REVISION LETTER

Thank you very much for your valuable feedback. We appreciate the depth of your reviews and the work you spent towards our paper. We carefully implemented your suggestions and high-lighted the changes in blue in the revised paper. The major changes of our paper are summarized as follows.

(1) We updated the real-life applications in section I.

(2) We rewrote the sixth paragraph of section I to emphasized the difference between our work and the existing work. And we also discussed detailed differences in section II.

(3) We redefined the rskyline probability based on possible world semantics [29] in section III.B, which equals to previous one, and redesigned example 1 and figure 1 to explain it.

(4) We removed summary of notations due to lack of space.

(5) We formally defined linear constraints with matrix inequality in the first paragraph of section IV.

(6) We added an enumeration-based based baseline.

(7) We redesigned example 3, 4, and figure 3 to provided more intuitive explanation of the proposed algorithm.

(8) We improved the description of multi-level strategy.

(9) We added two real datasets IIP [5] and CAR [6]. We also expanded the NBA dataset in terms of both data cardinality and data dimensionality. Now, it contains 354,698 game records of 1,878 players with 8 attributes. Code for generating these datasets can found at [43]. Details of these three datasets and methods to generate existence probabilities for records in these datasets can be found in the first paragraph of section VI.A.

(10) We removed the table for parameter settings and introduced them at the beginning of section VI.C. We also added references for parameter settings and involved constraints.

(11) We re-conducted the experiments for verifying the effectiveness of ARSP on NBA in section IV.B. We compared ARSP and the aggregated rskyline by selecting four representative players. And all statements were derived based on their score distributions (see figure 4).

(12) We added experiments to evaluate the effect of each parameter on the number of instances with none-zero rskyline probabilities. The results were plotted at the second y-axis of each experimental figure in section IV.C.

(13) We reported the performance of proposed algorithms on real and correlated synthetic datasets in section VI.C.

(14) We replotted experimental results of section VI.C.

The following is our detailed responses to your comments.

Detailed Comments to Meta Review

R1. We improved the motivation for the studied problem from the following two aspects.

(1) We provided two more real-life scenarios that the studied problem can be applied in the revised introduction. Since Reviewer 4 O1 pointed out that booking probabilistic hotels is not common in practice as users usually select a specific hotel, instead of a region, we rewrote the e-commerce scenario with car rental services provided on Hotwire (www.hotwire.com/car-rentals/). In this case, the uncertain data comes from the sale strategy and $\mathcal{F}$ describes a user's imprecise preference for cars' attributes. We also introduced the application of our problem to stock market prediction services. The existence probability of data reflects the confidence of the prediction and $\mathcal{F}$ describes the user's rough preference for selecting stocks. In addition, our problem can also be used in sensor datasets (e.g., IIP in our experiments). The uncertainty comes from the limitations of equipment, and $\mathcal{F}$ contains the scoring functions for evaluating the emergence of an iceberg. In job hunting scenario described in [25], the probability of a job shows the likelihood of it being offered to the applicant. And, $\mathcal{F}$ to our problem can be used to characterize the applicant's requirement that if the job-security decrease by 1, then salary must increase by 1000 (e.g., $\mathcal{F} = \{\omega_1 Salary + \omega_2 Security \mid \omega_1 \geq 1000\omega_2\}$).

(2) We emphasized the difference between our work and the existing work. Previous researches on uncertain datasets performed different tasks, such as skyline queries [7], [15]–[17], top-k queries [18]–[22], etc. They only considered uncertainty in datasets but not in user preferences. However, as stated in [4], [23], determining a precise scoring function for a user is unrealistic, which is required for top-k queries, and skyline queries lack personalization. The input $\mathcal{F}$ to our problem can contain all possible scoring functions of the user. This can be efficiently obtained by learning techniques like [24]. And works for uncertain preferences [4], [10], [12], [23] only consider datasets without uncertainty. Our work extend the $\mathcal{F}$-dominance-based operators to uncertain datasets.

R2. It was pointed out the independence assumption may be violated in player selection scenario. However, the assumption do hold in a lot of applications. In e-commerce scenario, when a customer rents a compact SUV from Hotwire, he may be offered a Ford Eco Sport or similar with the predetermined probability. This will not affect the probability of offering any car to him when he rents a mid-size sedan. Thus, two objects (probabilistic cars here) are independent. In prediction service, the learning algorithm's output for two stocks is independent. In sensor dataset, the confidence of records collected by a sensor only depends on its own device accuracy. Moreover, the same assumption can be found in almost all work on uncertain skyline queries [6]–[8], [15]–[17], [25]–[27]. This also explains the meaning of existence probability. In probabilistic car renting, it means the likelihood that a user can get Ford Eco Sport if he rents a compact SUV. And in prediction service and sensor dataset, it characterize the confidence of the data.

R3. The reason why the KD-tree is constructed on-the-fly in algorithm 1, while all instances are organized into an R-tree in advance in algorithm 2 is as follows. If the uncertain dataset is static, given any input $\mathcal{F}$, algorithm 2 can process $\mathcal{F}$ by performing the best-first traverse on the same R-tree. Different $\mathcal{F}$ only affect the min-heap used in Algorithm 2 but do not affect the R-tree structure on I. And when the dataset is updated, it is also efficient to update the R-tree. However, given an input $\mathcal{F}$, algorithm 1 first needs to compute $S_V(t)$ for each $t \in I$, where V is the set of vertices of the preference region in $\mathcal{F}$, and then constructs a KD-tree on $I' = \{S_V(t) \mid t \in I\}$. Different $\mathcal{F}$ result in different sets I', thus also different KD-trees. Therefore, we do not consider the construction time of the R-tree on I in the efficiency evaluation of algorithm 2. This is reasonable since the construction is a

one-time cost for all subsequent input $\mathcal{F}$. The same assumption can also be found in previous related work [4]. And we consider the construction time of the KD-tree in the efficiency evaluation of algorithm 1 and the construction time of m aggregated R-trees in the efficiency evaluation of algorithm 2, since they all need to be built from scratch for different $\mathcal{F}$. We clarified this in the first paragraph of section IV.C.

R4. We enhanced the experiments according to the comments of each reviewer. Details can be found at (6), (9), (11), (12), (13), and, (14) of the change summary.

Detailed Comments to Reviewer 3

O1. As stated in the response to Meta Review-R1, our problem considered uncertainty not only in datasets, but also in users' preferences. Unlike probabilistic top-k skyline queries (e.g., [16]), the additional input $\mathcal{F}$ describes a user's preference, which can contain any number of scoring functions. This is also different from uncertain the top-k query (e.g., [18]) since it requires that only one exact scoring function can be provided. Moreover, since the rskyline result is always a subset of the skyline result, given a set of scoring functions $\mathcal{F}$, a skyline object may be $\mathcal{F}$-dominated by other objects. Therefore, an object with high skyline probabilities may have low rskyline probabilities, making it less attractive under $\mathcal{F}$. And work like [4] considered how to answer queries with uncertain preference on certain datasets. As for [28], it only considered the priority of dimensions. Whereas, $\mathcal{F}$ to our problem can describes more kinds of the user's preferences.

O2. Our problem is completely different from theirs. These works aim to retrieve k uncertain objects with the highest skyline probabilities. We aim to compute all uncertain objects' rskyline probabilities. The rskyline probabilities can better serve the specific preferences of individual users, i.e., given different input $\mathcal{F}$, the rskyline probabilities of uncertain objects are different. However, the skyline probability of an uncertain object also remains the same for any user.

O3. In the revised version, we pointed out the differences between our problem and existing ones (section I and II).

O4. The algorithm is introduced in the first paragraph of section IV.A and experimental studies can be found in section VI. However, due to its exponential time complexity, the baseline cannot finish within the limited time even under the least time-consuming parameter settings.

O5. Please refer to (11) of the change summary.

O6. Please refer to the reply to Meta Review-R1 and R2.

O7. Please see the response to Meta Review-R3 for details.

O8. This is a helpful comment as the original NBA dataset is too small to verify the performance of proposed algorithms. As stated in (9) of the change summary, we not only expanded the NBA dataset in terms of data cardinality and attributes number, but also added two real datasets.

O9. Parameter settings for synthetic datasets followed the previous work on skyline queries processing over uncertain datasets [7], [8]. Since they did not verify the effects of cnt, l, and ϕ, we only set default values as they did, and then set ranges ourselves. As for methods to generate constraints, we involved the WR method as specifying rankings on attributes is one of the most common ways to express preference [9] and WR was also considered in the first work for rskyline queries [10]. Moreover, inspired by preference learning techniques like [24], we considered the IM method. In practice, it is easy and cheap to obtain a user's rough preferences by asking him/her to compare several pairs of options. These references were all added in the revised paper.

Detailed Comments to Reviewer 4

O1. (1) Please refer to the response to Meta Review-R1.

(2) In the sixth paragraph of Section I, we added a discussion about why the aggregated result may be meaningless and fails to provide comprehensive view on uncertain datasets.

O2. (1) We involved three real datasets in the revised experiments. See (9) of the change summary for details.

(2) Actually, the original baseline is an extension of rskyline algorithms on certain datasets proposed in [10]. We renamed it LOOP in the revised version and compared it with our proposed methods under all parameter configurations.

(3) As correlated datasets are the least challenging when it comes to computing skyline or rskyline, we ignored them for lack of space. In the revised paper, we reported the results on correlated datasets in section IV.C.

(4) This suggestion is quite helpful. We added experimental results about this number. Please refer to (12) of the change summary for details.

O3. Thanks for careful reviewing. We checked our paper to ensure that all symbols were used after formal definition.

Detailed Comments to Reviewer 5

D1. Thanks for your advice. We replotted all experimental figures following your comment.

D2. As stated in (9) of the change summary, we involved three real datasets in the revised experiments. We also plotted the baseline results as a reference in every experimental figure.

D3 D5. We redesigned Example 3 and 4 to provide more intuitive explanation of the proposed algorithm. In Example 3, we used an uncertain dataset to illustration the reduction. We explained how to partition the data space with an instance and plotted the hyperplane used to find all instances that $\mathcal{F}$-dominate this instance in each region. In Example 4, we continued with one resulted region in Example 4 and showed the information kept for each face in that region. Then, we explained how to use this information to compute the rskyline probability of this instance.

D4. We rewrote the description following your suggestion.

D6. Sorry for our mistake. We replotted these two figures and checked the other figures to ensure their readability.

D7. For lack of space, we finally putted them on arxiv [33].

D8. Under general linear constraints, the reduction established in Section V.A relies on the number of vertices in V, where V is set of vertices of the preference region (see Theorem 2). In such case, the number of point location queries performed for each instance is $|V|$. Since $|V| = O(c^{\lfloor d/2 \rfloor})$, where c is the number of constraints, we can not use the introduced strategies to improve the query time.

D9. Thanks for careful reviewing. We checked the entire paper to avoid such errors.

Computing All Restricted Skyline Probabilities 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@siat.ac.cn

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

***Abstract*—Restricted skyline (rskyline) query is widely used in multi-criteria decision making. It generalizes the skyline query by additionally considering a set of personalized scoring functions $\mathcal{F}$. Since uncertainty is inherent in datasets for multi-criteria decision making, we study rskyline queries on uncertain datasets from both complexity and algorithm perspective. We formalize the problem of computing rskyline probabilities of all data items and show that no algorithm can solve this problem in truly subquadratic-time, unless the orthogonal vectors conjecture fails. Considering that linear scoring functions are widely used in practical applications, we propose two efficient algorithms for the case where $\mathcal{F}$ is a set of linear scoring functions whose weights are described by linear constraints, one with near-optimal time complexity and the other with better expected time complexity. For special linear constraints involving a series of weight ratios, we further devise an algorithm with sublinear query time and polynomial preprocessing time. Extensive experiments demonstrate the effectiveness, efficiency, scalability, and usefulness of our proposed algorithms.**

***Index Terms*—Uncertain data, probabilistic restricted skyline**

I. Introduction

Restricted skyline (rskyline) query is a powerful tool for supporting multi-criteria decision making, which extends the skyline query by serving the specific preferences of an individual user. Given a dataset of multidimensional objects and a set of monotone scoring functions $\mathcal{F}$, the rskyline query retrieves the set of objects that are not $\mathcal{F}$-dominated by any other object. Here an object t is said to $\mathcal{F}$-dominate another object s if t scores better than s under all functions in $\mathcal{F}$. It was shown that objects returned by the rskyline query preserve the best score with respect to any function in $\mathcal{F}$, and the result size is usually smaller compared to the skyline query [10]. Due to its effectiveness and wide applications, many efficient algorithms have been proposed to efficiently answer rskyline queries on datasets where no uncertainty is involved [10], [12].

However, uncertainty is inherent in datasets used for multi-criteria decision making caused by limitations of measuring equipment, privacy issues, data incompleteness, outdated data sources, etc. [13]. Below are two application scenarios that involve answering rskyline queries on uncertain datasets.

E-commerce Scenario: Probabilistic selling is a novel sales strategy in e-commerce [14]. Sellers create probabilistic products by setting the probability of getting any one from a set of products. A typical case is renting cars on *Hotwire* (www.hotwire.com/car-rentals/). The platform groups cars with varying horsepower (HP) and miles per gallon (MPG) by categories (e.g., compact SUV, median sedan) and provides these groups as probabilistic cars. When a customer chooses a probabilistic car, the platform will provide any car from the corresponding group to him/her with a predetermined probability. All probabilistic cars form an uncertain dataset for customers to make multi-criteria decisions. Suppose the score of a car is defined as the weighted sum of its attributes. It is unrealistic to expect the customer to precisely determine weights of attributes. He/She can only specify rough demands like MPG is more important than HP. Then performing rskyline queries on such uncertain dataset with $\mathcal{F} = \{\omega_1\text{HP} + \omega_2\text{MPG} \mid \omega_1 \leq \omega_2\}$ can retrieve choices with high probabilities getting a car with good fuel economy to aid decision-making.

Prediction Service: With the rapid development of machine learning, prediction services are commonly provided in fields such as finance [1], disease control [2], healthcare [3], etc. For example, given historical data of stock market, algorithms like [1] can predict the price (P) and growth rate (GR) of a socket. Such prediction is usually associated with a confidence value (i.e., the probability) and all predictions form an uncertain dataset. By performing rskyline queries over this uncertain dataset with $\mathcal{F} = \{\omega_1\text{P} + \omega_2\text{GR} \mid 0.5 \times \omega_2 \leq \omega_1 \leq 2 \times \omega_2\}$, we can mine an overview of stocks with high probabilities of having advantages in both price and growth rates.

Motivated by these applications, in this paper, we investigate how to conduct rskyline queries on uncertain datasets. Similar to previous work on uncertain datasets [7], [8], [22], [25]–[28], we model uncertainty in a dataset by describing each uncertain object with a discrete probability distribution over a set of instances. Then, we adopt the possible world semantics [29] and define the rskyline probability of an object as the accumulated probabilities of all possible worlds that have one of its instances in their rskylines. Instead of identifying objects with top-k rskyline probabilities or rskyline probabilities above a given threshold, we study the problem of computing rskyline probabilities of all objects. This overcomes the difficulty of selecting an appropriate threshold and is convenient for users to retrieve results with different sizes.

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

To our knowledge, no work has been done to address this problem to date. Previous researches on uncertain datasets performed different tasks, such as skyline queries (e.g., [7], [15]–[17]), top-k queries (e.g., [18]–[22]), etc. These work only considered uncertainty in datasets but not in user preferences. However, as stated in [4], [23], determining a precise scoring function for a user is hardly realistic, which is required for top-k queries, and skyline queries lack personalization. The input $\mathcal{F}$ to our problem can contain all possible scoring functions of the user, which can be efficiently obtained by learning techniques like [24]. Meanwhile, due to the neglect of uncertainty in data, existing algorithms for answering rskyline queries can not be applied to our problem [10], [12]. An alternative approach is to convert the uncertain dataset into a certain one by representing each attribute of an object with an aggregate function like weighted sum. However, such aggregated values lose important distribution information. Objects with equal aggregated values but different instances will be treated equally. Our experiments show that this makes the aggregated result ignore objects with slightly lower aggregated values, but still appear in the rskyline result of a great number of possible worlds. We also observe that the object in the aggregated result with low rskyline probability will have many instances $\mathcal{F}$-dominated by others' instances. Therefore, this object is less attractive because it only belongs to the rskyline result in a small set of possible worlds.

The work most related to ours is researches on computing skyline probabilities of all objects on uncertain datasets [8], [25]–[27]. This problem is a special case of our problem because the dominance relation is equivalent to the $\mathcal{F}$-dominance relation when $\mathcal{F}$ contains all monotone scoring functions [10]. Although efficient algorithms were proposed for computing all skyline probabilities in [8], [25]–[27], none of them investigated the hardness of this problem. By establishing a fine-grained reduction from the orthogonal vector problem [30], we prove that no algorithm can compute rskyline probabilities of all objects within truly subquadratic time. This also proves the near optimality of algorithms proposed in [25]–[27] for computing all skyline probabilities.

In practice, one of the most common ways of specifying $\mathcal{F}$ is to impose linear constraints on weights in linear scoring functions [10]. Unfortunately, existing algorithms for computing all skyline probabilities [8], [25]–[27] do not suit for this case. The reason is that the constraints on weights makes the instance's dominance region, i.e., the region contains all instances $\mathcal{F}$-dominated by this instance, irregular. We overcome this obstacle by mapping instances into a higher dimensional data space. With this methodology, we propose a near-optimal algorithm with time complexity $O(n^{2-1/d'})$, where d' is the dimensionality of the mapped data space. Furthermore, by conducting the mapping on the fly and designing effective pruning strategies, we propose an algorithm with better expected time complexity based on the branch-and-bound paradigm.

Then, we focus on a special linear constraint called weight ratio constraint, which is also studied in [12] for rskyline queries on certain datasets. In such case, we improve the time complexity of the $\mathcal{F}$-dominance test from $O(2^{d-1})$ to $O(d)$. This newly proposed test condition implies a Turing reduction from the problem of computing rskyline probabilities of all objects to the half-space reporting problem [31]. Based on this reduction, we propose an algorithm with polynomial preprocessing time and $O(2^d mn \log n)$ query time, where m and n is the number of objects and instances, respectively. Subsequently, we introduce the multi-level strategy and the data-shifting strategy to further improve the query time complexity to $O(2^{d-1} \log n + n)$. Note that the additional linear time is only required for reporting the final results. Although this algorithm is somewhat inherently theoretical, experimental results shows that its extension for this special rskyline query on certain datasets outperforms the state-of-the-art index-based method proposed in [12]. The main contributions of this paper are summarized as follows.

- We formalize the problem of computing rskyline probabilities of all objects and prove that there is no algorithm can solve this problem in $O(n^{2-\delta})$ time for any $\delta > 0$, unless the orthogonal vectors conjecture fails.
- When $\mathcal{F}$ is a set of linear scoring functions whose weights are described by linear constraints, we propose an near-optimal algorithm with time complexity $O(n^{2-1/d'})$, where d' is the number of vertices of the preference region, and an algorithm with expected time complexity $O(mn \log n)$.
- When $\mathcal{F}$ is a set of linear scoring functions whose weights are described by weight ratio constraints, we propose an algorithm with polynomial preprocessing time and $O(2^{d-1} \log n + n)$ query time.
- We conduct extensive experiments over real and synthetic datasets to demonstrate the effectiveness of the problem studied in this paper and the efficiency and scalability of the proposed algorithms.

II. Related Work

In this section, we elaborate on two pieces of previous work that are most related to ours.

Queries on uncertain datasets. Pei et al. first studied how to conduct skyline queries on uncertain datasets [7]. They proposed two algorithms to identify objects whose skyline probabilities are higher than a threshold p. Considering inherent limitations of threshold queries, Atallah and Qi first addressed the problem of computing skyline probabilities of all objects [25]. They proposed a $\tilde{O}(n^{2-1/(d+1)})$-time algorithm by using two basic all skyline probabilities computation methods, weighted dominance counting method and grid method, to deal with frequent and infrequent objects, respectively. With a more efficient sweeping method for infrequent objects, Atallah et al. improved the time complexity to $\tilde{O}(n^{2-1/d})$ [26]. However, the utilities of these two algorithms are limited to 2D datasets because of a hidden factor exponential in the dimensionality of the dataset, which came from the high dimensional weighted dominance counting algorithm. To get rid of this, Afshani et al. calculated skyline probabilities of all instances by performing a pre-order traversal of a modified KD-tree [27]. With the well-know property of the KD-tree, it is proved that the time

complexity of their algorithm is $O(n^{2-1/d})$. More practically, Kim et al. introduced an in-memory Z-tree structure in all skyline probabilities computation to reduce the number of dominance tests, which has been experimentally demonstrated to be efficient [8]. However, it is non-trivial to revise these algorithms for computing all skyline probabilities to address the problem studied in this paper. This is because all of them rely on the fact that the dominance region of an instance is a hyper-rectangle, which no longer holds under $\mathcal{F}$-dominance.

Somehow related to what we study in this paper are those works on top-k queries on uncertain datasets [18]–[22]. Under the possible world model, top-k semantics are unclear, which give rise to different definitions, e.g., to compute the most likely top-k set, the object with high probability to rank i-th, the objects having a probability greater than a specified threshold to be included in top-k, etc. Our work differs from theirs as an exactly input weight is required in these studies, whereas we focus on finding a set of non-$\mathcal{F}$-dominated objects where $\mathcal{F}$ is a set of user-specified scoring functions. In other word, our work can be regarded as extending theirs by relaxing the input preference into a region.

Operators with restricted preference. Given a set of monotone scoring functions $\mathcal{F}$, Ciaccia and Martinenghi defined that an object t $\mathcal{F}$-dominates another object s if t scores better than s for any $f \in \mathcal{F}$ [10]. Based on $\mathcal{F}$-dominance, they introduced two restricted skyline operators, ND for retrieving the set of non-$\mathcal{F}$-dominated objects and PO for finding the set of objects that are optimal according to at least one function in $\mathcal{F}$. And they designed several linear programming based methods for these two queries, respectively. Mouratidis and Tang extended PO under top-k semantic when $\mathcal{F}$ is a convex preference polytope Ω, i.e., they studied the problem of identifying all objects that appear in the top-k result for at least one $\omega \in \Omega$ [4]. They first disqualified records $\mathcal{F}$-dominated by k or more others, and then determined the top-k-th in each partition of Ω among the remaining candidates. Liu et al. investigated a case of $\mathcal{F}$-dominance where $\mathcal{F}$ consists of $d-1$ constraints on the weight ratio of other dimensions to the user-specified reference dimension [12]. They defined *eclipse* query as retrieving the set of all non-eclipse-dominated objects and proposed a series of algorithms. These works only consider datasets without uncertainty, and we extend above dominance-based operators to uncertain datasets. Their techniques can not be applied to our problem since the introduction of uncertainty makes the problem challenging as for each instance, we now need to identify all instances that $\mathcal{F}$-dominate it.

III. Problem Definition and Hardness

In this section, we first review the restricted skyline query, then formally define the all rskyline probabilities problem and investigate its hardness.

A. Restricted Skyline

Let D be a d-dimensional dataset consisting of n objects. Each object $t \in D$ has d numeric attributes, denoted by $t = (t[1], \cdots, t[d])$. W.l.o.g., we assume the numeric domain of each attribute is normalized to $[0, 1]$ and the lower values are preferred than higher ones. Given a scoring function $f : [0,1]^d \to \mathbb{R}^+$, the value $f(t[1], \cdots, t[d])$ is called the score of object t under f, also written as $f(t)$. Function f is called monotone if for any two objects t and s, it holds that $f(t) \leq f(s)$ if $\forall 1 \leq i \leq d$, $t[i] \leq s[i]$. Let $\mathcal{F}$ be a set of monotone scoring functions, an object t $\mathcal{F}$-dominates another object $s \neq t$, denoted by $t \prec_{\mathcal{F}} s$, if $\forall f \in \mathcal{F}$, $f(t) \leq f(s)$. The restricted skyline (rskyline) of D with respect to $\mathcal{F}$ is the set of objects that are not $\mathcal{F}$-dominated by any other object, i.e., $\mathrm{RSKY}(D, \mathcal{F}) = \{t \in D \mid \nexists s \in D, s \prec_{\mathcal{F}} t\}$.

B. Restricted Skyline Probability

Let $\mathcal{D}$ denote a d-dimensional uncertain dataset including m objects. Each uncertain object $T_i \in \mathcal{D}$ is a discrete probability distribution over the d-dimensional data space. In other word, the sample space of T_i is a set of points $\{t_{i,1}, \cdots, t_{i,n_i}\}$ in the d-dimensional data space. Each point $t_{i,j}$ is called an instance of T_i and T_i has probability $p(t_{i,j})$ to occur as $t_{i,j}$. We also use T_i to denote the set of its instances $\{t_{i,1}, \cdots, t_{i,n_i}\}$ and write $t \in T_i$ to mean that t is an instance of T_i. For any object T_i, we assume $\sum_{t \in T_i} p(t_i) \leq 1$ and T_i can only take one instance at a time. Let $I = \cup_{i=1}^{m} T_i$ denote the set of all instances and $n = |I| = \sum_{i=1}^{m} n_i$. To cope with datasets of large scale, we use a spatial index R-tree to organize I.

Similar to previous work [6]–[8], [25]–[27], [32], we adopt the possible world semantics [29] and assume objects are independent of each other. The uncertain dataset $\mathcal{D}$ is interpreted as a probability distribution over a set of datasets $D \sqsubseteq \mathcal{D}$ obtained by sampling each object T_i. And the probability of observing the possible world D is

$$\Pr(D) = \prod_{t \in D} p(t) \cdot \prod_{1 \leq i \leq m, |T_i \cap D| = 0} (1 - \sum_{t \in T_i} p(t)). \quad (1)$$

Given an uncertain dataset $\mathcal{D}$ and a set of monotone scoring functions $\mathcal{F}$, the rskyline probability of an instance $t \in T_i$ is the accumulated possible world probabilities of all possible worlds that have t in their rskyline with respect to $\mathcal{F}$. Formally,

$$\Pr_{\mathrm{rsky}}(t) = \sum_{D \sqsubseteq \mathcal{D}} \times \mathbf{1}[t \in \mathrm{RSKY}(D, \mathcal{F})] \quad (2)$$

where $\mathbf{1}[\cdot]$ is the indicator function. And the rskyline probability of an object T_i, denoted by $\Pr_{\mathrm{rsky}}(T_i)$, is defined as the sum of rskyline probabilities of all its instances.

Obj	Instance		Obj	Instance	
T_1	$t_{1,1} = (3, 16)$	$p(t_{1,1}) = 1/2$	T_3	$t_{3,1} = (6, 5)$	$p(t_{3,1}) = 1/3$
	$t_{1,2} = (5, 18)$	$p(t_{1,2}) = 1/2$		$t_{3,2} = (8, 9)$	$p(t_{3,2}) = 1/3$
T_2	$t_{2,1} = (4, 13)$	$p(t_{2,1}) = 1/3$		$t_{3,3} = (12, 6)$	$p(t_{3,3}) = 1/3$
	$t_{2,2} = (5, 14)$	$p(t_{2,2}) = 1/3$	T_4	$t_{4,1} = (8.5, 15)$	$p(t_{4,1}) = 1/2$
	$t_{2,3} = (9, 12)$	$p(t_{2,3}) = 1/3$		$t_{4,2} = (13, 10)$	$p(t_{4,2}) = 1/2$

Fig. 1. An uncertain dataset $\mathcal{D}$ of 4 objects and 10 instances.

Example 1. *Consider the uncertain dataset $\mathcal{D}$ shown in Fig. 1. $D = \{t_{1,1}, t_{2,1}, t_{3,1}, t_{4,1}\}$ is a possible world of $\mathcal{D}$ and*

$\Pr(D) = p(t_{1,1}) \times p(t_{2,1}) \times p(t_{3,1}) \times p(t_{4,1}) = 1/36$. *Let* $\mathcal{F} = \{\omega[1]t[1] + \omega[2]t[2] \mid 0.5 \times \omega[2] \leq \omega[1] \leq 2 \times \omega[2]\}$, *the set of possible worlds that have* $t_{1,1}$ *in their rskyline with respect to* $\mathcal{F}$ *is* $S = \{t_{1,1}\} \times \{t_{2,2}, t_{2,3}\} \times \{t_{3,2}, t_{3,3}\} \times \{t_{4,1}, t_{4,2}\}$. *Therefore,* $\Pr_{\text{rsky}}(t_{1,1}) = \sum_{D \in S} \Pr(D) = 2/9$. *Similarly, we know* $\Pr_{\text{rsky}}(t_{1,2}) = 0$. *Hence,* $\Pr_{\text{rsky}}(T_1) = \Pr_{\text{rsky}}(t_{1,1}) + \Pr_{\text{rsky}}(t_{1,2}) = 2/9$.

The main problem studied in this paper is as follows.

Problem 1 (All RSkyline Probabilities (ARSP))**.** *Given an uncertain dataset* $\mathcal{D} = \{T_1, \cdots, T_m\}$ *and a set of monotone scoring functions* $\mathcal{F}$*, compute rskyline probabilities of all instances in* $I = \cup_{i=1}^{m} T_i$*, i.e., return the set*

$$\text{ARSP} = \{(t, \Pr_{\text{rsky}}(t)) \mid t \in I\}.$$

C. Conditional Lower Bound

We show that no algorithm can solve the ARSP problem in truly subquadratic time without preprocessing, unless the orthogonal vectors conjecture fails.

► **Orthogonal Vectors Conjecture [30].** Given two sets A, B, each of n vectors in $\{0,1\}^d$, for every $\delta > 0$, there is a $c \geq 1$ such that no $O(n^{2-\delta})$-time algorithm can determine if there is a pair $(a, b) \in A \times B$ such that $a \times b = 0$ with $d = c \log n$.

Theorem 1. *Given an uncertain dataset* $\mathcal{D}$ *and a set of monotone scoring functions* $\mathcal{F}$*, no algorithm can compute rskyline probabilities of all instances within* $O(n^{2-\delta})$ *time for any* $\delta > 0$*, unless the Orthogonal Vectors conjecture fails.*

Proof. We establish a fine-grained reduction from the orthogonal vectors problem to the ARSP problem. Given two sets A, B, each of n vectors in $\{0,1\}^d$, we construct an uncertain dataset $\mathcal{D}$ and a set $\mathcal{F}$ of monotone scoring functions as follows. First, for each vector $b \in B$, we construct an uncertain tuple T_b with a single instance b and $p(b) = 1$. Then, we construct an uncertain tuple T_A with n instances $\xi(a)$ and $p(\xi(a)) = \frac{1}{n}$ for all vectors $a \in A$, where $\xi(a)[i] = \frac{3}{2}$ if $a[i] = 0$ and $\xi(a)[i] = \frac{1}{2}$ if $a[i] = 1$ for $1 \leq i \leq d$. Finally, let $\mathcal{F}$ consists of d linear scoring functions $f_i(t) = t[i]$ for $1 \leq i \leq d$, which means instance t $\mathcal{F}$-dominates another instance s if and only if $t[i] \leq s[i]$ for $1 \leq i \leq d$. We claim that for each instance $\xi(a) \in T_A$, there exists an instance b from other uncertain tuple T_b $\mathcal{F}$-dominating $\xi(a)$ if and only if a is orthogonal to b. Suppose there is a pair $(a, b) \in A \times B$ such that $a \times b = 0$, then $a[i] = 0$ or $b[i] = 0$ for $1 \leq i \leq d$. If $a[i] = 0$, then $b[i]$ can be either 0 or 1 and $\xi(a)[i] = \frac{3}{2} > b[i]$. Or if $b[i] = 0$, then $a[i]$ can be either 0 or 1 and $\xi(a)[i] \geq \frac{1}{2} > b[i]$. That is $b \prec_{\mathcal{F}} \xi(a)$. On the other side, suppose there is a pair of instances b and $\xi(a)$ such that $b \prec_{\mathcal{F}} \xi(a)$. For each $1 \leq i \leq d$, $b[i]$ is either 0 or 1 and $\xi(a)[i]$ is either $\frac{3}{2}$ and $\frac{1}{2}$. If $b[i] = 0$, then $b[i] \cdot a[i] = 0$. Or if $b[i] = 1$, then $\xi(a)[i] = \frac{3}{2}$ since $b[i] \leq \xi(a)[i]$. So $a[i] = 0$ according to the mapping $\xi(\cdot)$. Hence $a[i] \cdot b[i] = 0$. Thus we conclude that there is a pair $(a, b) \in A \times B$ such that $a \times b = 0$ if and only if there exists an instance $\xi(a) \in T_A$ with $\Pr_{\text{rsky}}(\xi(a)) = 0$. Since $\mathcal{D}$ can be constructed in $O(nd)$ time and whether such instance exists can be determined in $O(n)$ time, any $O(n^{2-\delta})$-time algorithm for all rskyline probabilities computation for some $\delta > 0$ would yield an algorithm for Orthogonal Vectors in $O(nd + n^{2-\delta} + n) = O(n^{2-\delta'})$ time for some $\delta' > 0$ when $d = \Theta(\log n)$, which contradicts the OVC. □

IV. Algorithms for ARSP Problem with Linear Scoring Functions

The linear scoring function is one of the most commonly used scoring functions in practice [34]. Given a weight ω, the score of an object t is defined as $S_\omega(t) = \sum_{i=1}^{d} \omega[i]t[i]$. Since ordering any two objects by $S_\omega(\cdot)$ is independent from the magnitude of ω, we assume ω belongs to the unit $(d-1)$-simplex $\mathbb{S}^{d-1}$, i.e., $\forall 1 \leq i \leq d$, $\omega[i] \geq 0$, and $\sum_{i=1}^{d} \omega[i] = 1$. To serve the specific preferences of an individual user, a notable approach is to add linear constraints $A \times \omega^T \leq b$ on $\mathbb{S}^{d-1}$, where A is a $c \times d$ matrix and b is a $c \times 1$ matrix. In this section, we propose two efficient algorithms to compute ARSP in case of $\mathcal{F} = \{S_\omega(\cdot) \mid \omega \in \mathbb{S}^{d-1} \wedge A \times \omega \leq b\}$.

A. Baseline Algorithms

According to equation (2), a baseline algorithm to compute ARSP is to enumerate each possible worlds $D \sqsubseteq \mathcal{D}$, compute $\text{RSKY}(D, \mathcal{F})$, and add $\Pr(D)$ to $\Pr_{\text{rsky}}(t)$ for each $t \in \text{RSKY}(D, \mathcal{F})$. However, this brute force algorithm is infeasible due to the exponential time complexity.

Note that for any $D \sqsubseteq \mathcal{D}$, an instance $t \in T_i$ belongs to $\text{RSKY}(D, \mathcal{F})$ if and only if T_i occurs as t in D and none of other objects appears as an instance that $\mathcal{F}$-dominates t in D. Thus, $\Pr_{\text{rsky}}(t)$ can be equivalently represented as

$$\Pr_{\text{rsky}}(t) = p(t) \cdot \prod_{j=1, j \neq i}^{m} (1 - \sum_{s \in T_j, s \prec_{\mathcal{F}} t} p(s)). \quad (3)$$

The major challenge of equation (3) is to compute the product of probabilities that all other objects occur as instances that do not $\mathcal{F}$-dominate t. A straight approach is to perform $\mathcal{F}$-dominance tests between t and all instances from other objects. With the fact that the preference region $\Omega = \{\omega \in \mathbb{S}^{d-1} \mid A \times \omega \leq b\}$ is a closed convex polytope, the $\mathcal{F}$-dominance relation between two instances can be determined by comparing their scores under the set of vertices V of Ω. Here a weight ω is called a vertex of Ω if and only if it is the unique solution to a d-subset inequalities of $A \times \omega \leq b$.

Theorem 2 ($\mathcal{F}$-dominance test [10])**.** *Given a set of linear scoring functions* $\mathcal{F} = \{S_\omega(\cdot) \mid \omega \in \mathbb{S}^{d-1} \wedge A \times \omega \leq b\}$*, let* V *be the set of vertices of the preference region* $\Omega = \{\omega \in \mathbb{S}^{d-1} \mid A \times \omega \leq b\}$*, an instance* t $\mathcal{F}$*-dominates another instance* s *if and only if* $S_\omega(t) \leq S_\omega(s)$ *holds for all weights* $\omega \in V$*.*

With Theorem 2, we construct another baseline algorithm as follows. Since the preference region Ω is closed, the set of linear constraints can be transformed into a set of points using the *polar duality* [35] such that the intersection of the linear constraints is the dual of the convex hull of the points. After the transformation, the baseline invokes the quickhull algorithm proposed in [36] to compute the set of vertices V

of Ω. Then it sorts the set of instances using a scoring function $S_\omega(\cdot)$ for some $\omega \in V$. This guarantees that if an instance t precedes another instance s in the sorted list, then $s \not\prec_{\mathcal{F}} t$. After that, for each instance t, the baseline tests t against every instance of other objects preceding t in the sorted list to compute $\Pr_{\text{rsky}}(t)$ according to equation (3). Since V can be computed in $O(c^2)$ time [37], where c is the number of linear constraints, and each $\mathcal{F}$-dominance test can be performed in $O(dd')$ time, where $d' = |V|$, the time complexity of the baseline algorithm is $O(c^2 + dd'n^2)$. Although the theoretical upper bound of d' is $\Theta(c^{\lfloor d/2 \rfloor})$ [38], the actual size of V is experimentally observed to be small.

B. Tree-Traversal Algorithm

We say an object t dominates another object $s \neq t$, denoted by $t \preceq s$, if $\forall 1 \leq i \leq d, t[i] \leq s[i]$. Given an uncertain dataset $\mathcal{D}$, the skyline probability of an instance $t \in T_i$ is defined as

$$\Pr_{\text{sky}}(t) = p(t) \cdot \prod_{j=1, j \neq i}^{m} (1 - \sum_{s \in T_j, s \preceq t} p(s)).$$

The all skyline probabilities (ASP) problem aims to compute skyline probabilities of all instances [8], [25]–[27]. In case of $\mathcal{F} = \{S_\omega(\cdot) \mid \omega \in \mathbb{S}^{d-1} \wedge A \times \omega \leq b\}$, we show how to transform the ARSP problem to the ASP problem.

Given a d-dimensional uncertain dataset $\mathcal{D}$ and a set of linear scoring functions $\mathcal{F} = \{S_\omega(\cdot) \mid \omega \in \mathbb{S}^{d-1} \wedge A \times \omega \leq b\}$, let $V = \{\omega_1, \cdots, \omega_{d'}\}$ be the set of vertices of the preference region $\Omega = \{\omega \in \mathbb{S}^{d-1} \mid A \times \omega \leq b\}$ and $d' = |V|$. For each $t \in I$, $S_V(t) = (S_{\omega_1}(t), \cdots, S_{\omega_{d'}}(t))$ is a d'-dimensional point whose i-th coordinate is the score of instance t under $\omega_i \in V$. We construct a d'-dimensional uncertain dataset $\mathcal{D}'$ as follows. For each uncertain object $T_i \in \mathcal{D}$, we create an uncertain object T_i' in $\mathcal{D}'$. Then, for each instance $t \in T_i$, we compute $S_V(t)$ as an instance of T_i' and set $p(S_V(t)) = p(t)$. From Theorem 2 and the definition of dominance relationship, it is directly to know that for any two instance $t, s \in I$, $t \prec_{\mathcal{F}} s$ if and only if $S_V(t) \preceq S_V(s)$. This means, for each $t \in I$, $\Pr_{\text{rsky}}(t) = \Pr_{\text{sky}}(S_V(t))$. Therefore, after constructing $\mathcal{D}'$, we employ the procedure kd-ASP* on $\mathcal{D}'$ to compute skyline probabilities of all instances in $I' = \cup_{i=1}^{m} T_i'$.

kd-ASP* is an optimized implementation of the state-of-the-art algorithm for the ASP problem proposed in [27]. The original algorithm first constructs a kd-tree T on I', and then progressively computes skyline probabilities of all instances by performing a preorder traversal of T. We optimized it by integrating the preorder traversal into the construction of T and pruning the construction of a subtree if all instances included in the subtree have zero skyline probabilities. Although these optimizations does not improve the time complexity, they do enhance the experimental performance.

Concretely, kd-ASP* always keeps a path from the root of T to the current reached node in the main memory. And for each node N in the path, let P be the set of instances contained in N and $P_{\min}$ ($P_{\max}$) denote the minimum (maximum) corner of the minimum bounding rectangle of P, kd-ASP* maintains

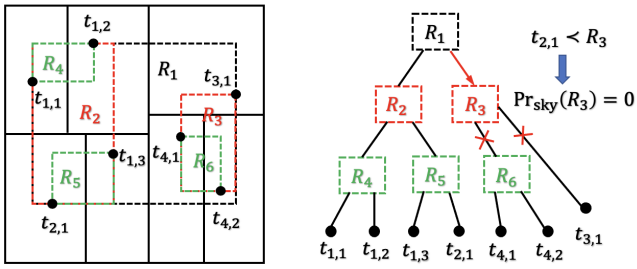
Algorithm 1: KDTree-Traversal Algorithm

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: ARSP

```
1  Compute vertices V of Ω = {ω ∈ S^{d-1} | A × ω ≤ b};
2  Construct uncertain dataset D';
3  ARSP ← ∅; χ ← 0; β ← 1;
4  foreach i ← 1 to m do σ[i] ← 0;
5  kd-ASP*(I', I');
6  return ARSP;
7  Procedure kd-ASP*(P, C)
8      C_par ← C; C ← ∅; D ← ∅;
9      foreach S_V(t) ∈ C_par do
10         if S_V(t) ⪯ P_min (say t ∈ T_i) then
11             Insert S_V(t) into D;
12             σ[i] ← σ[i] + p(t);
13             if σ[i] = 1 then
14                 χ ← χ + 1; β ← β/p(t);
15             else
16                 β ← β × (1 − σ[i])/(1 − σ[i] + p(t));
17         else if S_V(t) ⪯ P_max then
18             Insert S_V(t) into C;
19     if χ = 0 and |P| = 1 then
20         // suppose P = {S_V(t)} and t ∈ T_i
21         Insert (t, β × p(t)/(1 − σ[i])) into ARSP;
22     else if χ = 0 and |P| > 1 then
23         Partition P into P_l and P_r with selected axis;
24         kd-ASP*(P_l, C);
25         kd-ASP*(P_r, C);
26     foreach t ∈ D do
27         Undo the changes to restore σ, χ, β;
28     C ← C_par;
```

the following information, (1) a set C including instances that dominates $P_{\max}$, (2) an array $\sigma = \langle \sigma[1], \sigma[2], \cdots \sigma[m] \rangle$, where $\sigma[i] = \sum_{t \in T_i, S_V(t) \preceq P_{\min}} p(t)$ is the sum of existence probabilities over instances of T_i' that dominate $P_{\min}$, (3) a value $\beta = \prod_{1 \leq i \leq m, \sigma[i] \neq 1} (1 - \sigma[i])$, and (4) a counter $\chi = |\{i \mid \sigma[i] = 1\}|$.

At the beginning, kd-ASP* initializes $C = I'$, $\sigma[i] = 0$ for $1 \leq i \leq m$, $\beta = 1$, and $\chi = 0$ at the root node of T. Supposing the information of all nodes in the maintained path is available, kd-ASP* constructs the next arriving node N as follows. Again, let P denote the set of instances in N. For each instance $S_V(t) \in C_{par}$, where C_{par} is the set C of the parent node of N, it tests $S_V(t)$ against $P_{\min}$. If $S_V(t) \preceq P_{\min}$, say $t \in T_i$, it updates $\sigma[i]$, β, and χ as stated in lines 12-16 of Algorithm 1. Otherwise, it further tests $S_V(t)$ against $P_{\max}$ and inserts $S_V(t)$ into the set C of N if $S_V(t) \preceq P_{\max}$. When χ becomes to one, we know that $\Pr_{\text{sky}}(P_{\min}) = 0$, and so are all instances



(a) kd-tree for I'. (b) Pruning at node R_3.

Fig. 2. Running example for kd-ASP*.

in N due to the transitivity of dominance. Therefore, kd−ASP* prunes the construction of the subtree rooted at N and returns to its parent node. Otherwise, kd−ASP* keeps growing the path by partitioning set P like a kd-tree until it reaches a node including only one instance $S_{V}(t)$, then computes $\Pr_{\text{sky}}(S_{V}(t))$ based on β and σ.

Example 2. *As shown in Fig. 2, suppose all instances of an object occur with the same probability. The original algorithm keeps a whole kd-tree in the main memory but kd−ASP* only maintains a path from the root node, e.g., $R_1 \rightarrow R_2 \rightarrow R_5$. Moreover, when kd−ASP* traverses from R_1 to R_3, it updates $\sigma[2]$ to 1 and χ to 1 since $t_{2,1} \preceq R_3$. This indicates that the skyline probabilities of all instances in the subtree rooted at R_3 is zero, thus kd−ASP* prunes the construction of the subtree rooted at R_3 as shown in Fig. 2(b).*

The pseudocode of the entire algorithm is shown in Algorithm 1. As stated previously, the computation of V takes $O(c^2)$ time, where c is the number of linear constraints. For each instance $t \in I$, $S_{V}(t)$ can be computed in $O(dd')$ time, where $d' = |V|$. And given a set of n instances in d'-dimensional data space, the time complexity of kd−ASP* is $O(n^{2-1/d'})$ [27]. Therefore, the overall time complexity of Algorithm 1 is $O(c^2 + d'dn + n^{2-1/d'}) = O(n^{2-1/d'})$.

Next, we claim that Theorem 1 still holds even if we limit $\mathcal{F}$ into linear scoring functions whose weights are described by a set of linear constraints. Let $\mathcal{F}$ be the set of all linear scoring functions. Given two instances t and s, if $t \prec_{\mathcal{F}} s$, then $t[i] \leq s[i]$ for $1 \leq i \leq d$ since $\omega_i \in \Omega$ where $\omega_i[i] = 1$ and $\omega_i[j] = 0$ for all $1 \leq j \neq i \leq d$. If $t[i] \leq s[i]$ for $1 \leq i \leq d$, it is known that $t \prec_{\mathcal{F}} s$ since all linear scoring functions are monotone. Hence, we can also conclude that $t \prec_{\mathcal{F}} s$ if and only if $t[i] \leq s[i]$ for $1 \leq i \leq d$. Thus, with the same reduction established in the proof of Theorem 1, it is known that there is no subquadratic-time algorithm for the ARSP problem even if $\mathcal{F}$ is limited into linear scoring functions whose weights are described by a set of linear constraints. This proves that Algorithm 1 achieves a near-optimal time complexity.

Remark. Algorithm 1 also works when kd−ASP* adopts any other space-partitioning tree. The only detail that needs to be modified is the method to partition the data space (line 23-25 of Algorithm 1). In our experimental study, we implement a variant of Algorithm 1 based on the quadtree, which partitions the data space in all dimensions each time. It is observed that choosing an appropriate space-partitioning tree can improve the performance of Algorithm 1. For example, the quadtree-based implementation works well in low-dimensional data spaces, while the kd-tree-based implementation have better scalability for data dimensions.

C. Branch-and-Bound Algorithm

A drawback of Algorithm 1 is that it needs to map $\mathcal{D}$ to $\mathcal{D}'$ in advance, in this subsection, we show how to do the mapping on the fly so that unnecessary computations can be avoided.

Recall that if instances in I are sorted in ascending order according to their scores under a scoring function $f \in \mathcal{F}$, then an instance t will not be $\mathcal{F}$-dominated by any instance s after t. Supposing instances are processed in the sorted order, $S_{V}(t)$ is unnecessary until t is to be processed. According to this observation, we design efficient pruning strategies to tell whether an instance or a set of instances can be safely ignored during the mapping, and if so, their computations can be avoided. Unlike conducting probabilistic rskyline analysis under top-k or threshold semantics, it is easy to see that maintaining upper and lower bounds on each instance's rskyline probability as pruning criteria is helpless since our goal is to compute exact rskyline probabilities of all instances. Thus, the only pruning strategy can be utilized is that if an instance t is $\mathcal{F}$-dominated by another instance s and $\Pr_{\text{rsky}}(s)$ is zero, then $\Pr_{\text{rsky}}(t)$ is also zero due to the transitivity of $\mathcal{F}$-dominance. A straightforward method for efficiently performing this pruning strategy is to keep a rskyline of all instances processed so far whose rskyline probability is zero and compare the next instance to be processed against all instances in the rskyline beforehand. However, the maintained rskyline may suffer from huge scale on anti-correlated datasets. In the following theorems, we prove that a set P of size at most m is sufficient for pruning tests and all instances with zero rskyline probability can be safely ignored without affecting subsequent rskyline probabilities computation.

Theorem 3. *All instances with zero rskyline probability can be safely discarded.*

Proof: Let $t \in T_i$ be an instance with $\Pr_{\text{rsky}}(t) = 0$. Recall the formulation of rskyline probability in equation 3, all other instances of T_i will not be affected by t. This also holds for instances of other objects T_j that are not $\mathcal{F}$-dominated by t. Now, suppose s is an instance of $T_{j \neq i}$ and s is $\mathcal{F}$-dominated by t. Since $t \prec_{\mathcal{F}} s$ and $\Pr_{\text{rsky}}(t) = 0$, it is easy to see that there exists a set of objects $\mathcal{T} = \{T_k \mid k \neq j \wedge k \neq i\}$ such that all instances of each object $T_k \in \mathcal{T}$ $\mathcal{F}$-dominate t. Moreover, because $\mathcal{F}$-dominance is asymmetric, it is known that there exists at least one object $T_k \in \mathcal{T}$, all instances of which have non-zero rskyline probability. Therefore, according to the transitivity of $\mathcal{F}$-dominance, s is also $\mathcal{F}$-dominated by all instances of T_k and thus $\Pr_{\text{rsky}}(s) = 0$. ■

Theorem 4. *Let $V = \{\omega_1, \cdots, \omega_{d'}\}$ be the set of vertices of the preference region $\Omega = \{\omega \in \mathbb{S}^{d-1} \mid A \times \omega \leq b\}$, there is a set P such that for any instance t, $\Pr_{\text{rsky}}(t) = 0$ if and only if $S_{V}(t)$ is dominated by some instance $p \in P$ and $|P| \leq m$.*

Proof: We start with the construction of the pruning set P. For each object T_i with $\sum_{t\in T_i} p(t) = 1$, we insert an instance $p_i = (\max_{t\in T_i} S_{\omega_1}(t), \cdots, \max_{t\in T_i} S_{\omega_{d'}}(t))$ into P. Note that the above construction also requires to map all instances into the score space in advance in order to facilitate the understanding of the proof. However, in the proposed algorithm, we construct P incrementally during the computation. It is straight to verify that $|P| \leq m$ from the construction of P. Then, let t denote an instance of object T_i, we prove that $\Pr_{\text{rsky}}(t) = 0$ if and only if $S_{\mathcal{V}}(t)$ is dominated by some $p_{j\neq i} \in P$. From equation 3, it is easy to see that $\Pr_{\text{rsky}}(t) = 0$ if and only if there must exist an object $T_{j\neq i}$ such that every instance $s \in T_j$ $\mathcal{F}$-dominates t and $\sum_{s\in T_j} p(s) = 1$. That is $S_{\mathcal{V}}(s) \preceq S_{\mathcal{V}}(t)$ holds for all instances $s \in T_j$ according to Theorem 2. Moreover, since a set of instances dominates another instance if and only if the maximum corner of their minimum bounding rectangle dominates that instance, it is derived that $\Pr_{\text{rsky}}(t) = 0$ if and only if $p_j = (\max_{s\in T_j} S_{\omega_1}(s), \cdots, \max_{s\in T_j} S_{\omega_{d'}}(s)) \preceq t$. Based on the construction of P, it is known that all p_j are included in P, thus completing the proof. ■

Now, we integrate the above strategies into the proposed algorithm and the pseudocode is shown in Algorithm 2. The algorithm first computes the set of vertices V of the preference region Ω and initializes m aggregated R-trees $R_1, \cdots, R_m$, where R_i is used to incrementally index $S_{\mathcal{V}}(t)$ for all instances $t \in T_i$ with $\Pr_{\text{rsky}}(t) > 0$ that have been processed by the algorithm. After that, the algorithm traverses the index R on I in a best-first manner. Specifically, it first inserts the root of R-tree into a minimum heap H sorted according to its score under some $S_{\omega\in V}(\cdot)$, where the score of a node N is defined as $S_\omega(N_{\min})$. Then, at each time, it handles the top node N popped from H. If $S_{\mathcal{V}}(N_{\min})$ is dominated by some instance in P, then the algorithm ignores all instances in N since their rskyline probabilities are zero due to the transitivity of $\mathcal{F}$-dominance. Otherwise, if N is a leaf node, say $t \in T_i$ is contained in N, the algorithm computes $S_{\mathcal{V}}(t)$ and issues the window query with the origin and $S_{\mathcal{V}}(t)$ on each aggregated R-tree $R_{j\neq i}$ to compute $\sigma[j] = \sum_{s\in T_j, s\prec_{\mathcal{F}} t} p(s)$ and inserts $S_{\mathcal{V}}(t)$ into R_i. Then it updates p_i, which records the maximum corner of the minimum bounding rectangle of $S_{\mathcal{V}}(t)$ for all instances $t \in T_i$ with $\Pr_{\text{rsky}}(t) > 0$ that have been processed so far, and inserts p_i into P if all instances in T_i have non-zero rskyline probability. Or if N is an internal node, it inserts all non-pruned child nodes of N into H for further computation.

With the fact that Algorithm 2 only visits the nodes which contain instances t with $\Pr_{\text{rsky}}(t) > 0$ and never access the same node twice, it is easy to prove that the number of nodes accessed by Algorithm 2 is optimal to compute ARSP. And since $m - 1$ orthogonal range queries are performed on aggregated R-trees for each instance in I, the expected time complexity of Algorithm 2 is $O(nm \log n)$.

V. Sublinear-time Algorithm for Weight Ratio Constraints

In this section, we use preprocessing to accelerate ARSP computation when $\mathcal{F}$ is a set of linear scoring functions whose weights are described by weight ratio constraints. Formally, let $R = \prod_{i=1}^{d-1}[l_i, h_i]$ denote a set of user-specified ranges, weight ratio constraints R on $\mathbb{S}^{d-1}$ require $\omega[d] > 0$ and $l_i \leq \omega[i]/\omega[d] \leq h_i$ holds for every $1 \leq i < d$. Liu et al. have investigated this special $\mathcal{F}$-dominance on traditional datasets, renamed as *eclipse-dominance*, and defined the *eclipse* query to retrieve the set of all non-*eclipse-dominated* objects [12]. We refer the readers to their paper for the wide applications of eclipse query. Although we focus on uncertain datasets, our methods can also be used to design improved algorithms for eclipse query processing as shown in our experiments.

Algorithm 2: 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: ARSP

```
Compute vertices V of Ω = {ω ∈ S^{d-1} | A × ω ≤ b};
Initialize a min-heap H with respect to S_ω(·) and m
  d'-dimensional aggregated R-trees R_1, ..., R_m;
P ← ∅; ARSP ← ∅;
Insert the root of R-tree on I into H;
while H is not empty do
    Let N be the top node in H;
    if N is not pruned by P then
        if N = {t} is a leaf node (say t ∈ T_i) then
            Compute S_V(t);
            Pr_rsky(t) ← p(t);
            foreach aggregated R-tree R_{j≠i} do
                σ[j] ← perform window query with the
                  origin and S_V(t) on R_j;
                Pr_rsky(t) ← Pr_rsky(t) × (1 − σ[j]);
            Insert S_V(t) into R_i;
            Insert (t, Pr_rsky(t)) into ARSP;
            p(T_i) ← p(T_i) + p(t);
            foreach j ← 1 to |V| do
                p_i[j] ← max(p_i[j], S_V(t)[j]);
            if p(T_i) = 1 then Insert p_i into P ;
        else
            foreach child node N' of N do
                if N' is not pruned by P then
                    Insert N' into H;
return ARSP;
```

A. Reduction to Half-space Reporting Problem

Given a set of user-specified ranges $R = \prod_{i=1}^{d-1}[l_i, h_i]$ and two instances t and s, let ω^* be the optimal solution of the

following linear programming (LP) problem,

$$\begin{aligned} \text{minimize} \quad & h(\omega) = \sum_{i=1}^{d} (s[i] - t[i]) \times \omega[i] \\ \text{subject to} \quad & l_i \le \omega[i]/\omega[d] \le h_i \qquad 1 \le i < d \qquad (4) \\ & \omega[d] > 0, \quad \sum_{i=1}^{d} \omega[i] = 1. \end{aligned}$$

Under weight ratio constraints R, the $\mathcal{F}$-dominance test condition stated in Theorem 2 can be equivalently represented as determining whether $h(\omega^*) \ge 0$. The crucial observation is that the sign of $h(\omega^*)$ can be determined more efficiently without solving problem (4). To be specific, since $\omega[d] > 0$, changing the object function $h(\omega) = \sum_{i=1}^{d}(t[i] - s[i]) \times \omega[i]$ into $h'(r) = \sum_{i=1}^{d-1}(t[i] - st[i]) \times r[i] + (s[d] - t[d])$, where $r[i] = \omega[i]/\omega[d]$, guarantees that $h(\omega^*) \ge 0$ iff $h'(r*) \ge 0$. Here r^* is the optimal solution of the problem (4) with $h'(r)$ as the objective function. Since each $r[i] = \omega[i]/\omega[d]$ can be choose independently from the corresponding interval $[l_i, h_i]$, r^* can be directly determined in $O(d)$ time. Based on this, we can perform $\mathcal{F}$-dominance test more efficiently.

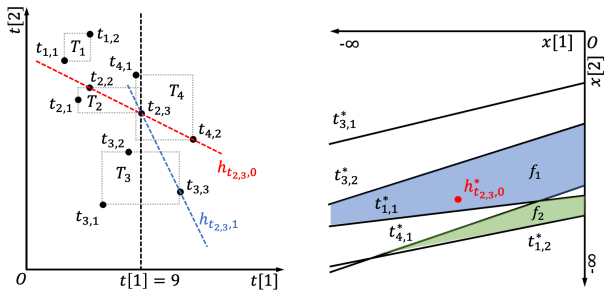
Theorem 5 (Efficient $\mathcal{F}$-dominance test)**.** *Let $\mathcal{F}$ be a set of linear scoring functions whose weights are described by weight ratio constraints $R = \prod_{i=1}^{d-1}[l_i, h_i]$, an instance t $\mathcal{F}$-dominates another instance s if and only if $t[d] - s[d] \le \sum_{i=1}^{d-1}(\mathbf{1}[s[i] > t[i]] \times l_i + (1 - \mathbf{1}[s[i] > t[i]]) \times h_i) \times (s[i] - t[i])$, where $\mathbf{1}[\cdot]$ is the indicator function.*

According to Theorem 5, we present a reduction of finding all instances in I that $\mathcal{F}$-dominate instance t to a series of 2^{d-1} half-space reporting problem [31], which aims to preprocess a set of points in $\mathbb{R}^d$ into a data structure so that all points lying below or on a query hyperplane can be reported quickly. We partition the data space $\mathbb{R}^d$ into 2^{d-1} regions using $d-1$ hyperplanes $x[i] = t[i]$ $(1 \le i < d)$. Then, each resulted region can be identified by a $(d-1)$-bit code such that the i-th bit is 0 if the i-th attributes of instances in this region are less than $t[i]$, and 1 otherwise. We refer the region whose identifier is k in decimal as region k. Suppose $t \in T_i$, let $I_{t,k}$ denote the set of instances of other uncertain objects contained in region k. As an example, $I_{t,0} = \{s \in I \setminus T_i \mid \forall 1 \le i < d. s[i] < t[i]\}$. It is easy to verify for each $0 \le k < 2^{d-1}$, when performing $\mathcal{F}$-dominance test between instances in $I_{t,k}$ and t, the results of $d-1$ indicator functions in Theorem 5 are identical for all instances in $I_{t,k}$. Geometrically, all instances in $I_{t,k}$ that $\mathcal{F}$-dominate t must lie below or on the following hyperplane,

$$h_{t,k} : x[d] = \sum_{i=1}^{d-1}((1 - |k|_2[i]) \times l_i + |k|_2[i] \times h_i) \times (t[i] - x[i]) + t[d], \qquad (5)$$

where $|k|_2[i]$ is i-th bit of the binary of number k.

Example 3. *The uncertain dataset used in Example 1 is plotted in Fig. 3(a). Consider instance $t_{2,3}$. Data space $\mathbb{R}^2$ is partitioned with the line $t[1] = t_{2,3}[1] = 9$. Region 0 contains the set of instances $I_{t_{2,3},0} = \{s \in I \setminus T_2 \mid s[1] \le t_{2,3}[1]\} = \{t_{1,1}, t_{1,2}, t_{3,1}, t_{3,2}, t_{4,1}\}$, and region 1 contains the set of instances $I_{t_{2,3},1} = \{t_{3,3}, t_{4,2}\}$. Given weight ratio constraints $R = [0.5, 2]$, according to equation (5), the hyperplane $h_{t_{2,3},0}$ in region 0 is $t[2] = -0.5t[1] + 16.5$ and the hyperplane $h_{t_{2,3},1}$ in region 1 is $t[2] = -2t[1] + 30$. Since $t_{3,1}$ and $t_{3,2}$ ly below or on $h_{t_{2,3},0}$, we know $t_{3,1} \prec_{\mathcal{F}} t_{2,3}$ and $t_{3,2} \prec_{\mathcal{F}} t_{2,3}$. And since $t_{3,3}$ lies on $h_{t_{2,3},1}$, we know $t_{3,3} \prec_{\mathcal{F}} t_{2,3}$.*



(a) Partition data space $\mathbb{R}^2$ with $t_{2,3}$, and two hyperplanes $h_{t_{2,3},0}, h_{t_{2,3},1}$ of $t_{2,3}$.

(b) Dual hyperplanes in $I^*_{t_{2,3},0}$, two faces f_1, f_2 of $\mathcal{A}(I^*_{t_{2,3},0})$, and dual point $h^*_{t_{2,3},0}$.

Fig. 3. An illustration of the reduction to half-space reporting problem and performing point location queries in dual space.

The half-space reporting problem can be efficiently solved using the well-known *point-hyperplane duality* [39]. The duality maps a point $p = (p[1], \cdots, p[d]) \in \mathbb{R}^d$ into the hyperplane $p^* : x[d] = p[1]x[1] + \cdots + p[d-1]x[d-1] - p[d]$, and a hyperplane $h : x[d] = \alpha[1]x[1] + \cdots + \alpha[d-1]x[d-1] - \alpha[d]$ into the point $h^* = (\alpha[1] \cdots, \alpha[d])$. It is proved that if p lies above (resp., below, on) h, then h^* lies above (resp., below, on) p^*. Thus, the dual version of the half-space reporting problem becomes that given a set of n hyperplanes in $\mathbb{R}^d$ and a query point q, report all hyperplanes lying above or through q. Let H be the set of n hyperplanes in $\mathbb{R}^d$, the *arrangement* of H, denoted by $\mathcal{A}(H)$, is a subdivision of $\mathbb{R}^d$ into *faces* of dimension k for $0 \le k \le d$. Each face in $\mathcal{A}(H)$ is a maximal connected region of $\mathbb{R}^d$ that lies in the same subset of H. For a query point q, let $\lambda(q, H)$ denote the set of hyperplanes in H lying above or through q. It is easy to verify that all points p lying on the same face f of $\mathcal{A}(H)$ have the same $\lambda(p, H)$, denoted by $\lambda(f, H)$. Thus, with a precomputation of $\lambda(f, H)$ for each face f of $\mathcal{A}(H)$ and the following structure for point location in $\mathcal{A}(H)$, $\lambda(q, H)$ can be found in logarithmic time.

Theorem 6 (Structure for Point Location [40])**.** *Given a set H of n hyperplanes in $\mathbb{R}^d$ and a query point q, there is a data structure of size $O(n^{d+\varepsilon})$ which can be constructed in $O(n^{d+\varepsilon})$ expected time for any $\varepsilon > 0$, so that the face of $\mathcal{A}(H)$ containing q can be located in $O(\log n)$ time.*

Accordingly, after building the point location structure for the set of dual hyperplanes of each $I_{t,k}$, by locating the dual point of $h_{t,k}$, we can find all instances in $I_{t,k}$ that $\mathcal{F}$-dominate t efficiently. However, according to equation (3), in order to compute $\Pr_{\text{rsky}}(t)$, we need to further calculate the cumulative probability of instances from the same uncertain object. In what follows, we propose an efficient algorithm to compute

ARSP by modifying the above algorithm.

In the preprocessing stage, for each instance $t \in I$, say $t \in T_i$, the algorithm partitions $I \setminus T_i$ into 2^{d-1} sets $I_{t,k} = \{s \in I \setminus T_i \mid s$ in region k derived by partitioning $[0,1]^d$ with $t\}$ $(0 \leq k < 2^{d-1})$. Then, for each set $I_{t,k}$, it computes the set of dual hyperplanes $I^*_{t,k} = \{s^* \mid s \in I_{t,k}\}$ and builds the point location structure on $I^*_{t,k}$. Finally, it constructs $\mathcal{A}(I^*_{t,k})$ and records an array $\sigma_f = \langle \sigma_f[1], \cdots, \sigma_f[m] \rangle$ for each face f of $\mathcal{A}(I^*_{t,k})$, where $\sigma_f[j] = \sum_{s^* \in \lambda(f, I^*_{t,k}) \wedge s \in T_j} p(s)$, i.e., the sum of probabilities over all instances of object T_j lying below or on the hyperplane p^*, where p^* is the dual hyperplane of some point p lying in face f.

In the query processing stage, given weight ratio constraints $R = \prod_{i=1}^{d-1}[l_i, h_i]$, the algorithm processes each instance t as follows. It first initializes $\Pr_{\text{rsky}}(t) = p(t)$ and $\sigma[i] = 0$ for $1 \leq i \leq m$, where $\sigma[i]$ is for recording the sum of existence probabilities of instances from object T_i that $\mathcal{F}$-dominate t found so far. Then, for each $0 \leq k < 2^{d-1}$, it compute the dual point $h^*_{t,k}$ of the hyperplane $h_{t,k}$ defined in equation (5), and performs point location query $h^*_{t,k}$ on the structure built on $I^*_{t,k}$. Let f be the face returned by the point location query $h^*_{t,k}$, for $1 \leq j \leq m$, it updates $\Pr_{\text{rsky}}(t)$ to $\Pr_{\text{rsky}}(t) \times (1 - \sigma[j] - \sigma_f[j])/(1 - \sigma[j])$ and adds $\sigma_f[j]$ to $\sigma[j]$. After all queries, it returns $\Pr_{\text{rsky}}(t)$ as the final rskyline probability of t. Since each point location query can be performed in $O(\log n)$ time and the update of $\Pr_{\text{rsky}}(t)$ requires $O(m)$ time for each σ_f, the time complexity of this algorithm is $O(2^d mn \log n)$.

Example 4. *Continue with Example 3. For instance $t_{2,3}$, in the preprocessing stage, the algorithm will compute dual hyperplanes of $I_{t_{2,3},0}$ and $I_{t_{2,3},1}$, build point location structures on $I^*_{t_{2,3},0}$ and $I^*_{t_{2,3},1}$, and record σ_f for each face f of $\mathcal{A}(I^*_{t_{2,3},0})$ and $\mathcal{A}(I^*_{t_{2,3},1})$. As an example, hyperplanes in $I^*_{t_{2,3},0}$ are plotted in Fig. 3(b). For face f_1, it records $\sigma_{f_1}[1] = \sigma_{f_1}[4] = 0$, $\sigma_{f_1}[3] = p(t_{3,1}) + p(t_{3,2}) = 2/3$ since $t^*_{3,1}$ and $t^*_{3,2}$ lie above or through every point in f_1 and for face f_2 it records $\sigma_{f_2}[1] = p(t_{1,1}) = 1/2$, $\sigma_{f_2}[3] = p(t_{3,1}) + p(t_{3,2}) = 2/3$, $\sigma_{f_2}[4] = p(t_{4,1}) = 1/2$ since $t^*_{1,1}, t^*_{3,1}, t^*_{3,2}$, and $t^*_{4,1}$ lie above or through every point in f_1.*

Then, given weight ratio constraints $R = [0.5, 2]$, to compute $\Pr_{\text{rsky}}(t_{2,3})$, the algorithm first initializes $\Pr_{\text{rsky}}(t_{2,3}) = p(t_{2,3}) = 1/3$ and $\sigma[i] = 0$ $(1 \leq i \leq 4)$, then performs point location query $h^*_{t_{2,3},0}$ and $h^*_{t_{2,3},1}$ on $I^*_{t_{2,3},0}$ and $I^*_{t_{2,3},1}$ respectively to update $\Pr_{\text{rsky}}(t_{2,3})$ and $\sigma[i]$ $(1 \leq i \leq 4)$. By locating $h^*_{t_{2,3},0} = (-0.5, -16.5)$, which is the dual point of $h_{t_{2,3},0} : t[2] = -0.5t[1] + 16.5$, face f_1 is returned. Since only $\sigma_{f_1}[3] \neq 0$, the algorithm updates $\Pr_{\text{rsky}}(t_{2,3})$ to $\Pr_{\text{rsky}}(t_{2,3}) * (1 - \sigma[3] - \sigma_{f_1}[3])/(1 - \sigma[3]) = 1/9$ and updates $\sigma[3]$ to $\sigma[3] + \sigma_{f_1}[3] = 2/3$.

B. Sublinear-time Algorithm

To achieve a sublinear query time, the following two bottlenecks of the above algorithm should be addressed. First, for each instance, 2^{d-1} arrays are accessed and it seems unrealistic to merge them efficiently based on equation (3). Second, since instances are sequentially processed, the query time can not be less than n. In subsequent, we introduce two strategies to overcome these two inefficiencies.

Multi-level strategy. The reason why the above algorithm has to access 2^{d-1} arrays for each instance t is that the query point $h^*_{t,k}$ is different for each $I^*_{t,k}$. Hence, it needs to perform 2^{d-1} different point location queries to retrieve σ_f from each $I_{t,k}$. Different from general linear constraints, the reduction ensures that given weight ratio constraints $R = \prod_{i=1}^{d-1}[l_i, h_i]$, the number of point location queries performed for each instance is always 2^{d-1}. In this case, we show how to resolve this issue with the help of *multi-level strategy* [41].

A 2^{d-1}-level structure on the set of dual hyperplanes $I^* = \{t^* \mid t \in I\}$ is a recursively defined point location structure. To be specific, an one-level structure is a point location tree built on I^*. And each face f of $\mathcal{A}(I^*)$ records the following information: (1) an array $\sigma_f = \langle \sigma_f[j] \mid 1 \leq j \leq m \rangle$, where $\sigma_f[j] = \sum_{s^* \in \lambda(f, I^*) \wedge s \in T_j} p(s)$, i.e., the sum of probabilities over all instances of object T_j lying below or on the hyperplane p^*, where p^* is the dual hyperplane of some point p lying in face f, (2) a product $\beta_f = \prod_{j=1, \sigma_f[j] \neq 1}^{m}(1 - \sigma_f[j])$, and (3) a count $\chi_f = |\{j \mid \sigma_f[j] = 1\}|$ are also recorded for each face $f \in \mathcal{A}(I^*)$. A k-level structure is an one-level structure built on I^* and each face f of $\mathcal{A}(I^*)$ additionally contains an associated $(k-1)$-level structure built on $\lambda(f, I^*)$.

After constructing the 2^{d-1}-level structure on I^*, the algorithm processes weight ratio constraints $R = \prod_{i=1}^{d-1}[l_i, h_i]$ as follows. For each instance t, it initializes hyperplanes set H as I^* and integer k as zero. While $k < 2^{d-1}$, it generate the dual point $h^*_{t,k}$ according to equation (5) and performs point location query $h^*_{t,k}$ on structure built on H. Then, let f be the returned face, it updates H as $\lambda(f, H)$ and k as $k + 1$. Note that query $h^*_{t,k}$ helps to find all instances lying below or on $h_{t,k}$ in the result of the first $k - 1$ queries. Let f be the last face returned. According to the information recorded for f, $\Pr_{\text{rsky}}(t)$ is calculated as follows. If $\chi_f = 0$, then $\Pr_{\text{rsky}}(t) = \beta_f \cdot p(t)/(1 - \sigma_f[i])$, or if $\chi_f = 1 \wedge \sigma_f[i] = 1$, $\Pr_{\text{rsky}}(t) = \beta_f \times p(t)$, otherwise, $\Pr_{\text{rsky}}(t) = 0$. Since for each instance $t \in I$, $\Pr_{\text{rsky}}(t)$ can be computed in constant time after performing 2^{d-1} point location queries, the total time complexity of the multi-level structure based algorithm for ARSP computation is $O(2^{d-1} n \log n)$ time.

Shift strategy. The major obstacle to the second bottleneck is that the 2^{d-1} point location queries $h^*_{t,k}$ $(0 \leq k < 2^{d-1})$ are different for each instance t. Geometrically speaking, $R = \prod_{i=1}^{d-1}[l_i, h_i]$ is a $(d-1)$-dimensional hyper-rectangle. Let $V = \{v = (v[1], \cdots, v[d-1]) \mid \forall 1 \leq i < d, v[i] \in \{l_i, h_i\}\}$ be the set of R's vertices. For $0 \leq k < 2^{d-1}$, we call a vertex $v \in V$ the k-vertex of R if there are k vertices before v in the lexicographical order of vertices in V. For example, $(l_1, \cdots, l_{d-1})$ is the 0-vertex of R and $(h_1, \cdots, h_{d-1})$ is the $(2^{d-1} - 1)$-vertex of R. Let v_k denote the k-vertex of R. According to equation (5), $h^*_{t,k} = (-v_k[1], \cdots, -v_k[d-1], -(\sum_{i=1}^{d-1} v_k[i]t[i] + t[d]))$. For any two instances t and s, each pair $h*_{t,k}$ and $h^*_{s,k}$ differs only in the last dimension. In what follows, we introduce the *shift strategy* to unify the pro-

cedures of performing point location queries for all instances by making their $h^*_{t,k}[d]$ the same for each $0 \le k < 2^{d-1}$.

Specifically, for each instance $t \in I$, say $t \in T_i$, the algorithm first creates a shifted dataset I_t by treating t as the origin, i.e., $I_t = \{s - t \mid s \in I \setminus T_i\}$. Then, it merges all sets I_t into a key-value pair set $\mathcal{I} = \{(s, \langle t \mid s \in I_t \rangle) \mid s \in \bigcup_{t \in I} I_t\}$. Finally, it construct a 2^{d-1}-level structure on the set of dual hyperplanes $\mathcal{I}^* = \{s^* \mid (s, -) \in \mathcal{I}\}$ as stated above, except that the information recorded in the one-level structure for each face f of $\mathcal{A}(\mathcal{I}^*)$ is redefined as $\Pr_f = \langle \Pr_f[t] \mid t \in I \rangle$, where $\Pr_f[t] = \prod_{j=1, j \neq i}^{m} (1 - \sum_{s^* \in \lambda(f, \mathcal{I}^*) \wedge s + t \in T_j} p(s))$.

Given weight ratio constraints $R = \prod_{i=1}^{d-1}[l_i, h_i]$, the algorithm generates 2^{d-1} point location queries $h^*_k = (v_k[1], \cdots, v_k[d-1], 0)$ $(0 \le k < 2^{d-1})$ and executes them on the 2^{d-1}-level structure built on $\mathcal{I}^*$. Let f be the last face returned, $\Pr_{\text{rsky}}(t)$ of each instance $t \in I$ is computed as $p(t) \times \Pr_f(t)$. Since the algorithm now performs a total of 2^{d-1} point location queries for all instances, the query time is at most $O(2^{d-1} \log n + n)$, where the additional linear time is only required for reporting the final results.

VI. Experiments

In this section, we present the experimental study of the algorithms proposed for the ARSP problem.

A. Experimental Setting

Datasets. We use both real and synthetic datasets for experiments. The real data includes three datasets. IIP [5] contains 19,668 sighting records of icebergs with 2 attributes: melting percentage and drifting days. Each record in IIP has a confidence level according to the source of sighting, including R/V (radar and visual), VIS (visual only), RAD (radar only). We treat each record as an uncertain object with one instance and convert these three confidence levels to probabilities 0.8, 0.7, and 0.6 respectively. CAR [6] contains 184,810 cars with 4 attributes: price, power, mileage, registration year. To convert CAR into an uncertain dataset, we organize cars with the same model into an uncertain object T and for each $t \in T$, we set $p(t) = 1/|T|$, i.e., when a customer wants to rent a specific model of car, any car of that model in the dataset will be offered with equal probability. NBA [8] includes 354,698 game records of 1,878 players with 8 metrics: points, assists, steals, blocks, turnovers, rebounds, minutes, field goals made. We treat each player as an object T and each record of the player as an instance t of T with $p(t) = 1/|T|$.

The synthetic datasets are generated with the same procedure described in [7], [8], [25], [32]. Let m be the number of uncertain objects. For $1 \le i \le m$, we first generate center c_i of object T_i in $[0, 1]^d$ following independent (IND), anti-correlated (ANTI), or correlated (CORR) distribution [42]. Then, we generate a hyper-rectangle R_i centered at c_i. And all instances of T_i will appear in R_i. The edge length of R_i follows a normal distribution in range $[0, l]$ with expectation $l/2$ and standard deviation $l/8$. And the number of T_i's instances follows a uniform distribution over interval $[1, cnt]$. We generate n_i instances uniformly within R_i and assign each instance the existence probability $1/n_i$. Finally, we remove one instance from the first $\phi \times m$ objects so that for any $1 \le j \le \phi \times m$, $\sum_{t \in T_j} p(t) < 1$. Therefore, the expected number of instances in a synthetic dataset is $(cnt/2 - \phi) \times m$.

Constraints. Our experiments consider two methods to generate linear constraints on weights. WR specifies weak rankings on weight attributes [9]. Given the number of constraints c, it requires $\omega[i] \ge \omega[i+1]$ for every $1 \le i \le c$. IM generates constraints in an interactive manner [24]. Specifically, it first chooses a weight ω^* randomly in $\mathbb{S}^{d-1}$. Then, for each $1 \le i \le c$, it generates two objects t_i, s_i uniformly in $[0, 1]^d$, divide $\mathbb{S}^{d-1}$ into two subspaces with $\sum_{j=1}^{d}(t_i[j] - s_i[j]) \times \omega[j] = 0$, and selects the one containing ω^* as the i-th input constraint. The main difference between these two methods is that the preference region generated by WR always has d vertices, while the number of vertices of the preference region generated by IM usually increases with c.

Algorithms. We implement the following algorithms in C++ and the source code is available at [43].

- ENUM: the first baseline algorithm in Section IV-A.
- LOOP: the second baseline algorithm in Section IV-A.
- KDTT: the kdtree-traversal algorithm in Section IV-B.
- KDTT+: the kdtree-traversal algorithm incorporating pre-order traversal into tree construction in Section IV-B.
- QDTT+: the quadtree-traversal algorithm incorporating pre-order traversal into tree construction in Section IV-B.
- B&B: the branch-and-bound algorithm in Section IV-C.
- DUAL (-M/S): the dual-based algorithm in Section V, where -M is for multi-level strategy, -S is for shift strategy.

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.

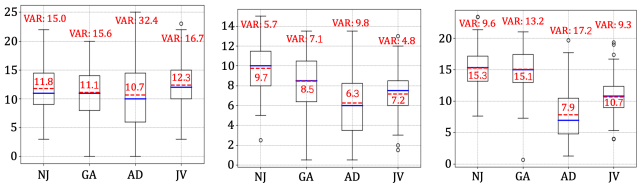
B. Effectiveness of ARSP

We verify the effectiveness of ARSP on the NBA dataset. To facilitate analysis, we extract game records in 2021 from NBA and consider 3 attributes for each player: rebound, assist, and points. We still treat each player as an object T and each record of the player as an instance t of T with $p(t) = 1/|T|$. We set $\mathcal{F} = \{\omega[1]\text{Rebound} + \omega[2]\text{Assist} + \omega[3]\text{Point} \mid \omega[1] \ge \omega[2] \ge \omega[3]\}$. Theorem 2 claims that $t \prec_{\mathcal{F}} s$ iff $S_\omega(t) \le S_\omega(s)$ holds for all weights $\omega \in V$, where $V = \{\omega_1 = (1, 0, 0), \omega_2 = (1/2, 1/2, 0), \omega_3 = (1/3, 1/3, 1/3)\}$. Table I reports the top-14 players in rskyline probability ranking along with their rskyline probabilities. As a comparison, we also conduct the rskyline query on the aggregated dataset, which is obtained by computing the average statistics for each player. We call the result aggregated rskyline for short hereafter and mark players in the aggregated rskyline with a "*" sign in Table I.

We first observe that the rskyline probability can reflect the difference between two incomparable players in the aggregated dataset under $\mathcal{F}$. See Fig. 4 for scores of Nikola Jokic (NJ) and Jonas Valanciunas (JV) under weights in V. Nikola Jokic not only has a good average performance so that he is in the aggregated rskyline, but also performs best in some games so that he has a high rskyline probability. As for Jonas Valanciunas,

TABLE I
TOP-14 PLAYERS IN RSKYLINE PROBABILITY RANKING.

Player	$\Pr_{\text{rsky}}$	Player	$\Pr_{\text{rsky}}$
* Russell Westbrook	0.349	* Rudy Gobert	0.142
* Nikola Jokic	0.331	* Clint Capela	0.134
Giannis Antetokounmpo	0.292	Nikola Vucevic	0.126
James Harden	0.213	Andre Drummond	0.109
Joel Embiid	0.186	Julius Randles	0.109
Luka Doncic	0.168	Kevin Durant	0.101
* Domantas Sabonis	0.162	* Jonas Valanciunas	0.095



(a) $S_{\omega_1}(\cdot)$ (b) $S_{\omega_2}(\cdot)$ (c) $S_{\omega_3}(\cdot)$

Fig. 4. Boxplots of players' scores under $\omega_1 = (1, 0, 0)$, $\omega_2 = (1/2, 1/2, 0)$, and $\omega_3 = (1/3, 1/3, 1/3)$, where average is marked with red dotted lines.

his average performance under ω_1 is great, making him belong to the aggregated rskyline. But his large performance variance under ω_1 and relatively poor performance under ω_2 and ω_3 suggests that many of his records are $\mathcal{F}$-dominated by other players' records. This results in his rskyline probability being pretty low. Therefore, compared to aggregated rskyline players with high rskyline probabilities, who consistently perform well, those with low rskyline probabilities are more likely to have many records being $\mathcal{F}$-dominated by other players' records, which may be less attractive.

Second, we find that players not in the aggregated rskyline but have high rskyline probabilities is also appealing. For example, Giannis Antetokounmpo is $\mathcal{F}$-dominated by Nikola Jokic in the aggregated dataset, but his rskyline probability is higher than another four aggregated rskyline players. Compared to NJ, his scores (GA) have slightly lower averages and higher variances. In other words, he has some records, like Nikola Jokic's, which $\mathcal{F}$-dominates most of other players' records and he also has some records that are $\mathcal{F}$-dominated by many of other players' records. Besides, the large performance variance also explains why Andre Drummond (AD) is $\mathcal{F}$-dominated by Jonas Valanciunas (JV) but has a higher rskyline probability. This suggests that looking for players with high rskyline probabilities can find excellent players with slightly lower averages but larger variances in performance.

Finally, a set of players with specified size can be retrieved by performing top-k queries on ARSP, while the size of the aggregated rskyline is uncontrollable. From these observations, we conclude that ARSP provides a more comprehensive view on uncertain datasets than the aggregated rskyline.

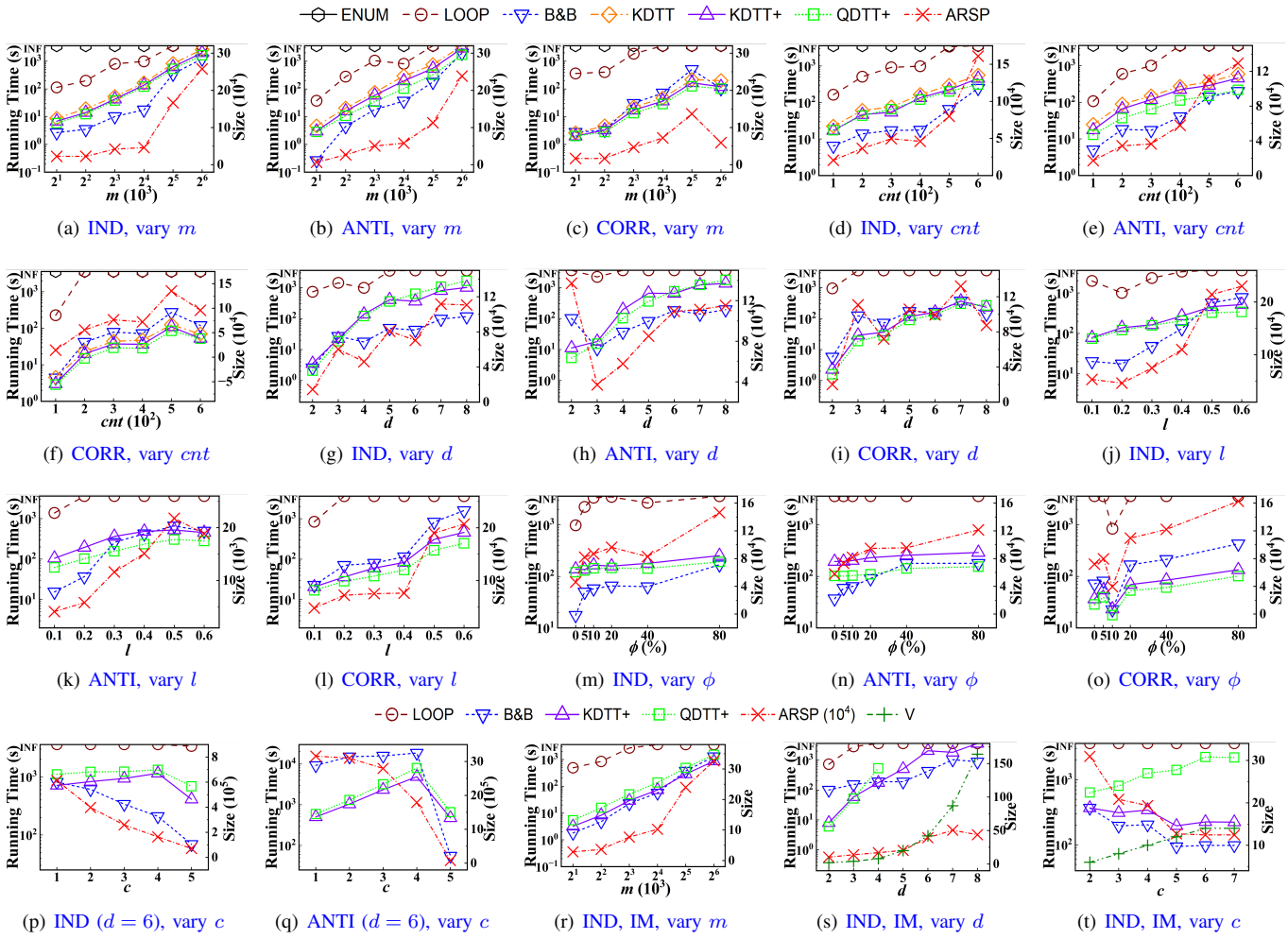
C. Experimental Results under Linear Constraints.

Fig. 5 and 6 show the running time of different algorithms and the size of ARSP on real and synthetic datasets. The size of ARSP is the number of instances with none-zero rskyline probabilities. Following [7], [8], the default values for object cardinality m, instance count cnt, data dimensionality d, region length l, and percentage ϕ of objects with $\sum_{t \in T} p(t) < 1$ of synthetic datasets are set as $m = 16$K, $cnt = 400$, $d = 4$, $l = 0.2$, and $\phi = 0$. Unless otherwise stated, WR is used to generate $c = d - 1$ input linear constraints for $\mathcal{F}$. All datasets are index by R-trees in main memory. Since the construction time of the R-tree is a one-time cost for all subsequent queries, it is not included in the query time. And the query time limit (INF) is set as 3,600 seconds.

Fig. 5 (a)-(c) present the results on synthetic datasets with m varying from 2K to 64K. Based on the generation process, the number of instances n increases as m grows. Thus, the running time of all algorithms and the size of ARSP increase. Due to the exponential time complexity, ENUM never finishes within the limited time. All proposed algorithms outperform LOOP by around an order of magnitude because LOOP always performs a large number of $\mathcal{F}$-dominance tests and does not include any pruning strategy. B&B runs fastest on IND and ANTI with the help of the incremental mapping and pruning strategies. As m grows, the gap narrows because the more objects, the more aggregated R-trees are queried per instance. KDTT+ and QDTT+ are more effective on CORR because pruning is triggered earlier by objects near the origin during space partitioning, e.g., when $m = 2$K, QDTT+ prunes 13 child nodes of the root node on CORR, compared with 9 on IND and 5 on ANTI. Although with similar strategies, QDTT+ performs better than KDTT+. The reason is that space is recursively divided into 2^d regions in QDTT+, which results in a smaller MBR and thus a greater possibility of being pruned. Results also demonstrate our optimization techniques significantly improve the experimental performance of KDTT. As shown in Fig. 5 (d)-(f), the relative performance of all algorithms remains basically unchanged with respect to cnt. And the size of ARSP also increases as cnt grows since the larger cnt, the more instances in I and the less likelihood of an instance being $\mathcal{F}$-dominated by all instance of an object.

Having established ENUM is inefficient to compute ARSP, henceforth it is excluded from the following experiments. The curve of KDTT is also omitted as it is always outperformed by KDTT+. Fig. 5 (g)-(i) plot the results on synthetic datasets with varying dimensionality d. With the increase of d, the cost of $\mathcal{F}$-dominance test increases. Thus, the running time of all algorithms increases. QDTT+ and KDTT+ are more efficient than B&B on low-dimensional datasets, but their scalability is relatively poor. This is because when d grows, the dataset becomes sparser, causing the subtrees pruned during the pre-order traversal get closer to leaf nodes in KDTT+ and QDTT+. Moreover, the exponential growth in the number of child nodes of QDTT+ also causes its inefficiency on high-dimensional datasets. When the dataset becomes sparser, an instance is more likely not to be $\mathcal{F}$-dominated by others. Therefore, the size of ARSP increases with higher dimensionality.

Fig. 5 (j)-(l) show the effect of l by varying l from 0.1 to 0.6. As l increases, the number of instances $\mathcal{F}$-dominated by all instances of an object decreases. Thus, the running time of all algorithms and the size of ARSP increases. Compared



(a) IND, vary m (b) ANTI, vary m (c) CORR, vary m (d) IND, vary cnt (e) ANTI, vary cnt

(f) CORR, vary cnt (g) IND, vary d (h) ANTI, vary d (i) CORR, vary d (j) IND, vary l

(k) ANTI, vary l (l) CORR, vary l (m) IND, vary ϕ (n) ANTI, vary ϕ (o) CORR, vary ϕ

(p) IND ($d = 6$), vary c (q) ANTI ($d = 6$), vary c (r) IND, IM, vary m (s) IND, IM, vary d (t) IND, IM, vary c

Fig. 5. Running time of different algorithms and the size of ARSP on synthetic datasets.

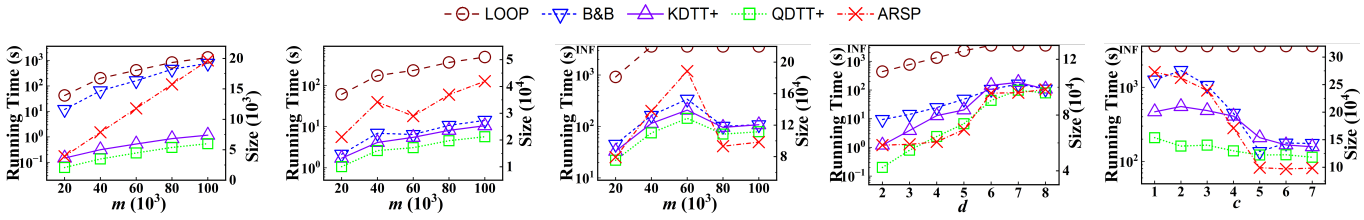
to others, B&B is more sensitive to l since it determines not only the number of instances to be processed but also the time consumed in querying aggregated R-trees.

Fig. 5 (m)-(o) show the runtime of different algorithms and the size of ARSP on synthetic datasets with different ϕ. According to equation 3, the more objects T with $\sum_{t\in T} p(t) < 1$, the less instances with zero rskyline probabilities. Hence, the running time and the size of ARSP both increase as ϕ increases. Similar to l, ϕ also affects B&B greatly since the larger ϕ, the fewer instances are added to the pruning set P.

Fig. 5 (p)-(q) plot the effect of c on IND and ANTI ($d = 6$). Results on CORR are omitted, in which the running time of all algorithms and the size of ARSP decrease as c grows. The reason is that the preference region narrows with the growth of c, which enhances the $\mathcal{F}$-dominance ability of each instance. Therefore, more instances are pruned during the computation. Whereas, this also results in the need to perform more $\mathcal{F}$-dominance tests to compute rskyline probabilities of unpruned instances. The trends of the running time on IND and ANTI reflect the compromise of these two factors. B&B perform inconsistently as its pruning strategy is more effective on IND.

Fig. 5 (r)-(t) show the results under linear constraints generated by IM. Running time of all algorithms and the size of ARSP show similar trends to WR under all parameters except c. As stated above, the $\mathcal{F}$-dominance ability of each instance improves with the growth of c. Thus, as shown in Fig. 5(t), the running time of all algorithms decreases, except QDTT+. This is because the number of vertices of the preference region generated by IM increases as c grows (see the curve of V), thus leading to the dimensional disaster of the quadtree. This also accounts for the failure of QDTT+ when $d \geq 5$ in Fig. 5(s).

Experimental results on real datasets confirm the above observations. Fig. 6 (a) shows the results on IIP with varying m. As introduced in the datasets' description, each records in IIP is treated as an uncertain object with one instance. This means $\phi = 1$, i.e., every object T in IIP satisfies $\sum_{t\in T} p(t)$. Thus, the size of ARSP is the number of input instances. And B&B almost degenerates into LOOP, since no instances are pruned and no computations are reused. Fig. 6 (b)-(c) show the results on CAR and NBA with different m. It is noticed that attribute variance is pretty large in these two datasets, e.g., in NBA, about half of the players got zero points in some games but more than 20 points in other games. Therefore, relative



(a) IIP, vary m (b) CAR, vary m (c) NBA, vary m (d) NBA, vary d (e) NBA, vary c

Fig. 6. Running time of different algorithms and the size of ARSP on real datasets.

performance of algorithms are similar to synthetic datasets with large l. This also holds for results on NBA with different d and c which are shown in Fig. 6 (d)-(e).

D. Experimental Results under Weight Ratio Constraints.

Since the data structure stated in Theorem 6 is theoretical in nature, we introduce a specialized version of DUAL-MS for $d = 2$ to avoid this. Recall that for each instance t, we reduce the computation of $\Pr_{\text{rsky}}(t)$ to 2^{d-1} half-space reporting problems in Section V-A. When $d = 2$, we reinterpret the two queries as a continuous range query. Continue with Example 3. As shown in Fig. 7(a), when processing $t_{2,3}$, we regard $t_{2,3}$ as the origin, ray $y = t_{2,3}[2], x \geq t_{2,3}[1]$ as a base and represent each instance by an angle, e.g., $\theta = \pi + \arctan\frac{12-5}{9-6}$ for $t_{3,1}$. Then the two query lines $h_{t_{2,3},0} : t[2] \leq -0.5t[1] + 16.5$ and $h_{t_{2,3},1} : t[2] \leq -2t[1] + 30$ can be transformed to a range query $[\pi - \arctan\frac{1}{2}, 2\pi - \arctan 2]$ with respect to angle. With this transformation, we can use a simple binary search tree to organize the instances instead of the point location tree. We give an implementation of this specialized DUAL-MS and evaluate its performance on the NBA dataset. For reference, we attach a simple preprocessing strategy to KDTT$^+$, which removes all instances with zero skyline probability from I. Fig. 7(b) shows the running time of these two algorithms. Although the efficiency is improved, the huge preprocessing time prevents its application on big datasets.

The above drawbacks of DUAL-MS are alleviated when it comes to process eclipse queries. This is because eclipse is always a subset of skyline S, which has a logarithmic size in expectation. Meanwhile, the multi-level strategy is no longer needed since for each object $t \in S$, t belongs to the eclipse of D if and only if all point location queries on $S^*_{t,k}$ $(0 \leq k < 2^{d-1})$ return emptiness. We implement the DUAL-S for eclipse query processing, in which we use a kd-tree to index the dataset constructed by performing shifted strategy. For comparison, we also implement the state-of-the-art index-based algorithm QUAD [12] in C++ and compare their efficiency and scalability with respect to data cardinality n, data dimensionality d, and ratio range q. Similar to [12], the defaulted value is set as $n = 2^{14}$, $d = 3$, and $q = [0.36, 2.75]$. As shown in Fig. 8(a) and 8(b), the running time of these two methods increases with the growth of both n and d. DUAL-S outperforms QUAD by at least an order of magnitude and even more on high-dimensional datasets. The reason is that QUAD needs to iterate over the set of hyperplanes returned by the window query performed on its Intersection Index, and then reports all objects with zero order vector as the final result. This takes $O(s^2)$ time, where s is the skyline size of the dataset. But DUAL-S excludes an object from the result if there is a query returns non-empty result, which only take $O(s)$ time. Moreover, the hyperplane quadtree adopted in QUAD scales poorly with respect to d for the following two reasons. On the one hand, the tree index has a large fan-out since it splits all dimensions at each internal node. On the other hand, the number of intersection hyperplanes of a node decreases slightly relative to its parent, especially on high-dimensional datasets, which results in an unacceptable tree height. Moreover, as shown in Fig. 8(c), QUAD is more sensitive to the ratio range than DUAL-S because the number of hyperplanes returned by the window query actually determines the running time.

VII. Conclusions

In this paper, we study the problem of computing ARSP to aid multi-criteria decision making on uncertain datasets. We prove that no algorithm can compute ARSP in truly subquadratic time without preprocessing, unless the orthogonal vectors conjecture fails. Then, we propose two efficient algorithms to compute ARSP when $\mathcal{F}$ is a set of linear scoring functions whose weights are described by linear constraints. We use preprocessing techniques to further improve the query time under weight ratio constraints. Our thorough experiments over real and synthetic datasets demonstrate the effectiveness of ARSP and the efficiency of our proposed algorithms.

References

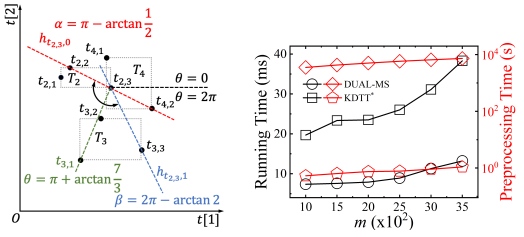
[1] J. Zhang, S. Cui, Y. Xu, Q. Li, and T. Li, "A novel data-driven stock price trend prediction system," *Expert Syst. Appl.*, vol. 97, pp. 60–69, 2018. [Online]. Available: https://doi.org/10.1016/j.eswa.2017.12.026

[2] Y. Zoabi, S. Deri-Rozov, and N. Shomron, "Machine learning-based prediction of covid-19 diagnosis based on symptoms," *npj digital medicine*, vol. 4, no. 1, p. 3, 2021.

[3] A. Mujumdar and V. Vaidehi, "Diabetes prediction using machine learning algorithms," *Procedia Computer Science*, vol. 165, pp. 292–299, 2019.

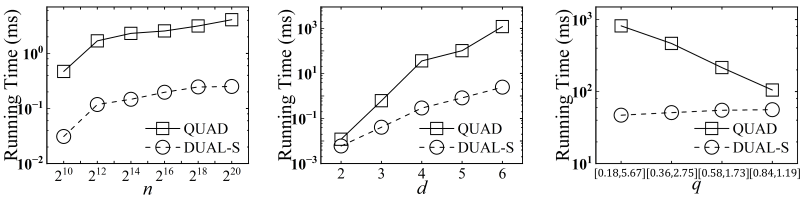
[4] K. Mouratidis and B. Tang, "Exact processing of uncertain top-k queries in multi-criteria settings," *Proc. VLDB Endow.*, vol. 11, no. 8, pp. 866–879, 2018. [Online]. Available: http://www.vldb.org/pvldb/vol11/p866-mouratidis.pdf

[5] C. Jin, K. Yi, L. Chen, J. X. Yu, and X. Lin, "Sliding-window top-k queries on uncertain streams," *VLDB J.*, vol. 19, no. 3, pp. 411–435, 2010. [Online]. Available: https://doi.org/10.1007/s00778-009-0171-0



(a) Specialized version (b) Running time on NBA

Fig. 7. A specialized version of DUAL-MS for $d = 2$ and its running time on NBA dataset.



(a) Effect of n (b) Effect of d (c) Effect of q

Fig. 8. Running time for eclipse query (IND).

[6] X. Liu, D. Yang, M. Ye, and W. Lee, "U-skyline: A new skyline query for uncertain databases," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 945–960, 2013. [Online]. Available: https://doi.org/10.1109/TKDE.2012.33

[7] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "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*, C. Koch, J. Gehrke, M. N. Garofalakis, D. Srivastava, K. Aberer, A. Deshpande, D. Florescu, C. Y. Chan, V. Ganti, C. Kanne, W. Klas, and E. J. Neuhold, Eds. ACM, 2007, pp. 15–26. [Online]. Available: http://www.vldb.org/conf/2007/papers/research/p15-pei.pdf

[8] D. Kim, H. Im, and S. Park, "Computing exact skyline probabilities for uncertain databases," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 12, pp. 2113–2126, 2012. [Online]. Available: https://doi.org/10.1109/TKDE.2011.164

[9] Y. S. Eum, K. S. Park, and S. H. Kim, "Establishing dominance and potential optimality in multi-criteria analysis with imprecise weight and value," *Comput. Oper. Res.*, vol. 28, no. 5, pp. 397–409, 2001. [Online]. Available: https://doi.org/10.1016/S0305-0548(99)00124-0

[10] P. Ciaccia and D. Martinenghi, "Reconciling skyline and ranking queries," *Proc. VLDB Endow.*, vol. 10, no. 11, pp. 1454–1465, 2017. [Online]. Available: http://www.vldb.org/pvldb/vol10/p1454-martinenghi.pdf

[11] W. Wang, R. C. Wong, and M. Xie, "Interactive search for one of the top-k," in *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, G. Li, Z. Li, S. Idreos, and D. Srivastava, Eds. ACM, 2021, pp. 1920–1932. [Online]. Available: https://doi.org/10.1145/3448016.3457322

[12] J. Liu, L. Xiong, Q. Zhang, J. Pei, and J. Luo, "Eclipse: Generalizing knn and skyline," in *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 2021, pp. 972–983. [Online]. Available: https://doi.org/10.1109/ICDE51399.2021.00089

[13] L. Li, H. Wang, J. Li, and H. Gao, "A survey of uncertain data management," *Frontiers Comput. Sci.*, vol. 14, no. 1, pp. 162–190, 2020. [Online]. Available: https://doi.org/10.1007/s11704-017-7063-z

[14] S. A. Fay and J. Xie, "Probabilistic goods: A creative way of selling products and services," *Mark. Sci.*, vol. 27, no. 4, pp. 674–690, 2008. [Online]. Available: https://doi.org/10.1287/mksc.1070.0318

[15] Y. Zhang, W. Zhang, X. Lin, B. Jiang, and J. Pei, "Ranking uncertain sky: The probabilistic top-k skyline operator," *Inf. Syst.*, vol. 36, no. 5, pp. 898–915, 2011. [Online]. Available: https://doi.org/10.1016/j.is.2011.03.008

[16] Z. Yang, K. Li, X. Zhou, J. Mei, and Y. Gao, "Top k probabilistic skyline queries on uncertain data," *Neurocomputing*, vol. 317, pp. 1–14, 2018. [Online]. Available: https://doi.org/10.1016/j.neucom.2018.03.052

[17] H. Yong, J. Lee, J. Kim, and S. Hwang, "Skyline ranking for uncertain databases," *Inf. Sci.*, vol. 273, pp. 247–262, 2014. [Online]. Available: https://doi.org/10.1016/j.ins.2014.03.044

[18] M. Hua, J. Pei, W. Zhang, and X. Lin, "Efficiently answering probabilistic threshold top-k queries on uncertain data," in *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico*, G. Alonso, J. A. Blakeley, and A. L. P. Chen, Eds. IEEE Computer Society, 2008, pp. 1403–1405. [Online]. Available: https://doi.org/10.1109/ICDE.2008.4497570

[19] M. A. Soliman, I. F. Ilyas, and K. C. Chang, "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*, R. Chirkova, A. Dogac, M. T. Özsu, and T. K. Sellis, Eds. IEEE Computer Society, 2007, pp. 896–905. [Online]. Available: https://doi.org/10.1109/ICDE.2007.367935

[20] X. Wang, D. Shen, and G. Yu, "Uncertain top-k query processing in distributed environments," *Distributed Parallel Databases*, vol. 34, no. 4, pp. 567–589, 2016. [Online]. Available: https://doi.org/10.1007/s10619-015-7188-8

[21] K. Yi, F. Li, G. Kollios, and D. Srivastava, "Efficient processing of top-k queries in uncertain databases," in *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico*, G. Alonso, J. A. Blakeley, and A. L. P. Chen, Eds. IEEE Computer Society, 2008, pp. 1406–1408. [Online]. Available: https://doi.org/10.1109/ICDE.2008.4497571

[22] T. Ge, S. B. Zdonik, and S. Madden, "Top-k queries on uncertain data: on score distribution and typical answers," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, U. Çetintemel, S. B. Zdonik, D. Kossmann, and N. Tatbul, Eds. ACM, 2009, pp. 375–388. [Online]. Available: https://doi.org/10.1145/1559845.1559886

[23] K. Mouratidis, K. Li, and B. Tang, "Marrying top-k with skyline queries: Relaxing the preference input while producing output of controllable size," in *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, G. Li, Z. Li, S. Idreos, and D. Srivastava, Eds. ACM, 2021, pp. 1317–1330. [Online]. Available: https://doi.org/10.1145/3448016.3457299

[24] L. Qian, J. Gao, and H. V. Jagadish, "Learning user preferences by adaptive pairwise comparison," *Proc. VLDB Endow.*, vol. 8, no. 11, pp. 1322–1333, 2015. [Online]. Available: http://www.vldb.org/pvldb/vol8/p1322-qian.pdf

[25] M. J. Atallah and Y. Qi, "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*, J. Paredaens and J. Su, Eds. ACM, 2009, pp. 279–287. [Online]. Available: https://doi.org/10.1145/1559795.1559837

[26] M. J. Atallah, Y. Qi, and H. Yuan, "Asymptotically efficient algorithms for skyline probabilities of uncertain data," *ACM Trans. Database Syst.*, vol. 36, no. 2, pp. 12:1–12:28, 2011. [Online]. Available: https://doi.org/10.1145/1966385.1966390

[27] P. Afshani, P. K. Agarwal, L. Arge, K. G. Larsen, and J. M. Phillips, "(approximate) uncertain skylines," *Theory Comput. Syst.*, vol. 52, no. 3, pp. 342–366, 2013. [Online]. Available: https://doi.org/10.1007/s00224-012-9382-7

[28] H. T. H. Nguyen and J. Cao, "Preference-based top-k representative skyline queries on uncertain databases," in *Advances in Knowledge Discovery and Data Mining - 19th Pacific-Asia Conference, PAKDD 2015, Ho Chi Minh City, Vietnam, May 19-22, 2015, Proceedings, Part II*, ser. Lecture Notes in Computer Science, T. H. Cao, E. Lim, Z. Zhou, T. B. Ho, D. W. Cheung, and H. Motoda, Eds., vol. 9078. Springer, 2015, pp. 280–292. [Online]. Available: https://doi.org/10.1007/978-3-319-18032-8_22

[29] S. Abiteboul, P. C. Kanellakis, and G. Grahne, "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*, U. Dayal and I. L. Traiger, Eds. ACM Press, 1987, pp. 34–48. [Online]. Available: https://doi.org/10.1145/38713.38724

[30] K. Bringmann, "Fine-grained complexity theory (tutorial)," in *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, ser. LIPIcs, R. Niedermeier and C. Paul, Eds., vol. 126. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 4:1–4:7. [Online]. Available: https://doi.org/10.4230/LIPIcs.STACS.2019.4

[31] P. K. Agarwal, "Simplex range searching and its variants: A review," *A Journey Through Discrete Mathematics*, pp. 1–30, 2017.

[32] J. Liu, H. Zhang, L. Xiong, H. Li, and J. Luo, "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*, J. Bailey, A. Moffat, C. C. Aggarwal, M. de Rijke, R. Kumar, V. Murdock, T. K. Sellis, and J. X. Yu, Eds. ACM, 2015, pp. 1511–1520. [Online]. Available: https://doi.org/10.1145/2806416.2806452

[33] X. Gao, J. Li, and D. Miao, "Computing all restricted skyline probabilities for uncertain data," 2023. [Online]. Available: https://arxiv.org/abs/2303.00259

[34] J. S. Dyer and R. K. Sarin, "Measurable multiattribute value functions," *Oper. Res.*, vol. 27, no. 4, pp. 810–822, 1979. [Online]. Available: https://doi.org/10.1287/opre.27.4.810

[35] F. P. Preparata and M. I. Shamos, *Computational geometry: an introduction*. Springer Science & Business Media, 2012.

[36] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, pp. 469–483, 1996. [Online]. Available: https://doi.org/10.1145/235815.235821

[37] J. S. Greenfield, "A proof for a quickhull algorithm," 1990.

[38] M. Henk, J. Richter-Gebert, and G. M. Ziegler, "Basic properties of convex polytopes," in *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017, pp. 383–413.

[39] d. B. Mark, C. Otfried, v. K. Marc, and O. Mark, *Computational Geometry Algorithms and Applications*. Spinger, 2008.

[40] S. Meiser, "Point location in arrangements of hyperplanes," *Inf. Comput.*, vol. 106, no. 2, pp. 286–303, 1993. [Online]. Available: https://doi.org/10.1006/inco.1993.1057

[41] J. L. Bentley, "Decomposable searching problems," *Inf. Process. Lett.*, vol. 8, no. 5, pp. 244–251, 1979. [Online]. Available: https://doi.org/10.1016/0020-0190(79)90117-0

[42] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*, D. Georgakopoulos and A. Buchmann, Eds. IEEE Computer Society, 2001, pp. 421–430. [Online]. Available: https://doi.org/10.1109/ICDE.2001.914855

[43] X. Gao. (2022, oct) Source code. [Online]. Available: https://github.com/gaoxy914/ARSP