2}{7} \\ f(X \cup \{c\}) = \log p(a) + \log(1 - p(b)) + \log p(c) = \log \frac{1}{14} \end{cases}$$

Substitute these values we get $0 \geq \gamma \cdot \log 2$, which infers $\gamma = 0$. □

Furthermore, according to the reduction established in the proof of Theorem 2.5, we know that for any valid skyline set S, $\Pr_{\text{sky}}(S)$ equals to $\Pr_{\text{rsky}}(S_{\text{rsky}})$ with respect to $\mathcal{F}$. Thus, with the non-approximability result established in [4], we claim that the most-likely rskyline problem is hard to approximate.

Theorem 2.8. *For any $\delta > 0$, the most-likely rskyline problem can not be approximated within $2^{-O(n^{1-\delta})}$ in polynomial time, unless* P = NP.

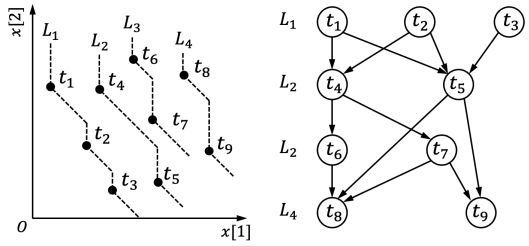


Figure 2: Maximal rskyline layers and $\mathcal{F}$-dominant graph.

3 DATA REDUCTION

To improve the practical performance of our proposed algorithms for the most-likely rskyline problem, in this section, we first try to reduce the data size as much as possible while guaranteeing that the result derived from the reduced dataset is the same as the one derived from the entire dataset. Since we focus on data complexity in this paper, it is assumed that there is an oracle for determining the $\mathcal{F}$-dominance relationship between any two tuples. In our experimental study, we will give concrete implementations of $\mathcal{F}$-dominance tests involved.

Given an uncertain dataset $\mathcal{D} = (T, p)$ and a set of monotone scoring functions $\mathcal{F}$, a reduction rule allows us to identify tuples that must be in some MRSKY$(\mathcal{D}, \mathcal{F})$ or discard tuples that must not belong to any MRSKY$(\mathcal{D}, \mathcal{F})$. In [21], the authors designed two reduction rules for the most-likely skyline problem. We notice that they can also be applied to the most-likely rskyline problem.

Reduction Rule 1 (Point Reduction [21]). *Given an uncertain dataset $\mathcal{D} = (T, p)$ and a set of monotone scoring functions $\mathcal{F}$, if there exists a tuple t such that $p(t) > 1/2$, then all tuples $\mathcal{F}$-dominated by t can be safely discarded from $\mathcal{D}$.*

Reduction Rule 2 (Region Reduction [21]). *Given an uncertain dataset $\mathcal{D} = (T, p)$ and a set of monotone scoring functions $\mathcal{F}$, if there exist two tuples t and s such that $t \prec_{\mathcal{F}} s$ and*

$$\frac{p(t)}{(1 - p(t))(1 - p(s))} \cdot \prod_{x \in T, t \prec_{\mathcal{F}} x \prec_{\mathcal{F}} s} (1 - p(x))^{-1} > 1,$$

then all tuples $\mathcal{F}$-dominated by s can be safely discarded from $\mathcal{D}$.

For completeness, we include the proofs and algorithms for these two reduction rules in the appendix of the full version of this paper [14]. An intuitive explanation is that substituting the pruned tuple in a candidate rskyline with t stated in the rules will always result in a candidate with strictly higher rskyline probability. While effective, performing region reduction takes $O(n^3)$ time since for each unpruned tuple s, it needs to check all tuples $\mathcal{F}$-dominating s to see if tuples dominated by s can be discarded [21]. To improve the efficiency, we introduce path reduction based on the notion of $\mathcal{F}$-dominant graph as a fast approximation.

A $\mathcal{F}$-dominant graph G of T is a directed acyclic graph where nodes represent tuples and edges represent $\mathcal{F}$-dominance relationships. In what follows, the words "tuple" and "node" will be used interchangeably. For each node t in G, the ancestors $A(t)$ of t includes all tuples that $\mathcal{F}$-dominate t and the descendants $D(t)$ of t includes all tuples that are $\mathcal{F}$-dominated by t. Moreover, nodes in G are partitioned into *maximal layers*, where the first layer L_1 is

the set of nodes with zero in-degree, i.e., rskyline tuples of T, and for $i > 1$, the i-th layer L_i is the set of nodes with zero in-degree when nodes at $\bigcup_{j=1}^{i-1} L_j$ are removed from G, i.e., rskyline tuples of $T - \bigcup_{j=1}^{i-1} L_j$. We refer to L_i as the i-th layer of G. It is easy to verify that edges can only exist from nodes at L_j to nodes at L_i for $0 < j < i$. An example graph of nine tuples with four layers is shown in Figure 2. For easy of visualization, we omit all indirect $\mathcal{F}$-dominance edges such as $t_2 \prec_{\mathcal{F}} t_6$.

To construct G, we first sort tuples in T in ascending order according to their scores under a function $f \in \mathcal{F}$. Then, we scan the sorted list once to assign each tuple to the layer that contains it. Suppose there exist k layers when we process a tuple t. We compare t with tuples currently at layer $L_{\lceil k/2 \rceil}$. If t is $\mathcal{F}$-dominated by some tuple at that layer, then t must be at some layer higher than $\lceil k/2 \rceil$. Otherwise, t belongs to that layer or some layer lower than $\lceil k/2 \rceil$. We conduct this binary search recursively until t is assigned to a layer L_i. If $i > 1$, we compare t against each tuple $s \in \bigcup_{j=1}^{i-1} L_j$ and insert an edge from s to t if $s \prec_{\mathcal{F}} t$.

Given two distinct nodes s and t in G, a *path* $p(s,t)$ from s to t is an ordered sequence of edges. The nodes s and t are called the *source* and *target* nodes respectively, while the remaining ones constitute the set $Int(p(s,t))$ of *internal nodes*. For example, as shown in Figure 2, a path from t_1 to t_8 is $p(t_1, t_8) = ((t_1, t_4), (t_4, t_6), (t_6, t_8))$ and the internal nodes of this path are $Int(p(t_1, t_8)) = \{t_4, t_6\}$.

Reduction Rule 3 (Path Reduction). *Given an uncertain dataset $\mathcal{D} = (T, p)$ and a set of monotone scoring functions $\mathcal{F}$, if there exists two tuples t and s such that $t \prec_{\mathcal{F}} s$ and*

$$\frac{p(t)}{(1-p(t))(1-p(s))} \cdot \prod_{x \in Int(p(t,s))} (1-p(x))^{-1} > 1$$

where $p(t,s)$ is a path from t to s in the $\mathcal{F}$-dominant graph G, then all tuples $\mathcal{F}$-dominated by s can be safely discarded from $\mathcal{D}$.

Instead of considering all tuples in $D(t) \cap A(s)$ like region reduction, path reduction only considers tuples on a path from t to s. Therefore, it is a sufficient condition for region reduction.

To perform path reduction, for each tuple s, we use $\beta(s)$ to record the maximum value among all paths ending in s, i.e.,

$$\beta(s) = \max_{p(t,s), t \in A(s)} \frac{p(t)}{(1-p(t))(1-p(s))} \cdot \prod_{x \in Int(p(t,s))} (1-p(x))^{-1}.$$

It is observed that

$$\beta(s) = \max_{(x,s) \in G} \max_{p(t,x), t \in A(x)} \frac{1}{1-p(s)} \cdot \frac{p(t)}{(1-p(t))(1-p(x))} \cdot \prod_{y \in Int(p(t,x))} (1-p(y))^{-1} = \max_{(x,s) \in G} \frac{\beta(x)}{1-p(s)}.$$

According to above recursive formula, we can perform path reduction while constructing the $\mathcal{F}$-dominant graph G in $O(n^2)$ time.

Moreover, we observe that point reduction takes the pruning power of tuple t with $p(t) > 1/2$, and path and region reductions take the pruning power of tuple $t \in L_i$ with $p(t) \leq 1/2$ for $i > 1$. As a complement, we keep working with tuple $t \in L_1$ with $p(t) \leq 1/2$ and propose virtual region reduction

Algorithm 1: Data Reduction

Input: an uncertain dataset $\mathcal{D} = (T, p)$, a set of monotone scoring functions $\mathcal{F}$
Output: the reduced dataset $\mathcal{D}$, $\mathcal{F}$-dominance Graph G

```
P ← ∅;
Sort T according to some f ∈ F;
foreach t ∈ T do
    Use binary search to find j* = arg min_j L_j ⊀_F t;
    L_{j*} ← L_{j*} ∪ {t};
    if j* = 1 then
        if p(t) > 1/2 then P ← P ∪ {t} ;
        else
            Find t' according to virtual region reduction;
            if t' exists then P ← P ∪ {t'} ;
    else if P ≺_F t then
        Remove t from G;
    else
        foreach s ∈ ⋃_{j=1}^{j*-1} L_j do
            if s ≺_F t then
                Add an edge (s, t) into G;
                β(t) ← max(β(t), β(s)/1 − p(t));
        if p(t) > 1/2 or β(t) > 1 then P ← P ∪ {t} ;
        else
            if ∃s(s ≺_F t ∧ p(s)/((1−p(s))(1−p(t))) ·
              ∏_{∀x,s≺_F x≺_F t} (1 − p(x))^{-1} > 1 then
                P ← P ∪ {t};
foreach t ∈ L_1 do
    if p(t) > 1/2 then Add t into S* ;
    else if p(t) ≤ 1/2 and t's out-degree is zero then
        Remove t from G;
Group G into connected components;
return D, G;
```

Reduction Rule 4 (Virtual Region Reduction). *Given an uncertain dataset $\mathcal{D} = (T, p)$ and a set of monotone scoring functions $\mathcal{F}$, if there exists a tuple t and a set S such that $t \prec_{\mathcal{F}} S$ and*

$$\frac{p(t)}{1-p(t)} \cdot \prod_{s \in S} (1-p(s))^{-1} > 1,$$

let $t' = (\max_{s \in S} s[1], \cdots, \max_{s \in S} s[d])$, then all tuples $\mathcal{F}$-dominated by t' can be safely discarded from $\mathcal{D}$.

Contrary to region reduction, which tests every tuple that $\mathcal{F}$-dominates the tuple t being processed (seeks forward), virtual region reduction keeps adding tuples $\mathcal{F}$-dominated by t into S (seeks backward) until the accumulated value is greater than one. Then it discards all tuples $\mathcal{F}$-dominated by the maximum corner of the minimum bounding box of S. The total time consumption of performing this rule is at most $O(|L_1|n)$. Finally, we involve a simple supplementary rule for point reduction.

Reduction Rule 5 (Point Inclusion and Exclusion). *Given an uncertain dataset $\mathcal{D} = (T, p)$ and a set of monotone scoring functions $\mathcal{F}$, each tuple $t \in L_1$ with $p(t) > 1/2$ must belong to some* MRSKY$(\mathcal{D}, \mathcal{F})$ *and each tuple $t \in L_1$ with zero out-degree and $p(t) < 1/2$ must not belong to any* MRSKY$(\mathcal{D}, \mathcal{F})$.

All formal proofs of the correctness of these reduction rules can be found in the appendix of the full version of our paper [14]. The overall pseudo-code for performing reduction rules is presented in algorithm 1. Due to the inefficiency of region reduction, algorithm 1 has a worst-case time complexity of $O(n^3)$. And if region reduction (line 19-21) is not involved, the time complexity of algorithm 1 reduces to $O(n^2)$. In our experimental studies, we compare these two algorithms and show that path reduction serves as a good approximation of region reduction which prunes a considerable amount of tuples (1.2% fewer tuples on average) in less time (an order of magnitude on average).

Remark that since subsequent computations on different connected components of the $\mathcal{F}$-dominate graph do not affect each other, we group nodes at L_1 at the end of algorithm 1 to find the connected components of G. Then, algorithms proposed below will be executed on these components separately and the final result is obtained by combing these partial results.

An $O(n^2)$-time algorithm for region reduction in special case. Here, we also present an $O(n^2)$-time space partitioning method for region reduction under the dominance relationship in 2-dimensional data space. Formally, let $\mathcal{D} = (T, p)$ be a 2-dimensional uncertain dataset, given any two tuples $s, t \in T$ where $s \prec t$, we aim to compute value $\prod_{x \in T, s \prec x \prec t}(1 - p(x))$ efficiently. We denote this value as $\rho(s, t)$. Let X_1 and X_2 denote the set of $x[1]$ and $x[2]$ coordinates of tuples in T, respectively. The elements of $X_1 \times X_2$ form a grid of at most n^2 tuples. Except for n tuples in T, we call the remaining tuples as virtual tuples and define their existence probabilities as zero. Figure 3 illustrates the grid in an example where T consists of 7 tuples, which are denoted by black points. All the other white points on the grid are virtual tuples, e.g., v_1, v_2 and $p(v_1) = p(v_2) = 0$.

For each point p on the grid, we maintain three values $h(p), v(p), \rho(p)$. Specifically, let l_h and l_v denote the horizontal and vertical lines that contain p, respectively. $h(p) = \prod_{x <_{l_h} p}(1 - p(x))$, where $<_{l_h}$ denote the relationship point x is to the left of t and is on the same horizontal line h as t. Similarly, $v(t) = \prod_{s <_{l_v} t}(1 - p(t))$, where $<_{l_v}$ denote the relationship s is to the bottom of t and is on the same vertical line v as t. And $\rho(t) = \rho(O, t) = \prod_{x \in T, x \prec t}(1 - p(x))$, where O is the origin point. Given two tuples $s = (s[1], s[2])$, $t = (t[1], t[2])$ and $s \prec t$, it is observed that

$$\rho(s, t) = \frac{\rho(t) \cdot \rho(s)}{\rho(t') \cdot \rho(s')} \cdot \frac{h(t')}{h(s)} \cdot \frac{v(s')}{v(s)},$$

where $t' = (t[1], s[2])$ and $s' = (s[1], t[2])$. Continue with the example in Figure 3, $h(v_2) = 1 - p(t_7)$, $v(v_1) = 1 - p(t_7)$, $\rho(t_1) = \prod_{2 \le i \le 7}(1 - p(t_i))$, and $\rho(t_7) = 1$. To compute $\rho(t_7, t_1)$, we have

$$\frac{\rho(t_1) \cdot \rho(t_7)}{\rho(v_1) \cdot \rho(v_2)} \cdot \frac{h(v_2)}{h(t_7)} \cdot \frac{v(v_1)}{v(t_7)} = \frac{\prod_{2 \le i \le 7}(1 - p(t_i))(1 - p(t_7))^2}{(1 - p(t_3))(1 - p(t_5))(1 - p(t_7))^2}.$$

Next, we show how to compute $h(p), v(p), \rho(p)$ for each point p on the grid within $O(n^2)$ time. For each horizontal line l_h, we compute $h(p)$ for each point p of that line from left to right. Concretely, for each l_h, if p is the first point on l_h, then $h(p) = 1$. Otherwise, let the left neighbor of p on l_h be p', then $h(p) = (1 - p(p')) \cdot h(p)$. Similarly, we can compute $v(p)$ for each point of each vertical line from bottom to up. As for value ρ, we process each vertical line l_v from left to right. For each l_v, we compute $\rho(p)$ for each point of l_v from bottom to up. If p is on the leftmost line, then $\rho(p) = v(p)$. Or if p is the very bottom point on l_v, then $\rho(p) = h(p)$. Otherwise, $\rho(p) = \rho(p') \cdot (1 - p(p')) \cdot h(p'') \cdot (1 - p(p''))$.

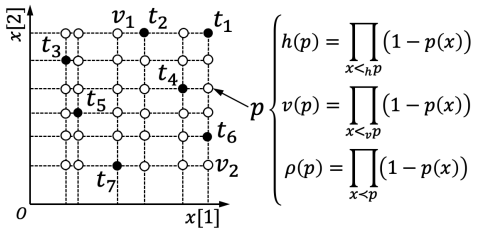


Figure 3: An improved algorithm for 2D dominance relation.

4 EXACT ALGORITHMS

In this section, we present two exact algorithms for solving the most-likely rskyline problem.

4.1 Dynamic Search Algorithm

To solve the most-likely rskyline problem, our first attempt is to adapt the dynamic programming based search algorithm, called DSA, proposed for the most-likely skyline problem [21]. Recall that given an uncertain dataset $D = (T, p)$, the most-likely skyline problem aims to find a subset S with the largest skyline probability which is defined as

$$\Pr_{\text{sky}}(S) = \prod_{t \in S} p(t) \cdot \prod_{t \in T \setminus S, S \not\prec t} (1 - p(t)).$$

DSA first sorts T in ascending order according to their scores under the monotone function $\min_{1 \le i \le d} t[i]$. Then, DSA scans the sorted list once to construct candidate skylines based on the following dynamic programming recursive formula. Specifically, let t_i denote the i-th tuple in the sorted list and $T_i = \{t_1, \cdots, t_i\}$, the *partial skyline probability* of a candidate skyline $S \subseteq T_i$ is defined as

$$\Pr^i_{\text{sky}}(S) = \prod_{t \in S} p(t) \cdot \prod_{t \in T_i \setminus S, S \not\prec t} (1 - p(t)).$$

Now, consider each candidate skyline $S \subseteq T_i$ and tuple t_{i+1}, according to the definition, we have $\Pr^{i+1}_{\text{sky}}(S) = \Pr^i_{\text{sky}}(S) \cdot (1 - p(t_{i+1}))$ if $S \not\prec t_{i+1}$, otherwise $\Pr^{i+1}_{\text{sky}}(S) = \Pr^i_{\text{sky}}(S)$. Moreover, in case of t_{i+1} is not dominated by S, $S \cup \{t_{i+1}\}$ is a new candidate skyline which is a subset of T_{i+1} and $\Pr^{i+1}_{\text{sky}}(S \cup \{t_{i+1}\}) = \Pr^i_{\text{sky}}(S) \cdot p(t_{i+1})$. Finally, when i reaches n, since all tuples that dominate a tuple t appear before t in the sorted list, it is easy to verify that $\Pr_{\text{sky}}(S) = \Pr^n_{\text{sky}}(S)$ holds for every candidate skyline $S \subseteq T$.

According to this, DSA maintains a set of candidate skylines $\mathcal{S}$ and their partial skyline probabilities in memory while accessing tuples in order. At the beginning, $\mathcal{S}$ is initialized as $\{\emptyset\}$ and $\Pr^0_{\text{sky}}(\emptyset) = 1$. Then, supposing tuple t_{i+1} is accessed, DSA tests for

each candidate skyline $S \in \mathcal{S}$ if merging t_{i+1} into S forms a new candidate skyline. There are two cases to be considered. If $S \prec t_{i+1}$, then $S \cup \{t_{i+1}\}$ is not a valid candidate skyline, thus no new candidate skyline is created. Otherwise, DSA inserts a new candidate skyline $S \cup \{t_{i+1}\}$ into $\mathcal{S}$ and computes $\Pr^{i+1}_{\text{rsky}}(S \cup \{t_{i+1}\}) = \Pr^{i}_{\text{sky}}(S) \cdot p(t_{i+1})$ and $\Pr^{i+1}_{\text{sky}}(S) = \Pr^{i}_{\text{sky}}(S) \cdot (1 - p(t_{i+1}))$.

In addition, there are two pruning techniques can be applied during the construction procedure. First, a candidate skyline S can be removed from $\mathcal{S}$ if it satisfies the early termination condition: $\min_{s \in S} \max_{1 \le i \le d} s[i] < \min_{1 \le i \le d} t[i]$, where t is the tuple currently being processed. The condition implies that all remaining tuples are dominated by S, hence we can conclude that the partial skyline probability of S reaches its final value in advance. Second, let S^* denote the best solution found so far. Since the partial rskyline probability is always an upper bound of the rskyline probability, a candidate skyline S can be discarded from $\mathcal{S}$ once its partial rskyline probability becomes less than $\Pr_{\text{rsky}}(S^*)$.

The major obstacle in applying DSA to the most-likely rskyline problem is the inability to the monotone function $\min_{1 \le i \le d} t[i]$. This is because it no longer guarantees that all tuples $\mathcal{F}$-dominating a tuple t appear before t in the sorted list. For example, let $t = (3, 3)$, $s = (5, 2)$, and $\mathcal{F} = \{\omega_1 t[1] + \omega_2 t[2] \mid \omega_1 \ge \omega_2\}$, we have $t \prec_{\mathcal{F}} s$ but $\min(s[1] = 5, s[2] = 2) = 2 < 3 = \min(t[1] = 3, t[2] = 3)$. With the same reason, the early termination condition is also invalid.

We observe that under some types of $\mathcal{F}$, rskyline queries can be transformed to skyline queries [16, 19]. This prompts us to overcome this obstacle for different types of $\mathcal{F}$.

Case 1: $\mathcal{F}$ is a convex set. Here, we say a set $\mathcal{F}$ of monotone scoring functions is convex if there exist m functions $f_1, f_2, \cdots, f_m$ in $\mathcal{F}$ such that for any tuple $t \in \mathbb{R}^d$ and any function $f \in \mathcal{F}$, $f(t) = \sum_{i=1}^{m} \alpha_i f_i(t)$, where $\sum_{i=1}^{m} \alpha_i = 1$. For example, the set $\mathcal{F}$ including linear scoring functions whose weights are described by linear constraints is convex. In this case, we sort tuples in ascending order according to their scores under $\min_{f \in \mathcal{F}} f(t)$ and use $\sum_{1 \le i \le m} f_i(t)$ to break ties. Correspondingly, the early termination condition is settled as $L(S) = \min_{s \in S} \max_{f \in \mathcal{F}} f(s) \le \min_{f \in \mathcal{F}} f(t)$. It is easy to verify that if a tuple s appears ahead of another tuple t in the sorted list, then $t \not\prec_{\mathcal{F}} s$ since there is at least one function f_i such that $f_i(t) > f_i(s)$. Moreover, let s be the tuple in S having minimum $\max_{f \in \mathcal{F}} f(s)$. For any tuple t with $\min_{f \in \mathcal{F}} f(t) \ge \max_{f \in \mathcal{F}} f(s)$, we have $\forall f' \in \mathcal{F}$, $f'(t) \ge \min_{f \in \mathcal{F}} f(t) \ge \max_{f \in \mathcal{F}} f(s) \ge f'(s)$, which means $s \prec_{\mathcal{F}} t$.

Case 2: $\mathcal{F}$ is a non-convex set. In this case, we resort to the $\mathcal{F}$-dominant graph G. To ensure that for each tuple t, all tuples $\mathcal{F}$-dominating t have been processed when t is being processed, we access tuples in a layer ordering of G, i.e., tuples at L_1 are first processed (tuples at the same layer are accessed arbitrarily), then tuples at L_2, etc. As for the early termination test, its essence is to check whether all remaining tuples are $\mathcal{F}$-dominated by a candidate rskyline S. We determine this by checking whether there is a layer in G completely dominated by S. Let L_{i^*} be the highest layer of tuples in S, i.e. $i^* = \arg\max_i L_i \cap S \neq \emptyset$. If S $\mathcal{F}$-dominates all tuples at L_{i^*}, then S $\mathcal{F}$-dominates all tuples at each $L_{j \ge i^*}$. Thus, for each candidate rskyline S, we maintain a set $L(S) = \{t \in L_{i^*} \mid S \prec_{\mathcal{F}} t\}$ where and discard S if $|L(S)| = |L_{i^*}|$. We will update $L(S)$ as follows. When a new candidate rskyline $S' = S \cup \{t\}$ is created, we set $L(S')$ as $L(S) \cup \{t\}$ if $i = i^*$. Otherwise ($i > i^*$), we perform a graph traversal from $L(S) \cup \{t\}$ to L_i to collect $L(S')$.

Algorithm 2: Dynamic Search Algorithm

Input: an uncertain dataset $\mathcal{D} = (T, p)$, a set of monotone scoring functions $\mathcal{F}$
Output: MRSKY($\mathcal{D}, \mathcal{F}$)

```
1  Perform data reduction on D;
2  S ← {∅};
3  S* ← ∅; Pr_rsky(S*) ← 0;
4  /* Case 2: Sort T accroding to layer of t                    */
5  Sort T according to min_{f∈F} f(t);
6  foreach t ∈ T do
7      foreach S ∈ S do
8          /* Case 2: if |L(S)| = |L_{i*}| then                 */
9          if L(S) ≤ min_{f∈F} f(t) then
10             S ← S \ {S};
11             if Pr_rsky(S) > Pr_rsky(S*) then
12                 S* ← S; Pr_rsky(S*) ← Pr_rsky(S);
13         else if S ⊀_F t then
14             S' ← S ∪ {t};
15             Pr_rsky(S') ← Pr_rsky(S) · p(t);
16             Collect L(S');
17             Pr_rsky(S) ← Pr_rsky · (1 − p(t));
18             if Pr_rsky(S') > Pr_rsky(S*) then S ← S ∪ {S'} ;
19             if Pr_rsky(S) < Pr_rsky(S*) then S ← S \ {S} ;
20 return S*;
```

The pseudo-code of the modified DSA is presented in algorithm 2. As stated in [21], the time complexity of algorithm 2 is $O(|\mathcal{S}|n^2)$ where $|\mathcal{S}|$ is the number of possible candidate rskylines and n now represents the data size after data reduction. Since a candidate rskyline is an incomparable tuples subset of the whole dataset, $|\mathcal{S}|$ can be approximated as $O(\gamma^n)$ with $1 < \gamma < 2$.

4.2 Branch-and-Bound Algorithm

The intuition behind the DSA is to perform a breadth-first search on the search space, i.e., the set of all candidate rskylines. There are several drawbacks of DSA: (1) the breadth-first search strategy makes it maintain a large number of candidate rskylines in $\mathcal{S}$, (2) the partial rskyline probability is such a loose upper bound that few candidate rskylines are pruned, (3) no properties of reduced data are utilized to speed up the search. To conquer these, in this section, we introduce a branch-and-bound algorithm, called BBA.

Generally speaking, given an uncertain dataset $\mathcal{D} = (T, p)$ and a set of monotone scoring functions $\mathcal{F}$, BBA performs a depth-first search by recursively partitioning the search space into multiple subspaces, also called branches, via branching. At each search space, BBA maintains three disjoint sets S, C, and E, where S is a set of tuples that must be included in each candidate rskyline within the space, C is a set of tuples that can be included in S in order to form

a candidate rskyline within the space, and E is a set of tuples that must not be included in any candidate rskyline within the space.

A trivial idea is to start BBA from the full search space ($S = \emptyset, C = T, E = \emptyset$). But this will result in a large fan-out of the search tree, which reduces the efficiency. Here, we use the $\mathcal{F}$-dominant graph G to narrow down the candidate set C. Specifically, at each search space, we use C to keep all nodes with in-degree zero in $G \setminus E$, where $G \setminus E$ is the graph derived by removing all tuples in E from G. The search process of BBA now starts from the search space ($S = \emptyset, C = L_1, E = \emptyset$), where L_1 is the set of rskyline tuples of T. The invariance kept by C guarantees that for each tuple t, t is inserted into C if and only if all tuples $\mathcal{F}$-dominating t are included in E. Therefore, we can redefine the partial rskyline probability of a candidate rskyline S at the search space (S, C, E) as

$$\Pr_{\text{rsky}}(S, C, E) = \prod_{t \in S} p(t) \cdot \prod_{t \in E} (1 - p(t)).$$

And it is easy to verify that for all candidate rskylines $S' \supseteq S$ within the search space $\Pr_{\text{rsky}}(S') \leq \Pr_{\text{rsky}}(S, C, E)$ and $\Pr_{\text{rsky}}(S, \emptyset, E) = \Pr_{\text{rsky}}(S)$ if C is empty.

Suppose BBA reaches the search space (S, C, E). Before branching, BBA first adds all tuples $t \in C$ with $p(t) > 1/2$ into S. This is because the out-degree of all such nodes is zero in G according to point reduction. After that, if C is empty, BBA compares S with the best solution S^* found so far, and then backtracks. Otherwise, let $\{t_1, t_2, \cdots, t_m\}$ denote the sequence of tuples in C and for $1 \leq i \leq m$, $p(t_i) \leq 1/2$. BBA sorts them in a non-decreasing order of existence probability and partitions the current space into $m + 1$ subspaces. For $1 \leq i \leq m + 1$, the i-th subspace includes all candidate rskylines that must include $S_i = S \cup \{t_i\}$ and exclude $E_i = E \cup \{t_1, \cdots, t_{i-1}\}$. Here, t_0 and t_{m+1} both correspond to null. And the partial rskyline probability of S_i is computed as

$$\Pr_{\text{rsky}}(S_i, C_i, E_i) = \Pr_{\text{rsky}}(S, C, E) \cdot p(t_i) \cdot \prod_{1 \leq j \leq i-1} (1 - p(t_j)),$$

where $p(t_0) = 0$ and $p(t_{m+1}) = 1$. As for C_i, BBA construct it dynamically. It initializes C_1 as $C \setminus \{t_1\}$. Then for each $2 \leq i \leq m+1$, it collects C_i as $C_{i-1} \setminus \{t_i\}$ along with any out-neighbor of t_{i-1} whose in-degree reduces to zero after removing t_{i-1} from G.

Several pruning techniques can be applied here. First, since the partial rskyline probability reaches its final value when C is empty, a more tight upper bound of $\Pr_{\text{rsky}}(S')$ for all candidate rskylines within the search space (S, C, E) can be derived as

$$\Pr_{\text{rsky}}(S) \leq \Pr_{\text{rsky}}(S, C, E) \cdot \prod_{t \in C} \max(p(t), 1 - p(t)) = \Pr^{+}_{\text{rsky}}(S, C, E),$$

where the second part of the right-hand side accounts for adding all tuples in C to either S or E. Thus, a branch is pruned if its upper bound is less than the rskyline probability of the best solution found so far. Second, we observe that for $1 \leq i \leq m - 1$,

$$\Pr^{+}_{\text{rsky}}(S_i, C_i, E_i) = \Pr_{\text{rsky}}(S, C, E) \cdot p(t_i) \cdot \prod_{1 \leq j \neq i \leq m} (1 - p(t_j)) \cdot \prod_{t \in C_i \setminus C} \max(p(t), 1 - p(t)).$$

Since $p(t_i) \geq p(t_{i+1})$ and $C_i \setminus C \subseteq C_{i+1} \setminus C$, we know

$$\Pr^{+}_{\text{rsky}}(S_i, C_i, E_i) \geq \Pr^{+}_{\text{rsky}}(S_{i+1}, C_{i+1}, E_{i+1}).$$

Algorithm 3: Branch-and-Bound Algorithm

Input: an uncertain dataset $\mathcal{D}$, a set of linear scoring functions $\mathcal{F} = \{S_\omega(\cdot) \mid \omega \in \mathbb{S}^{d-1} \wedge A \times \omega \leq b\}$
Output: MRSKY($\mathcal{D}, \mathcal{F}$)

```
Perform data reduction on D;
S* ← ∅; Pr_rsky(S*) ← 0;
S ← ∅; C ← L_1; E ← ∅; Pr_rsky(S, C, E) ← 1;
BB-MRSKY-rec(S, C, E, Pr_rsky(S, C, E));
return S*;
Procedure BB-MRSKY-rec(S, C, E, Pr_rsky(S, C, E))
    if C = ∅ then
        if Pr_rsky(S, C, E) > Pr_rsky(S*) then
            S* ← S; Pr_rsky(S*) ← Pr_rsky(S, C, E);
    Sort C ← {t_1, ··· , t_m} in descending order of p(t_i);
    foreach i ← 1 to m + 1 do
        S_i ← S ∪ {t_i};
        E_i ← E ∪ {t_1, ··· t_{i−1}};
        Pr_rsky(S_i, C_i, E_i) ←
          Pr_rsky(S, C, E) · p(t_i) · ∏_{j=1}^{i−1} (1 − p(t_j));
        if Pr+_rsky(S_i) < Pr_rsky(S*) then
            i ← m + 1; continue;
        C_i ← C \ {t_1, ··· , t_{i−1}};
        Add nodes in G \ E_i with zero in-degree to C_i;
        BB-MRSKY-rec(S_i, C_i, E_i Pr_rsky(S_i, C_i, E_i));
```

Therefore, if the i-th branch is pruned, then all branches expect the last one can also be pruned.

The pseudo-code of BBA is presented in algorithm 3. Although the worst-cast time complexity of BBA is also exponential in data size, the different search strategy, the precise terminal condition ($C = \emptyset$), and more effective pruning strategies make BBA outperform DSA by two orders of magnitude in experiments.

5 APPROXIMATE ALGORITHMS

Although efficient for small-to-medium sized uncertain datasets, exponential time complexity prevents exact algorithms from handling uncertain datasets of enormous size. Considering the hardness results of approximately computing a most-likely rskyline with accuracy guarantee, in this section, we resort to heuristic approaches to compute a near-most-likely rskyline without accuracy guarantee.

The main idea is to first find an approximate result S and then improve its quality through some local search operations.

Initialization. Recall that in algorithm 1, we first sort tuples in T based on their scores under a function $f \in \mathcal{F}$. Then, for each tuple $t \in T$, if a better replacement tuple s is found for all tuples $\mathcal{F}$-dominated by t according to our proposed reduction rules, then we add t to the pruning set P and discard all tuples $\mathcal{F}$-dominated by P in what follows. We initialize S as the rskyline of all such better replacement tuples. To this end, we modify algorithm 1 by associating a set R, which is used to collect better replacement tuples, with the pruning set P. If tuple t is added to P for point reduction

or virtual region reduction (in line 7, 10, 18 of algorithm 1), then we insert t into R. Otherwise, we add the tuple s, which makes t satisfy path reduction or region reduction, into R. Note that for region reduction, s can be easily obtained when checking the condition in line 20 algorithm 1. While for path reduction, we need to explicitly record s during the maintenance of $\beta(t)$.

Theorem 5.1. *Let S^* be the most-likely rskyline and S be the rskyline of R, then $2^n \cdot \Pr_{\text{rsky}}(S) \geq \Pr_{\text{rsky}}(S^*)$.*

Proof. We first claim that $\Pr_{\text{rsky}}(S) \geq \Pr_{\text{rsky}}(S')$, where S' is the rskyline of all tuples t with $p(t) > 1/2$. Since all tuples t with $p(t) > 1/2$ will be included in R, we know every tuple in $S' \setminus S$ is $\mathcal{F}$-dominated by some tuple in $S \setminus S'$. Let t be a tuple in $S \setminus S'$. If t is a tuple with $p(t) \leq 1/2$ and there is a tuple $s \in S' \setminus S$ such that $t \prec_{\mathcal{F}} s$. Inserting t into S' will enlarge $\Pr_{\text{rsky}}(S')$ by at least $p(t) \cdot p(s)^{-1} > 1$ since $p(s) > 1/2$. Otherwise, t is a tuple with $p(t) \leq 1/2$ and for all tuples $s \in S' \setminus S$, $t \not\prec_{\mathcal{F}} s$. According to the definition of R, t is involved in R due to path, region, or virtual region reduction rule. Based on the definitions of these rules, there is a set S such that $t \prec_{\mathcal{F}} S$ and $p(t) > (1 = p(t)) \cdot \prod_{s \in S}(1 - p(s))$. Thus, Inserting t into S' will enlarge $\Pr_{\text{rsky}}(S')$ by at least $p(t) \cdot (1 - p(t))^{-1} \cdot \prod_{s \in S}(1 - p(s))^{-1} > 1$.

Then, we prove that $2^n \cdot \Pr_{\text{rsky}}(S') \geq \Pr_{\text{rsky}}(S^*)$. Since there are at most $n - |S'|$ tuples that are not $\mathcal{F}$-dominated by S' and their existence probabilities are no more than 1/2, $\Pr_{\text{rsky}}(S') \geq 2^{-(|S|+n-|S|)} = 2^{-n}$. With the fact that $\Pr_{\text{rsky}}(S^*) \leq 1$, we have $2^n \cdot \Pr_{\text{rsky}}(S') \geq \Pr_{\text{rsky}}(S^*)$. Thus, the proof is completed. □

Local Search. Recall that in the $\mathcal{F}$-dominat graph G, we use $A(t)$ to denote the set of ancestors of a tuple t, which includes all tuples $\mathcal{F}$-dominating t, and $D(t)$ to denote the set of descendants of t, which includes all tuples $\mathcal{F}$-dominated by t. These two sets can be easily collected during the construction of G. By a abuse of notation, we also use $D(S)$ to denote $\bigcup_{t \in S} D(t)$ hereafter. To perform the local search efficiently, we maintain a set $S(t) = (A(t) \cup D(t)) \cap S$ for each tuple t not in S. Whenever a tuple t is added to S or removed from S, we can update $S(s)$ for every $s \in A(t) \cup D(t)$ in $O(n)$ time.

After initialization, we first observe that there are some redundant tuples in S, where a tuple $t \in S$ is said to be redundant if $D(S) = D(S \setminus \{t\})$. Removing such t from S will multiply $\Pr_{\text{rsky}}(S)$ by $\frac{1-p(t)}{p(t)}$. Thus, we can construct a better candidate rskyline by removing all redundant tuples t with $p(t) < 1/2$ from S. To this end, for each tuple $t \in S$ with $p(t) < 1/2$, we iterate over $D(t)$ to see if every $s \in D(t)$ has $|S(s)| > 1$. If so, we claim that t is redundant and remove t from S. This takes $O(|S|n)$ time in total.

Then, consider any tuple t not in S, there are two possible relationships between t and S. Correspondingly, we consider two insertion operations to further enlarge $\Pr_{\text{rsky}}(S)$.

Case 1: $S \prec_{\mathcal{F}} t$. In this case, we can construct a new candidate rskyline by inserting t into S and removing all tuples $\mathcal{F}$-dominating t from S, i.e., $S \leftarrow S \cup \{t\} \setminus A(t)$. The ratio change of $\Pr_{\text{rsky}}(S)$ under this operation, which is also called the gain ratio of this operation hereafter, can be expressed as

$$r(t) = p(t) \cdot \prod_{s \in A(t) \cap S} \frac{1 - p(s)}{p(s)} \cdot \prod_{s \in D(S) \setminus D(S \cup \{t\} \setminus A(t))} (1 - p(s)).$$

The first two parts of $r(t)$ are easily to computed by accessing tuples in S. Whereas, the computation of third part, which accounts for tuples $\mathcal{F}$-dominated by S but not by $S \cup \{t\} \setminus A(t)$, is time consuming. We retrieve all such tuples by accessing every $s \in D(S) \setminus D(t)$ and test if $S(s)$ is a subset of $A(t) \cap S$. Thus, the time required for computing $r(t)$ is at most $O(|S|n)$.

Case 2: $S \not\prec_{\mathcal{F}} t$. In this case, a new candidate rskyline can be constructed by inserting t into S and removing tuples $\mathcal{F}$-dominated by t from S, i.e., $S \leftarrow S \cup \{t\} \setminus D(t)$. The gain ratio of this operation can be represented as,

$$r(t) = \frac{p(t)}{1 - p(t)} \cdot \prod_{s \in D(t) \cap S} \frac{1 - p(s)}{p(s)} \cdot \prod_{s \in D(t) \setminus D(S)} \frac{1}{1 - p(s)},$$

since when t is inserted into S, whether or not tuples $\mathcal{F}$-dominated by t appear in the dataset will not affect $\Pr_{\text{rsky}}(S)$. To compute $r(t)$, we just accessing all tuples in $D(t)$, which takes at most $O(n)$ time. Note that the set $D(t) \cap S$ may be empty. If so, then the second part of $r(t)$ is simply removed.

The pseudo-code of the local search algorithm is shown in algorithm 4. We first initialize S during the procedure of data reduction, which takes at most $O(n^3)$ time. After that, we remove all redundant tuples form S in $O(|S|n)$ time. Then, we compute $r(t)$ for each $t \notin S$ and whenever there is $t \notin S$ with $r(t) \geq \tau$, where $\tau > 1$ is a hyper-parameter, we insert t into S and update $\Pr_{\text{rsky}}(S)$ accordingly. Since the initial S is a 2^{-n}-approximation and each insertion operation multiplies its probability by at least τ, the algorithm will terminal after at most $O(n \log_\tau 2)$ rounds. With the fact that the gain ratio of each insertion operation can be computed in $O(|S|n)$ time, the total time complexity of algorithm 4 is $O(|S|n^3)$.

6 EXPERIMENTS

In this section, we report the experimental study for the most-likely rskyline problem.

6.1 Experimental Setting

Datasets. We use both real and synthetic datasets in our experiments. Two real datasets are involved in our experimental study, which are widely used in previous related work [17, 21, 23]. The NBA dataset extracted from https://www.nba.com/stats/ includes statistics of 23,160 players. Each player is modeled with eight professional metrics: points, assists, steals, blocks, turnovers, rebounds, minutes, field goals made and we use a uniform distribution to randomly assign the probability that a player's performance matches the statistics. The CAR dataset consists of 323,433 used cars with four attributes: price, registration year, power, mileage, downloaded from https://data.world/data-society/used-cars-data. We convert the dataset into an uncertain dataset following the method used in [21]. We obtain the date when each car is posted on the website and infer its available probability with a hidden Markov model (HMM) as shown in Figure 4. The transition graph assumes an independent selling probability for each car and once a car is sold, it shall not return to the available state.

To further study the efficiency and scalability of proposed algorithms, we generate synthetic datasets following the generation process of previous related work [17, 23]. Specifically, we configure

Algorithm 4: Local Search Algorithm

Input: an uncertain dataset $\mathcal{D}$, a set of linear scoring functions $\mathcal{F} = \{S_\omega(\cdot) \mid \omega \in \mathbb{S}^{d-1} \wedge A \times \omega \leq b\}$
Output: a near-optimal MRSKY($\mathcal{D}, \mathcal{F}$)

$P \leftarrow \emptyset$; $R \leftarrow \emptyset$;
Sort T according to some $f \in \mathcal{F}$;
foreach $i \leftarrow 1$ **to** n **do**
  if *t_i is inserted to P for point reduction or virtual region reduction* **then** $R \leftarrow R \cup \{t_i\}$; ;
  else if *t_i is inserted to P for path reduction or region reduction* **then**
    Let s be the tuple making rules hold;
    $R \leftarrow R \cup \{s\}$;
$S \leftarrow \text{RSKY}(R, \mathcal{F})$;
$\Pr_{\text{rsky}}(S) \leftarrow \prod_{t \in S} p(t) \cdot \prod_{t \in T, S \not\prec_{\mathcal{F}} t} (1 - p(t))$;
foreach $t \in S$ **do**
  if $D(S) = D(S \setminus \{t\})$ **and** $p(t) < 1/2$ **then**
    $S \leftarrow S \setminus \{t\}$; $\Pr_{\text{rsky}}(S) \leftarrow \Pr_{\text{rsky}}(S) \cdot \frac{1-p(t)}{p(t)}$;
while $\exists t \in T \setminus S (r(t) > \tau)$ **do**
  Insert t into S;
  $\Pr_{\text{rsky}}(S) \leftarrow \Pr_{\text{rsky}}(S) \cdot r(t)$;
  foreach $t \in T \setminus S$ **do** Update $r(t)$;
return S;

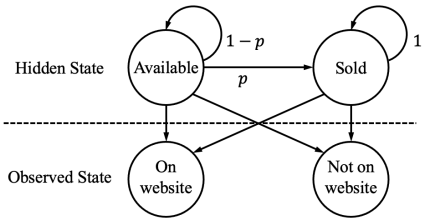


Figure 4: HMM for generating vehicles' probabilities.

the attributes of each tuple to follow two different distributions: independent (IND) or anti-correlated (ANTI), and use the benchmark data generator [8] to generate its values. Then, we uniformly generate the probability of each tuple in the interval $[\alpha - \delta, \alpha + \delta]$, where α is the probability center and δ represents one side width, i.e., $\delta = \min(\alpha, 1 - \alpha)$. This guarantees that the generated probabilities are always between $[0, 1]$.

Scoring functions. We consider the linear scoring function which is widely used in practice [12]. Given a preference ω, the score of a tuple t is defined as $S_\omega(t) = \sum_{i=1}^{d} \omega[i]t[i]$. Since the $\mathcal{F}$-dominance relation is independent from the magnitude of ω, we assume that $\sum_{i=1}^{d} \omega[i] = 1$. To describe the preferences of a user, we consider the approach of adding constraints on the preference domain $\sum_{i=1}^{d} \omega[i] = 1$. Similar to [11], we use *weak ranking* which is one of the most common types of constraints on weights [13]. For any number c of constraints, the input set is $\{\omega[i] \geq \omega[i+1] \mid 1 \leq i \leq c\}$. To determine the $\mathcal{F}$-dominance relationship between two given tuples, we compare their scores under the vertices of the preference region (see Theorem 2 in [16]), which takes at most $O(d)$ time.

Table 1: Parameter settings for experiments.

Parameter	Value Range	Default Value
Dataset Size n	10^2 - 10^6	10^4
Dimensionality d	2 - 10	6
Probability Center α	0.2 - 0.8	0.5
# of constraints c	1 - 5	$d/2$

Table 1 lists all parameters for synthetic datasets and constraints.

Algorithms. We implement the following algorithm in C++ and the source code can be accessed in [15].

- DSA: the dynamic search algorithm (see subsection 4.1).
- BBA: the branch-and-bound algorithm (see subsection 4.2).
- APA: the 2^n-approximation algorithm (see section 5).
- LSA: the local search algorithm with $\tau = 1.2$ (see section 5).

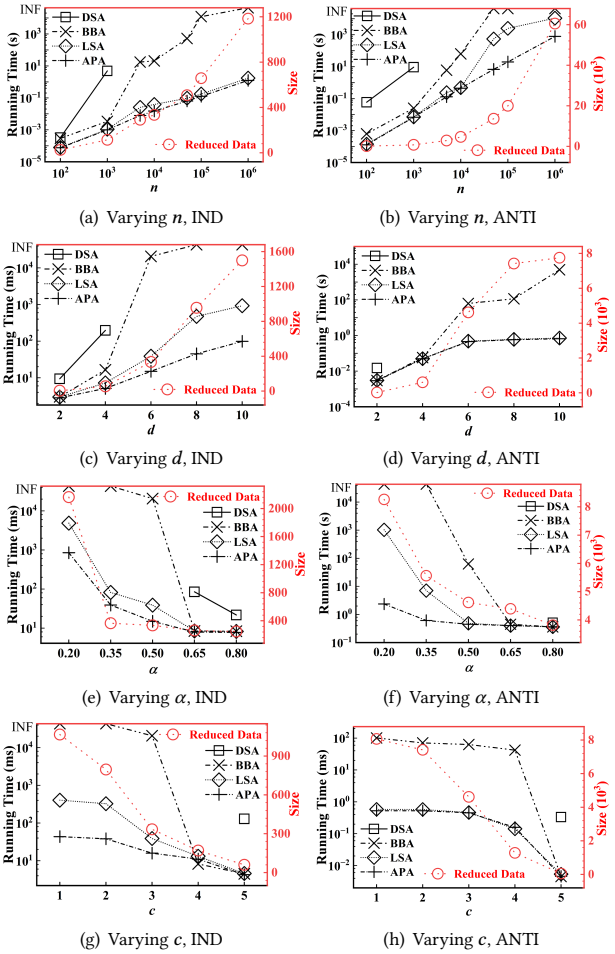
All the algorithms are complied by GNU G++ 7.5.0 with -O2 optimization and all experiments are conducted on a machine with a 3.5-GHz Intel(R) Core(TM) i9-10920X CPU, 256GB main memory, and 1TB hard disk running CentOS 7.

6.2 Performance Study

In Figure 5 and 6, we present the running time and solution quality of all algorithms versus each problem parameter. The results on synthetic datasets with varying data sizes n, data dimensions d, and probability centers α are shown in Figure 5 (a)-(f) and Figure 6 (a)-(c). Overall, the running time of all algorithms is proportional to the size of the reduced dataset, which is affected by the following factors. First, apparently, as n increases, the size of the reduced data increases. Second, the $\mathcal{F}$-dominant power of a tuple drops when d grows, which can be explained by the fact that the size of rskyline increases with d. Therefore, the number of pruned tuples also drops when d grows. As for α, since the existence probability of each tuple is uniformly generated in the interval $[\alpha - \delta, \alpha + \delta]$, where $\delta = \min(\alpha, 1 - \alpha)$, the smaller α is, the fewer tuples meet reduction rules, and thus the fewer tuples are reduced.

We continue with a fine-grained comparison of different algorithms. In terms of exact algorithms, some results of DSA are omitted since it failed to obtain any solution due to huge memory consumption. This is because that the breadth-first search strategy and coarse upper bound estimation force it store an enormous number of candidates in memory. Thanks to the different searching strategy and more effective pruning conditions, BBA outperforms DSA by around two orders of magnitude and shows better scalability. However, since the time complexity of BBA is also exponential with respect to the input data size, it can not terminal within the time limit (12 hours) under extreme parameters of n, d, and α.

Regarding approximation algorithms, both APA and LSA run faster than exact algorithms because they only find suboptimal solutions. APA runs fastest due to its simplicity. But this also leads to a poor quality of its solution as shown in Figure 6 (a)-(c). Typically, the extra time taken by LSA for local search is proportional to the quality gap between the solution of APA and the most-likely rskyline. Although without theoretical guarantee, the local search



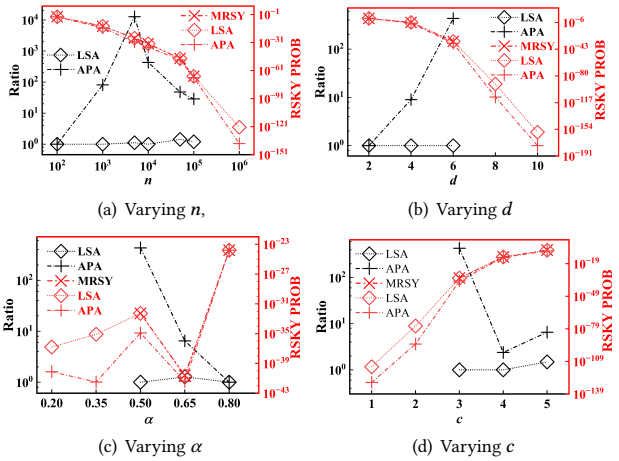
(a) Varying n, IND (b) Varying n, ANTI

(c) Varying d, IND (d) Varying d, ANTI

(e) Varying α, IND (f) Varying α, ANTI

(g) Varying c, IND (h) Varying c, ANTI

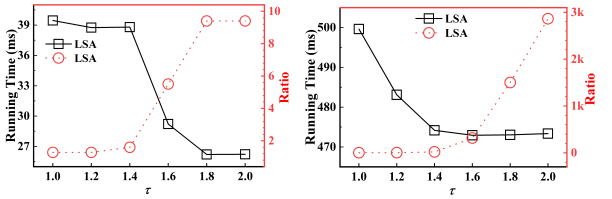
Figure 5: Running time on synthetic datasets



(a) Varying n, (b) Varying d

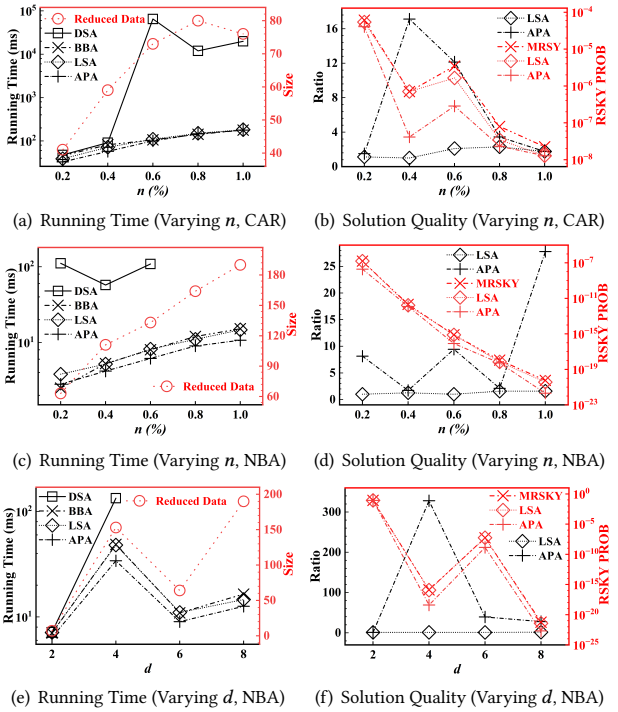
(c) Varying α (d) Varying c

Figure 6: Solution quality on synthetic datasets (IND)



(a) IND (b) ANTI

Figure 7: Effect of τ on LSA.



(a) Running Time (Varying n, CAR) (b) Solution Quality (Varying n, CAR)

(c) Running Time (Varying n, NBA) (d) Solution Quality (Varying n, NBA)

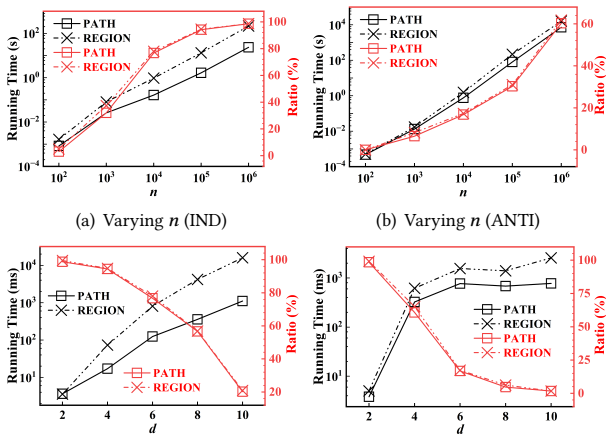
(e) Running Time (Varying d, NBA) (f) Solution Quality (Varying d, NBA)

Figure 8: Performance on real datasets

significantly improves the solution quality. In most cases, LSA returns a solution with approximation ratio less than 1.5. Figure 7 shows the effect of the hyper-parameter τ on the running time and solution quality of LSA. When τ increases, its running time decreases while its solution quality decreases dramatically.

We also evaluate the effect of the number c of constraints. Figure 5 (g)-(h) and Figure 6(d) plot the running time and solution quality with varying c. As c grows, the preference region becomes smaller, which improves the $\mathcal{F}$-dominant power of each tuple. Thus, the running time of each algorithm decreases as c grows since the size of the reduced dataset decreases. And the quality of the solution returned by LSA keeps close to optimal.

In summary, BBA outperforms DSA and is efficient enough to handle small-to-medium sized uncertain datasets. As for large scale uncertain datasets, LSA further improves the time efficiency at the expense of a bit of loss in the solution quality. Although without theoretical guarantee, LSA returns a near-optimal solution in most cases. We remark that similar trends can be observed on real datasets and the experimental results can be found in Figure 8.



(a) Varying n (IND) (b) Varying n (ANTI)

(c) Varying d (IND) (d) Varying d (ANTI)

Figure 9: Comparison between path reduction and region reduction ($\alpha = 0.2$).

6.3 Reduction rules comparison

We further compare the time efficiency and pruning ratio, i.e., the fraction of pruned tuples, of path reduction and region reduction on synthetic datasets. To avoid the impact of point reduction, we set the probability center α to 0.2. This guarantees that the probability of each tuple is in $[0, 0.4]$ and therefore no point reduction will be performed. Figure 9 reports the running time and pruning ratio of these two reductions on synthetic datasets with varying data sizes n and dimensions d. Overall, the pruning ratios of path reduction and region reduction keep roughly the same throughout all datasets. Region reduction prunes around 1.2% more tuples than path reduction. For time efficiency, when n and d increase, the running time increases for both path reduction and region reduction. Corresponding to the quadratic time complexity of path reduction and the cubic time complexity of region reduction, the running time of region reduction increases faster than that of path reduction with respect to n. As d grows, the advantage in time efficiency of path reduction over region reduction becomes more apparent. This is due to the following two reasons. First, region reduction performs more $\mathcal{F}$-dominance test than path reduction and the time consumption for each $\mathcal{F}$-dominance test increases linearly with d. Second, the $\mathcal{F}$-dominant power of a tuple decreases with the increase of d, so more tuples are tested for reducing the data size. The decrease in the $\mathcal{F}$-dominant power also explains the decrease in the pruning ratios of these two reduction rules as d grows.

To sum up, path reduction serves as an good approximation of region reduction, which improves time efficiency at the expense of a bit of loss in the pruning ratio. Thus, path reduction is more suitable for small sized uncertain datasets, where almost 90% of the runtime is consumed by performing data reduction. Alternatively, it can be used as a filter, i.e., region reduction is performed only if path reduction fails on a tuple. This can reduce a certain amount of time consumption while preserving an optimal pruning ratio.

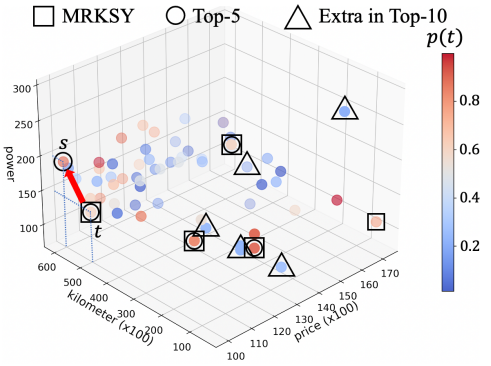


Figure 10: Case study using the CAR dataset.

6.4 Case Study: Multi-criteria Decision Making

We compare two kinds of uncertain rskyline queries, i.e., finding tuples with top-k probabilities of being in the rskyline [16] and finding the most-likely rskyline, for multi-criteria decision making on the CAR dataset. For each car, we consider three attributes: price, mileage, and power. And we still use the HMM shown in Figure 4 to infer its available probability. We set $\mathcal{F} = \{\omega_{Pr}\text{Price} + \omega_M\text{Mileage} + \omega_{Po}\text{Power} \mid \omega_{Pr} \geq \omega_M \geq \omega_{Po}\}$. Figure 10 plots a 50 samples from the dataset and results of these two queries under $\mathcal{F}$.

The most-likely rskyline consists of five cars (surrounded by squares). Thus, for comparison, we retrieve tuples with top-5 probabilities of being in the rskyline (surround by circles). It is observed that the top-5 result does not guarantee that tuples in it are not mutually $\mathcal{F}$-dominated. In other word, the top-5 result is not a valid rskyline in any possible world. As shown in Figure 10, t is always a better choice than s. This implies that it is meaningless to provide s to a shopper for further decision-making. A subsequent rskyline computation can be used to filter such tuples. However, this affects the diversity of the top-5 result. Meanwhile, the top-5 result does not provide any option with low mileage, even the one included in the most-likely rskyline. To enlarge the coverage of the most-likely rskyline, we reset $k = 10$ (lowest value). But at the same time, this also includes more tuples that are $\mathcal{F}$-dominated by others. It is challenging to specifying a proper k to balance diversity and incomparability. The most-likely rskyline does not has this tradeoff. It always provides the insights for a set of non-$\mathcal{F}$-dominated tuples with the maximum probability of being the rskyline result across all possible worlds. And almost all tuples in the most-likely rskyline have a high probability of being in the rskyline. However, due to the enormous number of candidate rskylines, its joint probability may be naturally small.

As a conclusion, these two queries provide complement perspectives of performing rskyline queries on uncertain datasets and may together serve as a better guidance for multi-criteria decision making on uncertain datasets. For example, a possible appropriate method is to first find the most-likely rskyline, then return a subset of the most-likely rskyline that also has a high individual probability of being in the rskyline.

7 RELATED WORK

In this section, we elaborate on previous work that are related to the most-likely rskyline problem.

Uncertain skyline (rskyline). The work most relevant to ours is [4, 21], which studied the problem of finding the most-likely skyline over uncertain datasets. Liu et al. proved that the most-likely skyline problem is NP-hard when $d \geq 3$ [21]. They transformed the most-likely skyline problem into an integer programming problem and design an integer programming based baseline algorithm. They also designed a search algorithm based on dynamic programming with pruning and early termination techniques. To further improve efficiency, they also proposed several data reduction and data partition methods to pre-eliminate unnecessary tuples and reduce the input size. Agrawal et al. introduced an $O(n \log n)$-time exact algorithm for the most-like skyline problem with $d = 2$ based on dynamic programming [4]. And they proved that when $d \geq 3$, the most-likely skyline problem can not be approximated within $2^{-O(n^{1-\delta})}$ in polynomial time for any $\delta > 0$, unless P = NP. They also gave a 2^n-approximation algorithm which almost completes the study of the most-likely skyline problem. As a further extension, Liu et al. investigated the problem of finding a set of k objects such that the probability of forming the skyline is maximized [20]. We show that the most-likely rskyline problem studied in this paper is much harder than the most-likely skyline problem due to the introduction of a user-specified function set $\mathcal{F}$. In particular, we prove that even when $d = 2$, the most-likely rskyline problem is NP-hard and is also hard to approximate. Moreover, the proposed approaches in this paper are different from previous work in that (1) the exact algorithm uses new search strategy and pruning condition, which outperforms the adapted state-of-the-art method for the most-likely skyline problem by at least two orders of magnitude, (2) the approximate algorithm uses a different approach to find a 2^n-approximate solution and it is proved to be better than the one derived in [4], and (3) the approximate algorithm uses local search strategies to further improve its quality.

Instead of considering skyline sets, Pei et al. are the first to investigate how to conduct skyline query on uncertain datasets from the perspective of individual tuples [23]. They proposed a notion of p-skyline query which aims to identify those tuples whose probability of being in the skyline is higher than a user-specified threshold. Considering the difficulty of specifying a suitable threshold, follow-up research extended the previous work by computing skyline probabilities of all tuples [2, 5, 6, 17] and finding tuples with top-k skyline probability [25]. Recently, Gao et al. studied the problem of computing rskyline probabilities of all tuples [16]. Our work differs theirs since focusing on a set of tuples makes the problem challenging as there might be an exponential number of candidate sets. These two different interpretations of queries over uncertain datasets represent different conceptual notions and are complementary to each other.

Uncertain top-k. Somehow related to the most likely rskyline problem is the most-likely top-k problem. In [24], the authors proposed two definitions for the most-likely top-k problem. Specifically, the U-Top$k$ problem aims to find a set of k tuples with the highest probability to be the top-k result in all possible worlds, and the U-kRanks problem aims to find a set of k tuples where the i-th tuple has the largest probability to be ranked at the i-th position in all possible worlds. However, it has been pointed out that the requirement of an exact weight vector in top-k query is hardly realistic [22]. Our work can be regarded as extending the weight vector in the most-likely top-k problem into a region $\mathcal{F}$ which can be estimated by a preference learning algorithm. And the non-$\mathcal{F}$-dominant tuples will preserve best score with any weight in $\mathcal{F}$.

8 CONCLUSION

In this paper, we study the most-likely rskyline problem which aims to find a set of non-$\mathcal{F}$-dominated tuples with the maximum probability of occurrence as the rskyline across all possible worlds. This problem extends the most-likely skyline problem by serving the specific preference of an individual user. We prove this problem is much harder than its special version. Specifically, even when $d = 2$, it is NP-hard to exactly solve the most-likely rskyline problem, and it is also NP hard to find a $2^{-O(n^{1-\delta})}$-approximate solution for any $\delta > 0$. Consider the hardness of this problem, we first design a series of reduction rules to reduce the input data size as much as possible meanwhile guaranteeing the correctness of the result. Then, we propose two exact algorithms DSA and BBA respectively. Although BBA is much more efficient than DSA due to different search strategy and more effective pruning conditions, the exponential time complexity prevents it handling large scale uncertain datasets. Therefore, we propose a polynomial-time 2^n-approximation algorithm and further improve the quality of its solution with some local search operations. Experimental results demonstrate the efficiency of the approximation algorithm and the result derived after local search is nearly optimal. For further directions, given the importance of flexible output-size control in practice, it is worthwhile to attach an additional input k to the problem, i.e., study the problem of finding a k-size set of non-$\mathcal{F}$-dominated tuples with the maximum probability of occurrence as a subset of the rskyline across all possible worlds. This can also address the problem that the probability of the most-likely rskyline is small to some extent.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (NSFC) Grant NOs. 61832003.

REFERENCES

[1] Serge Abiteboul, Paris C. Kanellakis, and Gösta Grahne. 1987. On the Representation and Querying of Sets of Possible Worlds. In *Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference, San Francisco, CA, USA, May 27-29, 1987*, Umeshwar Dayal and Irving L. Traiger (Eds.). ACM Press, 34–48. https://doi.org/10.1145/38713.38724

[2] Peyman Afshani, Pankaj K. Agarwal, Lars Arge, Kasper Green Larsen, and Jeff M. Phillips. 2013. (Approximate) Uncertain Skylines. *Theory Comput. Syst.* 52, 3 (2013), 342–366. https://doi.org/10.1007/s00224-012-9382-7

[3] Charu C Aggarwal and S Yu Philip. 2008. A survey of uncertain data algorithms and applications. *IEEE Transactions on knowledge and data engineering* 21, 5 (2008), 609–623.

[4] Akash Agrawal, Yuan Li, Jie Xue, and Ravi Janardan. 2020. The most-likely skyline problem for stochastic points. *Comput. Geom.* 88 (2020), 101609. https://doi.org/10.1016/j.comgeo.2020.101609

[5] Mikhail J. Atallah and Yinian Qi. 2009. Computing all skyline probabilities for uncertain data. In *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*, Jan Paredaens and Jianwen Su (Eds.). ACM, 279–287. https://doi.org/10.1145/1559795.1559837

[6] Mikhail J. Atallah, Yinian Qi, and Hao Yuan. 2011. Asymptotically efficient algorithms for skyline probabilities of uncertain data. *ACM Trans. Database Syst.* 36, 2 (2011), 12:1–12:28. https://doi.org/10.1145/1966385.1966390

[7] George Beskales, Mohamed A. Soliman, and Ihab F. Ilyas. 2008. Efficient search for the top-k probable nearest neighbors in uncertain databases. *Proc. VLDB Endow.* 1, 1 (2008), 326–339. https://doi.org/10.14778/1453856.1453895

[8] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. 2001. The Skyline Operator. In *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*, Dimitrios Georgakopoulos and Alexander Buchmann (Eds.). IEEE Computer Society, 421–430. https://doi.org/10.1109/ICDE.2001.914855

[9] Lin Chen, Moran Feldman, and Amin Karbasi. 2018. Weakly Submodular Maximization Beyond Cardinality Constraints: Does Randomization Help Greedy?. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 803–812. http://proceedings.mlr.press/v80/chen18b.html

[10] Lu Chen, Yunjun Gao, Aoxiao Zhong, Christian S. Jensen, Gang Chen, and Baihua Zheng. 2017. Indexing metric uncertain data for range queries and range joins. *VLDB J.* 26, 4 (2017), 585–610. https://doi.org/10.1007/s00778-017-0465-6

[11] Paolo Ciaccia and Davide Martinenghi. 2017. Reconciling Skyline and Ranking Queries. *Proc. VLDB Endow.* 10, 11 (2017), 1454–1465. https://doi.org/10.14778/3137628.3137653

[12] James S. Dyer and Rakesh K. Sarin. 1979. Measurable Multiattribute Value Functions. *Oper. Res.* 27, 4 (1979), 810–822. https://doi.org/10.1287/opre.27.4.810

[13] Yun Seong Eum, Kyung Sam Park, and Soung Hie Kim. 2001. Establishing dominance and potential optimality in multi-criteria analysis with imprecise weight and value. *Comput. Oper. Res.* 28, 5 (2001), 397–409. https://doi.org/10.1016/S0305-0548(99)00124-0

[14] Xiangyu Gao. 2023. *Full Paper.* Retrieved April 28, 2023 from https://github.com/gaoxy914/Most-likely-rskyline/blob/main/main.pdf

[15] Xiangyu Gao. 2023. *Source Code.* Retrieved April 23, 2023 from https://github.com/gaoxy914/Most-likely-rskyline

[16] Xiangyu Gao, Jianzhong Li, and Dongjing Miao. 2023. Computing All Restricted Skyline Probabilities for Uncertain Data. *CoRR* abs/2303.00259 (2023). https://doi.org/10.48550/arXiv.2303.00259 arXiv:2303.00259

[17] Dongwon Kim, Hyeonseung Im, and Sungwoo Park. 2012. Computing Exact Skyline Probabilities for Uncertain Databases. *IEEE Trans. Knowl. Data Eng.* 24, 12 (2012), 2113–2126. https://doi.org/10.1109/TKDE.2011.164

[18] Hans-Peter Kriegel, Peter Kunath, Martin Pfeifle, and Matthias Renz. 2006. Probabilistic Similarity Join on Uncertain Data. In *Database Systems for Advanced Applications, 11th International Conference, DASFAA 2006, Singapore, April 12-15, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 3882)*, Mong-Li Lee, Kian-Lee Tan, and Vilas Wuwongse (Eds.). Springer, 295–309. https://doi.org/10.1007/11733836_22

[19] Jinfei Liu, Li Xiong, Qiuchen Zhang, Jian Pei, and Jun Luo. 2021. Eclipse: Generalizing kNN and Skyline. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 972–983. https://doi.org/10.1109/ICDE51399.2021.00089

[20] Jinfei Liu, Haoyu Zhang, Li Xiong, Haoran Li, and Jun Luo. 2015. Finding Probabilistic k-Skyline Sets on Uncertain Data. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, James Bailey, Alistair Moffat, Charu C. Aggarwal, Maarten de Rijke, Ravi Kumar, Vanessa Murdock, Timos K. Sellis, and Jeffrey Xu Yu (Eds.). ACM, 1511–1520. https://doi.org/10.1145/2806416.2806452

[21] Xingjie Liu, De-Nian Yang, Mao Ye, and Wang-Chien Lee. 2013. U-Skyline: A New Skyline Query for Uncertain Databases. *IEEE Trans. Knowl. Data Eng.* 25, 4 (2013), 945–960. https://doi.org/10.1109/TKDE.2012.33

[22] Kyriakos Mouratidis and Bo Tang. 2018. Exact Processing of Uncertain Top-k Queries in Multi-criteria Settings. *Proc. VLDB Endow.* 11, 8 (2018), 866–879. https://doi.org/10.14778/3204028.3204031

[23] Jian Pei, Bin Jiang, Xuemin Lin, and Yidong Yuan. 2007. Probabilistic Skylines on Uncertain Data. In *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*, Christoph Koch, Johannes Gehrke, Minos N. Garofalakis, Divesh Srivastava, Karl Aberer, Anand Deshpande, Daniela Florescu, Chee Yong Chan, Venkatesh Ganti, Carl-Christian Kanne, Wolfgang Klas, and Erich J. Neuhold (Eds.). ACM, 15–26. http://www.vldb.org/conf/2007/papers/research/p15-pei.pdf

[24] Mohamed A. Soliman, Ihab F. Ilyas, and Kevin Chen-Chuan Chang. 2007. Top-k Query Processing in Uncertain Databases. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, Rada Chirkova, Asuman Dogac, M. Tamer Özsu, and Timos K. Sellis (Eds.). IEEE Computer Society, 896–905. https://doi.org/10.1109/ICDE.2007.367935

[25] Ying Zhang, Wenjie Zhang, Xuemin Lin, Bin Jiang, and Jian Pei. 2011. Ranking uncertain sky: The probabilistic top-k skyline operator. *Inf. Syst.* 36, 5 (2011), 898–915. https://doi.org/10.1016/j.is.2011.03.008

A PROOFS AND ALGORITHM FOR REDUCTION RULES

Algorithm 5: Point and Region Reduction

Input: an uncertain dataset $\mathcal{D} = (T, p)$, a set of monotone scoring functions $\mathcal{F}$
Output: the reduced dataset $\mathcal{D}$

$P \leftarrow \emptyset$;
Sort $T \leftarrow \{t_1, t_2, \cdots, t_n\}$ according to some $f \in \mathcal{F}$;
foreach $i \leftarrow 1$ **to** n **do**
  if $P \not\prec_{\mathcal{F}} t_i$ **then**
    if $p(t_i) > 1/2$ **then** $P \leftarrow P \cup \{t_i\}$; ;
    else
      if $\exists 1 \le j < i (t_j \prec_{\mathcal{F}} t_i \wedge \frac{p(t_j)}{(1-p(t_j))(1-p(t_i))} \cdot \prod_{\forall t_k, t_j \prec_{\mathcal{F}} t_k \prec_{\mathcal{F}} t_i} (1-p(t_k))^{-1} > 1$ **then**
        $P \leftarrow P \cup \{t_i\}$;
  else
    Remove t_i from $\mathcal{D}$;
return $\mathcal{D}$;

Reduction Rule 6 (Point Reduction [21]). *Given an uncertain dataset $\mathcal{D} = (T, p)$ and a set of monotone scoring functions $\mathcal{F}$, if there exists a tuple t such that $p(t) > 1/2$, then all tuples $\mathcal{F}$-dominated by t can be safely discarded from $\mathcal{D}$.*

Proof. Let s be a tuple that is $\mathcal{F}$-dominated by t. We prove that any candidate rskyline S containing s is not the most-likely rskyline and removing all such s does not affect the rskyline probability of any most-likely rskyline. Since $t \prec_{\mathcal{F}} s$, we construct a new candidate rskyline S' by inserting t into S and removing all tuples in S that are $\mathcal{F}$-dominated by t, i.e., $S' = S \cup \{t\} \setminus \{s \in S \mid t \prec_{\mathcal{F}} s\}$. The relationship between $\Pr_{\text{rsky}}(S)$ and $\Pr_{\text{rsky}}(S')$ is

$$\frac{\Pr_{\text{rsky}}(S')}{\Pr_{\text{rsky}}(S)} = \frac{p(t)}{1-p(t)} \cdot \prod_{s \in S, t \prec_{\mathcal{F}} s} \frac{1}{p(t)} \cdot \prod_{x \in T, t \prec_{\mathcal{F}} x \wedge S \not\prec_{\mathcal{F}} x} \frac{1}{1-p(x)} \ge \frac{p(t)}{1-p(t)} > 1.$$

Then, let S^* denote a most-likely rskyline. S^* must $\mathcal{F}$-dominate t or include t, otherwise inserting t into S^* will result in a strictly better candidate rskyline for the same reason stated above. Since $S^* \prec_{\mathcal{F}} t$ or $t \in S^*$ whether tuples $\mathcal{F}$-dominated by t appear in the dataset will not affect $\Pr_{\text{rsky}}(S^*)$ according to Equation 2. □

Reduction Rule 7 (Region Reduction [21]). *Given an uncertain dataset $\mathcal{D} = (T, p)$ and a set of monotone scoring functions $\mathcal{F}$, if there exist two tuples t and s such that $t \prec_{\mathcal{F}} s$ and*

$$\frac{p(t)}{(1-p(t))(1-p(s))} \prod_{x \in T, t \prec_{\mathcal{F}} x \prec_{\mathcal{F}} s} (1-p(x))^{-1} > 1,$$

then all tuples $\mathcal{F}$-dominated by s can be safely discarded from $\mathcal{D}$.

Proof. Also, let x be a tuple that is $\mathcal{F}$-dominated by s and S be a candidate rskyline containing x. We construct a new candidate rskyline S' by inserting t into S. Since $t \prec_{\mathcal{F}} s \prec_{\mathcal{F}} x$, the relationship between $\Pr_{\text{rsky}}(S)$ and $\Pr_{\text{rsky}}(S')$ is

$$\frac{\Pr_{\text{rsky}}(S')}{\Pr_{\text{rsky}}(S)} = \frac{p(t)}{1-p(t)} \cdot \prod_{s \in S, t \prec_{\mathcal{F}} s} \frac{1}{p(t)} \cdot \prod_{x \in T, t \prec_{\mathcal{F}} x \wedge S \not\prec_{\mathcal{F}} x} \frac{1}{1-p(x)} \ge \frac{p(t)}{(1-p(t))(1-p(x))} \prod_{y \in T, t \prec_{\mathcal{F}} y \prec_{\mathcal{F}} x} (1-p(y))^{-1} \ge \frac{p(t)}{(1-p(t))(1-p(s))} \prod_{y \in T, t \prec_{\mathcal{F}} y \prec_{\mathcal{F}} s} (1-p(s))^{-1} > 1.$$

Then, let S^* denote a most-likely rskyline. S^* must $\mathcal{F}$-dominate s or include s, otherwise inserting t into S^* will result in a strictly better candidate rskyline since

$$p(t) > (1-p(t)) \cdot (1-p(s)) \cdot \prod_{x \in T, t \prec_{\mathcal{F}} x \prec_{\mathcal{F}} s} (1-p(x)).$$

Since $S^* \prec_{\mathcal{F}} s$ or $s \in S^*$ whether tuples $\mathcal{F}$-dominated by s appear in the dataset will not affect $\Pr_{\text{rsky}}(S^*)$ according to Equation 2. □

Reduction Rule 8 (Path Reduction). *Given an uncertain dataset $\mathcal{D} = (T, p)$ and a set of monotone scoring functions $\mathcal{F}$, if there exists two tuples t and s such that $t \prec_{\mathcal{F}} s$ and*

$$\frac{p(t)}{(1-p(t))(1-p(s))} \cdot \prod_{x \in Int(p(t,s))} (1-p(x))^{-1} > 1$$

where $p(t, s)$ is a path from t to s in the $\mathcal{F}$-dominant graph G, then all tuples $\mathcal{F}$-dominated by s can be safely discarded from $\mathcal{D}$.

Proof. Since $Int(p(t, s))$ is a subset of $D(t) \cap A(s) = \{x \in T \mid t \prec_{\mathcal{F}} x \prec_{\mathcal{F}} s\}$, path reduction is a sufficient condition for region reduction. This completes the proof. □

Reduction Rule 9 (Virtual Region Reduction). *Given an uncertain dataset $\mathcal{D} = (T, p)$ and a set of monotone scoring functions $\mathcal{F}$, if there exists a tuple t and a set S such that $t \prec_{\mathcal{F}} S$ and*

$$\frac{p(t)}{1-p(t)} \cdot \prod_{s \in S} (1-p(s))^{-1} > 1,$$

let $t' = (\max_{s \in S} s[1], \cdots, \max_{s \in S} s[d])$, then all tuples $\mathcal{F}$-dominated by t' can be safely discarded from $\mathcal{D}$.

Proof. Let s be a tuple that is $\mathcal{F}$-dominated by t' and S be a candidate rskyline containing s. We construct a new candidate rskyline S' by inserting t into S. Because $S \prec t'$, we know all tuples in S $\mathcal{F}$-dominate t' with respect to any set $\mathcal{F}$. Since $t \prec_{\mathcal{F}} t' \prec_{\mathcal{F}} s$, the relationship between $\Pr_{\text{rsky}}(S)$ and $\Pr_{\text{rsky}}(S')$ is

$$\frac{\Pr_{\text{rsky}}(S')}{\Pr_{\text{rsky}}(S)} = \frac{p(t)}{1-p(t)} \cdot \prod_{s \in S, t \prec_{\mathcal{F}} s} \frac{1}{p(t)} \cdot \prod_{x \in T, t \prec_{\mathcal{F}} x \wedge S \not\prec_{\mathcal{F}} x} \frac{1}{1-p(x)} \ge \frac{p(t)}{(1-p(t))(1-p(s))} \prod_{x \in T, t \prec_{\mathcal{F}} x \prec_{\mathcal{F}} s} (1-p(x))^{-1} \ge \frac{p(t)}{1-p(t)} \prod_{x \in T, t \prec_{\mathcal{F}} x \prec_{\mathcal{F}} t'} (1-p(x))^{-1} > 1.$$

Then, let S^* denote a most-likely rskyline. S^* must $\mathcal{F}$-dominate or include some tuple in S, otherwise inserting t into S^* will result in a strictly better candidate rskyline since

$$p(t) > (1 - p(t)) \cdot \prod_{s \in T, t \prec_{\mathcal{F}} s \prec_{\mathcal{F}} t'} (1 - p(x)).$$

Therefore, whether tuples $\mathcal{F}$-dominated by s appear in the dataset will not affect $\Pr_{\text{rsky}}(S^*)$ according to Equation 2. □

The pseudocode of algorithm proposed in [21] for performing point reduction and region reduction is presented in algorithm 5. Since line 7 takes $O(n^2)$ time to check whether there is a tuple t_j makes the condition hold, the worst-case time complexity of algorithm 5 is $O(n^3)$.