Matching Dominance: Capture the Semantics of Dominance for Multi-dimensional Uncertain Objects

Ying Zhang[§], Wenjie Zhang[†], Xuemin Lin[†], Muhammad Aamir Cheema[‡], Chengqi Zhang[§]

[§] *QCIS, University of Technology, Sydney,* [†] *The University Of New South Wales,* [‡] *Monash University*

Ying.Zhang@uts.edu.au, {zhangw, lxue}@cse.unsw.edu.au, aamir.cheema@monash.edu, Chengqi.Zhang@uts.edu.au

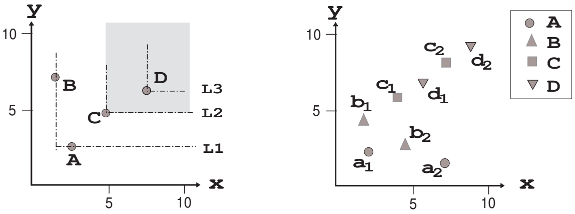
ABSTRACT

The dominance operator plays an important role in a wide spectrum of multi-criteria decision making applications. Generally speaking, a dominance operator is a *partial order* on a set $\mathcal{O}$ of objects, and we say the dominance operator has the monotonic property regarding a family of ranking functions $\mathcal{F}$ if o_1 *dominates* o_2 implies $f(o_1) \geq f(o_2)$ for any ranking function $f \in \mathcal{F}$ and objects o_1, $o_2 \in \mathcal{O}$. The dominance operator on the multi-dimensional points is well defined, which has the monotonic property regarding any monotonic ranking (scoring) function. Due to the uncertain nature of data in many emerging applications, a variety of existing works have studied the semantics of ranking query on uncertain objects. However, the problem of dominance operator against multi-dimensional uncertain objects remains open. Although there are several attempts to propose dominance operator on multi-dimensional uncertain objects, none of them claims the monotonic property on these ranking approaches.

Motivated by this, in this paper we propose a novel *matching* based *dominance* operator, namely **matching dominance**, to capture the semantics of the dominance for multi-dimensional uncertain objects so that the new dominance operator has the monotonic property regarding the monotonic *parameterized ranking* function, which can unify other popular ranking approaches for uncertain objects. Then we develop a layer indexing technique, Matching Dominance based Band (**MDB**), to facilitate the top k queries on multi-dimensional uncertain objects based on the *matching dominance* operator proposed in this paper. Efficient algorithms are proposed to compute the MDB index. Comprehensive experiments convincingly demonstrate the effectiveness and efficiency of our indexing techniques.

1. INTRODUCTION

Due to various factors such as data incompleteness, limitation of measuring equipment, delay or loss of data updates and privacy preservation, the objects in many applications such as data integration, environmental surveillance, geographic information system, and location based service are described by the uncertain object model; that is, an uncertain object is described by a probability density function (PDF) or a set of instances (points) with occurrence probabilities. For instance, in the meteorology system sensors collect the temperature and relative humidity at a large number of sites. The reading may be uncertain, and hence each site can be modeled by a 2-dimensional uncertain object. There are some popular web sites (e.g., `http://www.restaurantratingz.com`) in which each restaurant may be evaluated by different customers against food, ambience, and service. The weight (occurrence probability) of each rating (an instance recording food-rate, ambience-rate and service-rate) for a restaurant (a 3-dimensional uncertain object) can be derived based on the customers' knowledge and experience levels.



(a) Certain Objects (b) Uncertain Objects

Figure 1: Certain Objects and Uncertain Objects

Top k Queries. As ranking is an essential analytic method, it is natural and fundamental to investigate how to rank a set of uncertain objects. Conventional top k queries assume that the objects in the database are points, namely *certain objects*, in a multi-dimensional space as shown in Figure 1(a), where each dimension corresponds to one feature of the objects that users are interested in. Given a *scoring function* f based on a user's weight strategy on the features and a set $\mathcal{O}$ of certain objects, the top k query retrieves the k certain objects with the **highest scores**.

However, as shown in Figure 1(b), it is non-trivial to identify the top k uncertain objects due to the presence of multiple instances per uncertain object and their occurrence probabilities. For instance, in Figure 1(b) we may have $f(a_1) > f(b_1)$ and $f(a_2) < f(b_2)$ for a given *scoring* function f. In recent years, various ranking approaches (e.g., U-k rank [20], Global-top k [22] rank, expected rank [6] and parameterized rank [11]) have been proposed to retrieve the top k uncertain objects based on their score distributions derived from a *scoring* function f [6]. Particularly, the monotonic *parameterized ranking* method can unify other popular

SSDBM '14, June 30 - July 02 2014, Aalborg, Denmark.
Copyright 2014 ACM 978-1-4503-2722-0/14/06 ...$15.00.
http://dx.doi.org/10.1145/2618243.2618246.

ranking functions. Therefore, in this paper we only need to investigate the dominance operator over the *parameterized ranking* method.

Conventional Dominance Operator. The *dominance* operator on a set of points (certain objects) is well defined in the database literature and known to well capture the monotonicity of a ranking function. We say a point p *dominates* another point q, denoted by $p \prec q$, if p is not worse than q on each dimension and p is better than q on at least one dimension. Without loss of generality, we assume the **smaller coordinate value is preferred** in this paper. In Figure 1(a), we have A *dominates* C and A does not *dominate* B. Due to the monotonic property of the dominance operator, we have $f(A) \geq f(B)$ for any monotonic scoring function f, if A *dominates* B. For instance, the scores of the certain objects (e.g., D) within the shaded area in Figure 1(a) have scores smaller or equal to $f(C)$ for any monotonic *scoring* function f. This monotonic property is important for the top k queries since it enables the precomputation of a layer indexing structure $\{\mathcal{L}_1, \ldots, \mathcal{L}_K\}$, namely skyband [15, 14, 23], where an object is assigned to the i-th layer if it is *dominated* by $i-1$ other objects. Then for any monotonic *scoring* function, only the objects on the first k layers can be the candidate objects of a top k query, which significantly reduces the query costs since the skyband can be pre-computed off-line and usually only a small number of objects are accessed for each top k query, especially when k is small.

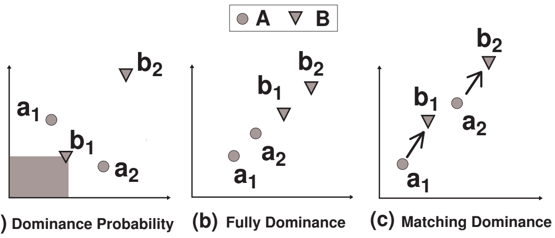
EXAMPLE 1. *In Figure 1(a), given four certain objects* $\{A, B, C, D\}$*, we have A dominates C, A dominates D, and C dominates D. Consequently, the first layer of the Skyband is* $\{A, B\}$ *since they are not dominated by any other object. Objects C and D go to the second and third layers respectively.*

Motivation. Motivated by the above observation, we aim to develop a dominance operator on multi-dimensional uncertain objects to effectively index multi-dimensional uncertain objects. Such a dominance operator should have the desirable **monotonic property** so that if a multi-dimensional uncertain object U dominates another uncertain object V, U is not ranked lower than V regarding any monotonic *parameterized ranking* function.

To the best of our knowledge, although there are some related works in the context of skyline computation on multi-dimensional uncertain objects, none of them claims the monotonic property on the monotonic *parameterized ranking* function. There are three categories of studies involving the dominance operator among multi-dimensional uncertain objects: 1) P-*domination* [1], 2) probabilistic dominance [16, 2], and 3) stochastic order [18, 13, 21].

P-*domination* is proposed in [1] to capture the dominance among uncertain objects. However, P-*domination* operator is defined based on the tuple-level uncertain object model (i.e., there is only one instance for each uncertain object with a particular existence probability); it is not generally applicable to uncertain objects with multiple instances. Moreover, the ranking function is limited to the *expected rank* [6] which is a special case of *parameterized ranking* function.

The probabilistic dominance in [16, 2] is to compute the probability of one object which is not dominated by the other objects based on the possible world semantics. Given two uncertain objects A and B, we use $Pr(A \prec B)$ to denote the probability that A dominates B, where $0 \leq Pr(A \prec B) \leq 1$. However, it is not safe to claim the monotonic property of the dominance unless $Pr(A \prec B) = 1$, namely **fully dominance**; that is, we have $a \prec b$ for any instance $a \in A$ and $b \in B$. Regarding the example in Figure 2(a), assuming A and B are independent, $Pr(A \prec B)$ can be infinitely close to 1 when the occurrence probability of b_1 approaches 0. However, we can have a monotonic *scoring* function f such that $f(p) = 1$ if $p \prec b_1$ (i.e., in the shaded area), and $f(p) = 0$ otherwise. Then B should have higher rank regarding f since $f(a_1) = f(a_2) = 0$ while $f(b_1) = 1$. Clearly, as shown in Figure 2(b), the *fully dominance* operator should support any reasonable ranking function. Nevertheless, we observe that the condition of the *fully dominance* is *strict* in the sense that two uncertain objects are incomparable when their instances slightly overlap.



(a) Dominance Probability (b) Fully Dominance (c) Matching Dominance

Figure 2: Motivation of Matching Dominance

The usual stochastic order [18] has been extensively used in the fields of finance and economics, which is a dominance operator against uncertain objects. An uncertain object U dominates another uncertain object V regarding usual stochastic order if the probabilistic mass of U is always larger than that of V regarding any lower set [1]. Recently, Zhang *et al.* [21] develop efficient algorithms to verify the dominance regarding the general stochastic order as well as the corresponding skyline computation. [18] shows that the usual stochastic order captures the monotonicity of the *expected scores* functions. Nevertheless, it is not clear that if the usual stochastic order has the monotonic property regarding popular ranking approaches proposed in the literature (e.g., *parameterized ranking* function).

These motivate us to propose a new dominance operator to properly capture the semantics of dominance of multi-dimensional uncertain objects with the desirable monotonic property.

Matching Dominance. To introduce the basic idea of the *matching dominance*, we start with a simple example in which we assume that objects have the same number of instances and each instance has the same occurrence probability. Then we say an uncertain object U *matching dominates* V if there is a *one-to-one* mapping function ζ between U and V, so that for each instance $u \in U$ we have $u \prec \zeta(u)$. In Figure 2(c), suppose both A and B have two instances and each instance has the occurrence probability 0.5, we have A *matching dominates* B since we can map a_1 and a_2 to b_1 and b_2 respectively. It is interesting that the problem of *matching dominance* verification in the example can be converted to the bipartite graph perfect matching problem [5] where vertices are instances of two uncertain objects and there is an edge between two vertices (i.e., instances) u and v if v is *dominated* by u.

[1] We say a points set $\mathcal{S}$ is a lower set if for each pair of points $x \preceq y$, $y \in \mathcal{S}$ implies $x \in \mathcal{S}$.

As in practice, however, the above assumptions may not hold for uncertain objects. In Section 3, we carefully define the *matching dominance* operator in a similar way but **without** the above equality assumptions on the number of instances and the occurrence probability. We theoretically show that the proposed *matching dominance* operator can support the monotonic *parameterized ranking* function; that is, an uncertain object U is not ranked lower than any uncertain object V if V is *matching dominated* by U. Moreover, we show that the *matching dominance* operator is equivalent to the usual stochastic order [18] since dominance verifications of both operators can be transformed to the well known max-flow problem [5]. This implies that the techniques developed in [21] can be immediately applied to verify the *matching dominance*. Given the definition of *matching dominance*, we can compute the MDB index of a set $\mathcal{O}$ of uncertain objects, and an uncertain object is assigned to the i-th layer if it is *matching dominated* by $i-1$ other uncertain objects in $\mathcal{O}$. We also develop an efficient algorithm to construct MDB index in Section 4. Note that the first layer of MDB index corresponds to the stochastic skyline [21], which can provide a minimal candidate for the optimal solutions (top 1 result) for any monotonic *parameterized ranking* function.

Contributions. Our principle contributions in this paper can be summarized as follows.

- We formally introduce a novel dominance operator on multi-dimensional uncertain objects, termed **matching dominance**, which is a natural extension of the traditional dominance operator. We show that *matching dominance* operator has the monotonic property regarding the monotonic *parameterized ranking* function. Moreover, it is shown that the *matching dominance* is equivalent to the well known usual stochastic order [18], and hence the dominance verification techniques developed in [21] can be immediately applied.
- Based on the *matching dominance* operator, we propose a layer indexing technique for uncertain objects, named *matching dominance* based band (**MDB** for short) index. We also develop an efficient MDB computation algorithm following the branch and bound paradigm.
- Comprehensive experiments demonstrate the effectiveness and efficiency of our indexing techniques.

Organization of the paper. The rest of the paper is organized as follows. Section 2 introduces the *parameterized ranking* function and problem statement. Section 3 formally defines the *matching dominance* operator and the corresponding verification algorithm. Section 4 develops efficient MDB algorithm. Some possible extensions are discussed in Section 5. The experimental results are reported in Section 6. This is followed by the related work presented in Section 7. We conclude our paper in Section 8.

2. BACKGROUND

In this section, we first formally introduce the uncertain object model and the *parameterized ranking* function, as well as the problem statement. Then we show how to calculate the rank scores of the uncertain objects based on the generating function. Table 1 summarizes notations frequently used throughout the paper.

Notation	Meaning
U, V, A, B	uncertain objects
u, v, a, b	instances (points) of the uncertain objects
$u \prec v$	u *dominates* v
$u.D_i$	i-th dimensional coordinate value of u
p_u	occurrence probability of the instance u
PRF^{ω}	*parameterized ranking* function
$\Upsilon(U)$	the rank score of U
U_{mbb}	minimal bounding box of U
U_f	score distribution of U regarding f
$Pr(U_f > c)$	Probability that U_f is larger than c
$P_{U,v}$	$Pr(U_f > f(v))$. Probability that score of U is larger than score of instance v
$M_{U \prec V}$	a dominance match for U and V
t ($t.u, t.v, t.p$)	a tuple t in the dominance match $M_{U \prec V}$ (instance from u, instance from v , probability of t)
$P(M_{U \prec V})$	the weight of a dominance match $M_{U \prec V}$
$Pr(U \prec V)$	dominance match probability for U and V
$U \prec_M V$	U *matching dominates* V
$\mu(U)$ ($\mu(e)$)	*mean* of an uncertain object U (entry e)
$\mathcal{L}(\mathcal{O})$	MDB index for a set $\mathcal{O}$ of objects

Table 1: The summary of notations.

2.1 Problem Definition

In this paper, a point (instance) p is in a d-dimensional space and the i-th dimensional coordinate value of p is denoted by $p.D_i$. Without loss of generality, we assume **smaller coordinate values** are preferred and the *monotonic* function refers to the **non-increasing** function. For two points p and q, *p dominates q*, denoted by $p \prec q$, if $p.D_i \leq q.D_i$ for all dimension $i \in [1, d]$ and there is a dimension $j \in [1, d]$ with $p.D_j < q.D_j$. Meanwhile, we use $p \preceq q$ to denote that *p dominates or equals q*.

The following lemma is immediate based on the definition of the monotonic *scoring* function.

Lemma 1. *For any two points p and q, we have $f(p) \geq f(q)$ if $p \prec q$ and f is a monotonic scoring function.*

In this paper, we assume a *scoring* function f is monotonic whenever the context is clear.

Uncertain Object Model. An uncertain object can be described either continuously or discretely. In this paper, we focus on the *discrete* case; that is, an uncertain object U consists of a set $\{u_1, u_2, \ldots, u_m\}$ of instances (points). An instance u_i occurs with probability p_{u_i}, and $\sum_{i=1}^{m} p_{u_i} = 1$. In Section 5, we discuss the cases where an uncertain object is described by a probabilistic density function (PDF) (i.e., continuous case). Moreover, in this paper we assume that the uncertain objects are *independent* to each other. In the following of the paper, we use *object* to denote *multi-dimensional uncertain object* whenever there is no ambiguity. Given an object U, U_{mbb} denotes the minimal bounding box which contains all of the instances of U. Let U^-_{mbb} (U^+_{mbb}) denote the lower (upper) corner of U_{mbb}, we have $U^-_{mbb} \preceq p$ and $p \preceq U^+_{mbb}$ for any point $p \in U_{mbb}$.

Score Distribution. Given an object U and a *scoring* function f, the score of U regarding f corresponds to a score distribution $U_f = \{f(u), p_u\}$ for all instances $u \in U$. We use $Pr(U_f > c)$ to denote the probability that U_f is larger than the value c, *i.e.*, $Pr(U_f > c) = \sum_{u \in U \wedge f(u) > c} p_u$.

Example 2. *Figure 3 shows the score distributions of three uncertain objects A, B and C. Particularly, as* $f(a_1) = 15$, $f(a_2) = 5$, *and* $p_{a_1} = p_{a_2} = 0.5$, *we have* $A_f = \{(15, 0.5), (5, 0.5)\}$. *Similarly, we have* $B_f = \{(20, 0.2), (2, 0.8)\}$, $C_f = \{(10, 0.5), (7, 0.5)\}$ *and* $Pr(B_f > 15) = 0.2$.

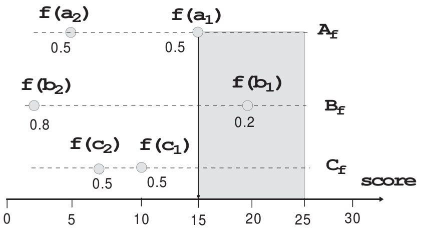


Figure 3: Score Distributions

Possible World Semantics. For a set $\mathcal{O}$ of objects, a possible world W is a set of instances with one instance from each object. Given a possible world W and a *scoring* function f, the score of an object U is $f(u)$ where u is the instance of U appearing in the possible world W. For each possible world W, objects are ranked based on the scores of their corresponding instances occurring in W. In this paper, we use $r_W(U)$ to denote the rank of an object in the possible world W regarding f, which is abbreviated to $r(U)$ whenever there is no ambiguity. Let $\mathcal{W}$ denote the set of all possible worlds, and we have $\sum_{W \in \mathcal{W}} Pr(W) = 1$ where $Pr(W)$ is the occurring probability of the possible world W.

Parameterized Ranking Function. The score distributions of a set of objects can be ranked by the top k semantics studied for uncertain data in the literature. In this paper, we focus on the *parameterized ranking* function (PRF^{ω}) proposed in [11] and the following is its formal definition.

Definition 1 (PRF^{ω}). *Let* ω *be a weighted function which maps an object-rank pair to a complex number, the* ***rank score*** *of an object* U*, denoted by* $\Upsilon(U)$*, is defined as follows.*

$$\Upsilon(U) = \sum_{i>0} \omega(U, i) \times Pr(r(U) = i) \qquad (1)$$

where $\omega(U, i)$ *denotes the weight of* U *if it is ranked at the i-th position in the possible world, and* $Pr(r(U) = i)$ *denotes the probability of* U *being ranked at the i-th position, i.e.,* $Pr(r(U) = i) = \sum_{W \in \mathcal{W} \wedge r_W(U) = i} Pr(W)$. *Recall that* $r_W(U)$ *denotes the rank of* U *in the possible world* W.

In this paper, we assume $\omega(U, i) = \omega(i)$ (i.e., the weight is independent of U). Therefore, we have

$$\Upsilon(U) = \sum_{i>0} \omega(i) \times Pr(r(U) = i) \qquad (2)$$

In practice, the *parameterized ranking* function is **monotonic**; that is, we have $w(i) \geq w(j)$ for any two ranking positions i and j where $i < j$ because the higher position is usually at least as desirable as those behind it and thus should be given a higher weight. Moreover, the *scoring* function f used is monotonic. As shown in [11], other popular ranking methods can be unified by the monotonic *parameterized ranking* functions.

In this paper, we return the k objects with the **highest rank scores**. Ties are broken arbitrarily in this paper.

Problem Statement. Given two multi-dimensional uncertain objects U and V, we aim to propose a *dominance* operator between U and V so that for a monotonic *parameterized ranking* function PRF^{ω}, we have $\Upsilon(U) \geq \Upsilon(V)$ if U *dominates* V. Then for a set $\mathcal{O}$ of multi-dimensional uncertain objects, a layer indexing $\mathcal{L}(\mathcal{O})$ is constructed where an object is located at the i-th layer, denoted by $\mathcal{L}_i(\mathcal{O})$, if it is *dominated* by $i - 1$ other objects in $\mathcal{O}$.

2.2 Computing Rank Score $\Upsilon(U)$

As shown in [11], the rank score of an object U can be calculated by the summation of the rank scores of its instances. For an instance $u \in U$, we can use the following generating function $\mathcal{F}(\mathbf{x}, u)$ to calculate the rank score of u, where $P_{V,u}$ is the probability that V_f is larger than $f(u)$ (i.e., $Pr(V_f > f(u))$). Intuitively, a small $P_{V,u}$ is in favor of the rank score of the instance u. Recall that V_f is the score distribution of the object V regarding f.

$$\mathcal{F}(u, \mathbf{x}) = \prod_{V \in \mathcal{O} \setminus U} (1 - P_{V,u} + P_{V,u}\ \mathbf{x})\ p_u\ \mathbf{x} \qquad (3)$$

As shown in [11], we have $Pr(r(u) = i) = c_i$ where $r(u)$ is the rank position of instance u and c_i is the coefficient of $\mathbf{x}^i$ in $\mathcal{F}(\mathbf{x}, u)$. Therefore, we have

$$\Upsilon(u) = \sum_{1 \leq i \leq n} c_i \times \omega(i) \qquad (4)$$

and

$$\Upsilon(U) = \sum_{u \in U} \Upsilon(u) \qquad (5)$$

Example 3. *In Figure 3, we have* $f(a_1) = 15$ *and hence* $Pr(B_f > f(a_1)) = 0.2$ *(i.e., the probability mass of the instances of B in the shaded area) and* $Pr(C_f > f(a_1)) = 0.0$. *According to Equation 4,* $\mathcal{F}(\mathbf{x}, a_1) = (0.8 + 0.2\ \mathbf{x}) \times (1) \times 0.5\ \mathbf{x} = 0.4\ \mathbf{x} + 0.1\ \mathbf{x}^2$. *Therefore, we have* $Pr(r(a_1) = 1) = 0.4$ *and* $Pr(r(a_1) = 2) = 0.1$. *Similarly,* $\mathcal{F}(\ \mathbf{x}, a_2) = (0.8 + 0.2\ \mathbf{x}) \times (\ \mathbf{x}\ ) \times 0.5\ \mathbf{x} = 0.4\ \mathbf{x}^2 + 0.1\ \mathbf{x}^3$, *and hence* $Pr(r(a_2) = 1) = 0$,$Pr(r(a_2) = 2) = 0.4$ *and* $Pr(r(a_2) = 3) = 0.1$. *Suppose* $\omega(1) = 3$, $\omega(2) = 2$, *and* $\omega(3) = 1$ *in* PRF^{ω}, *then* $\Upsilon(A) = \Upsilon(a_1) + \Upsilon(a_2) = (0.4 \times 3 + 0.1 \times 2) + (0.4 \times 2 + 0.1 \times 1) = 2.3$ *according to Equation 4. Similarly, we have* $\Upsilon(B) = 1.4$ *and* $\Upsilon(C) = 2.1$. *Therefore,* $\Upsilon(A) > \Upsilon(C) > \Upsilon(B)$.

3. MATCHING DOMINANCE OPERATOR

In this section, we first formally introduce the *matching dominance* operator. Then we present some of its important properties used in Section 4, and show that the *matching dominance* is equivalent to the well known usual stochastic order [18], and hence the dominance verification techniques developed in [21] can be immediately applied.

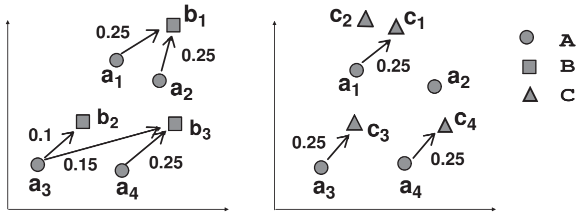
3.1 Definition of Matching Dominance

For two objects U and V, we define a *dominance match* for U and V as follows.

Definition 2 (**Dominance Match**). *Given two objects* U *and* V*, a dominance match between* U *and* V *is denoted by* $M_{U \prec V}$*, which consists of a set of tuples* $\{t <u, v, p>\}$ *where* $t.u \in U$*,* $t.v \in V$*,* $t.u \prec t.v$*, and* $t.p$ *is the probability mass of the tuples with* $\sum_{t \in M_{U \prec V} \wedge t.u = u} t.p \leq p_u$

and $\sum_{t\in M_{U\prec V}\wedge t.v=v} t.p \le p_v$. *The* ***weight*** *of a dominance match* $M_{U\prec V}$*, denoted by* $P(M_{U\prec V})$*, is the probability mass of the tuples in* $M_{U\prec V}$*, i.e.,* $\sum_{t\in M_{U\prec V}} t.p$*.*

Note that according to the above definition, an instance $u \in U$ or an instance $v \in V$ may appear in **multiple tuples** in $M_{U\prec V}$. But the total probabilities contributed from an instance u (v) cannot exceed its occurrence probability p_u (p_v). Without loss of generality, we assume the object U (V) does not have instances with duplicate locations.



(a) $\mathbf{M}_{A\prec B}$ **(b)** $\mathbf{M}_{A\prec C}$

Figure 4: Dominance Match

Example 4. *In Figure 4, suppose each instance in A and C has the occurrence probability* 0.25*,* $p_{b_1} = 0.5$*,* $p_{b_2} = 0.1$ *and* $p_{b_3} = 0.4$*. Figure 4(a) shows a dominance match for A and B where an edge from* a_1 *to* b_1 *denotes a tuple* (a_1, b_1, p) *where p is labeled beside the edge. Then* $M_{A\prec B} = \{(a_1, b_1, 0.25), (a_2, b_1, 0.25), (a_3, b_2, 0.1), (a_3, b_3, 0.15), (a_4, b_3, 0.25)\}$ *where* $P(M_{A\prec B}) = 1$*. As an alternative, we may have another dominance match* $M^2_{A\prec B}$ *in Figure 4(a), where* $M^2_{A\prec B} = \{(a_2, b_1, 0.25), (a_3, b_1, 0.20)\ (a_4, b_3, 0.20)\}$ *where* $P(M^2_{A\prec B}) = 0.65$*. Regarding* $M_{A\prec C}$ *in Figure 4(b), there are three tuples and* $P(M_{A\prec C}) = 0.75$*.*

Bellow, we use the dominance match with the maximum weight to define the *matching dominance probability* for U and V, denoted by $Pr(U \prec_M V)$.

Definition 3 (**Matching Dominance Probability**). *Given two objects* U *and* V*,*

$$Pr(U \prec_M V) = \max_{M_{U\prec V}\in\mathcal{M}} P(M_{U\prec V}) \tag{6}$$

where $\mathcal{M}$ *represents a set of all possible dominance matches for* U *and* V*.*

Clearly, for any two objects U and V, $Pr(U \prec_M V) \le 1$ according to Definition 2 and 3. Following is the definition of *matching dominance* operator.

Definition 4 (**Matching Dominance Operator**). *Given two objects* U *and* V*, we say* U *matching dominates* V*, denoted by* $U \prec_M V$*, if* $Pr(U \prec_M V) = 1$*, i.e., there is a dominance match for* U *and* V *with weight* 1*.*

Example 5. *In Figure 4(a), we have* $Pr(A \prec_M B) = 1$ *(i.e.,* $A \prec_M B$*) since we have* $\sum_{t\in M_{A\prec B}} t.p = 1$*.* $M_{A\prec C}$ *in Figure 4(b) already achieves the maximum weight since* a_2 *matching dominates none of the instances in C, and hence* $A \not\prec_M C$*.*

3.2 Important Properties

In this subsection, we introduce some important properties of the *matching dominance*. Specifically, we first show that *matching dominance* can guarantee that if an object U *matching dominates* another object V, U is not ranked lower than V regarding any monotonic *parameterized ranking* function. Then we show that *matching dominance* is a *strict* order over a set of objects. Finally, we introduce two theorems to provide effective *pruning* and *validation* rules.

Monotonic Property. Given two objects U and V, following theorem indicates that for any monotonic *parameterized ranking* function, we can safely claim that $\Upsilon(U) \ge \Upsilon(V)$ if $U \prec_M V$. The motivation of the proof is that we can decompose the instances of U (V) into m instances since there is a *dominance match* $M_{U\prec V}$ with m tuples and $\sum_{i=1}^{m} t_i.p = 1$, so that there is a *one-to-one* mapping function for U and V where $t.u$ is mapped to $t.v$ for each $t \in M_{U\prec V}$ with $p_{t.u} = p_{t.v}$ and $t.u \prec t.v$. We show that, $t_i.u$ is always ranked not lower than $t_i.v$ for $1 \le i \le m$, which is intuitive since $f(t_i.u) \ge f(t_i.v)$, then the correctness of the theorem follows.

Theorem 1. *Given two objects* U *and* V*, for any monotonic parameterized ranking function* PRF^ω*,* $\Upsilon(U) \ge \Upsilon(V)$ *if* $U \prec_M V$*.*

Proof. Since $U \prec_M V$, there is a dominance match $M_{U\prec V}$ with $P(M_{U\prec V}) = 1$. Let t_1, t_2, …, t_m denote m tuples in $M_{U\prec V}$ with $\sum_{i=1}^{m} t.p = 1$. The object U can be represented by m instances where $u_i = t_i.u$ and $p_{u_i} = t_i.p$ for $1 \le i \le m$. Similarly, the object V is represented by m instances with $v_i = t_i.v$ and $p_{v_i} = t_i.p$ $(1 \le i \le m)$. Since $u_i \prec v_i$ based on the definition of *dominance match* (Definition 2), we have $f(u_i) \ge f(v_i)$ for $1 \le i \le l$ according to Lemma 1. For an instance $u_i \in U$, Equation 3 can be rewritten as $\mathcal{F}(u_i, \mathbf{x}) = \prod_{W\in\mathcal{O}\setminus\{U,V\}} \rho(W, u_i, \mathbf{x}) \times \rho(V, u_i,\ \mathbf{x}) \times p_{u_i}\ \mathbf{x}$ where $\rho(W, u_i,\ \mathbf{x}) = 1 - P_{W,u_i} + P_{W,u_i}\ \mathbf{x}$ and $\rho(V, u_i, \mathbf{x}) = 1 - P_{V,u_i} + P_{V,u_i}\ \mathbf{x}$. Recall that P_{W,u_i} and P_{V,u_i} is the probability $Pr(W_f > f(u_i))$ and $Pr(V_f > f(u_i))$ respectively. Similarly, we have $\mathcal{F}(v_i,\ \mathbf{x}) = \prod_{W\in\mathcal{O}\setminus\{U,V\}} \rho(W, v_i, \mathbf{x}) \times \rho(U, v_i, \mathbf{x}) \times p_{v_i}\ \mathbf{x}$. Because $f(u_i) \ge f(v_i)$, we have $Pr(W_f > f(u_i)) \le Pr(W_f > f(v_i))$ for any object $W \in \mathcal{O}\setminus\{U, V\}$, i.e., $P_{W,u_i} \le P_{W,v_i}$. Moreover, as $p_{u_j} = p_{v_j}$ and $f(u_j) \ge f(v_j)$ $(1 \le i \le m)$, for any instance v_j with $f(v_j) > f(u_i)$, we have $f(u_j) > f(v_i)$ since $f(u_i) \ge f(v_i)$. This implies that we have $Pr(V_f > f(u_i)) \le Pr(U_f > f(v_i))$, i.e.,$P_{V,u_i} \le P_{U,v_i}$. Therefore, together with the fact that $p_{u_i} = p_{v_i}$, we have $\sum_{1\le j\le q} c_j(U) \ge \sum_{1\le j\le q} c_j(V)$ for any $1 \le q \le n$, where $c_q(U)$ and $c_q(V)$ denote the coefficient of $\mathbf{x}^q$ regarding $\mathcal{F}(u_i,\ \mathbf{x})$ and $\mathcal{F}(v_i,\ \mathbf{x})$ respectively. On the other hand, we have $\omega(q) \ge \omega(j)$ for any $q < j$ as PRF^ω is a monotonic function. we have $\sum_{1\le q\le n} c_q(U)\omega(q) \ge \sum_{1\le q\le n} c_q(V)\omega(q)$ where n is the total number of objects. Then we have $\Upsilon(u_i) \ge \Upsilon(v_i)$ according to Equation 4. Consequently, the theorem holds since $\Upsilon(U) = \sum_{1\le i\le m} \Upsilon(u_i)$ and $\Upsilon(V) = \sum_{1\le i\le m} \Upsilon(v_i)$ according to Equation 5. □

Transitivity Property. The following theorem indicates that *matching dominance* operator has the *transitivity* property.

Theorem 2. *For any three objects* U*,* V *and* W *in a set of objects* $\mathcal{O}$*, we have that* $U \prec_M V$ *and* $V \prec_M W$ *implies* $U \prec_M W$*.*

PROOF. Since $U \prec_M V$ and $V \prec_M W$, there are two dominance matches $M_{U \prec V}$ and $M_{V \prec W}$ with $P(M_{U \prec V}) = 1$ and $P(M_{V \prec W}) = 1$. For a tuple $t \in M_{U,V}$, it can be split into t_1 and t_2 with $t_1.u = t_2.u = t.u$, $t_1.v = t_2.v = t.v$ and $t_1.p + t_2.p = t.p$. Similarly, tuples in $M_{V \prec W}$ can be split as well, and the operation can be conducted recursively. Since $\sum_{t \in M_{U \prec V}} t.p = \sum_{t \in M_{V \prec W}} t.p = 1$, we can recursively split tuples in $M_{U,V}$ and $M_{V \prec W}$, which results in two new dominance matches $M^*_{U \prec V}$ and $M^*_{V \prec W}$ with the same number of tuples m. Moreover, we have $t^1_i.v = t^2_i.v$ and $t^1_i.p = t^2_i.p$ for $t^1_i \in M^*_{U \prec V}$ and $t^2_i \in M^*_{V \prec W}$ where $1 \leq i \leq m$. Then we can construct a dominance match $M_{U \prec W}$ with m tuples where $t_i.u = t^1_i.u$, $t_i.w = t^2_i.w$ and $t_i.p = t^1_i.p$ for $t^1_i \in M^*_{U \prec V}$ and $t^2_i \in M^*_{V \prec W}$. Since $t^1_i.u \prec t^1_i.v$, $t^1_i.u = t^2_i.u$ and $t^2_i.v \prec t^2_i.w$ for $1 \leq i \leq m$, we have $t_i.u \prec t_i.w$ for all $t_i \in M_{U \prec W}$. Consequently, $M_{U \prec W}$ is a valid dominance match and $P(M_{U \prec W}) = 1$. Therefore, $U \prec_M W$ holds. □

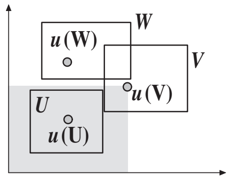


Figure 5: Pruning

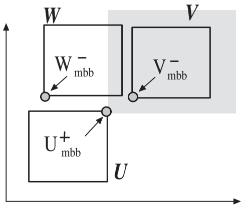


Figure 6: Validate

With similar rationale, it is trivial that *matching dominance* operator also satisfies the *irreflexive* property (i.e., $U \not\prec_M U$ for any object U) and *asymmetric* property (i.e., $U \prec_M V$ implies $V \not\prec_M U$). Therefore, *matching dominance* operator is a *strict partial order* on a set of objects.

Pruning Rule. We first define the *mean* of an object.

DEFINITION 5 (MEAN). *The mean of an uncertain object U, denoted by $\mu(U)$, is a point with $\mu(U).D_i = \sum_{u \in U} u.D_i \times p_u$. Recall that $q.D_i$ denotes the i-th dimensional coordinate value of a point q.*

The following theorem indicates that we can safely claim an object cannot *matching dominate* another object simply based on their *mean* information.

THEOREM 3. *Given two objects U and V, we have $U \not\prec_M V$ if $\mu(U) \not\prec \mu(V)$.*

PROOF. We show that $U \prec_M V$ implies $\mu(U) \prec \mu(V)$. Given $U \prec_M V$, we have a dominance match $M_{U \prec V}$ with $P(M_{U \prec V}) = 1$. For each tuple $t \in M_{U \prec V}$ we have $t.u \prec t.v$. Then, we have $\sum_{t \in M_{U \prec V}} t.u.D_i \times t_p \leq \sum_{t \in M_{U \prec V}} t.v.D_i \times t_p$ for any dimension i. This implies that $\mu(U) \prec \mu(V)$ or $\mu(U) = \mu(V)$ according to Definition 5 and the fact that $\sum_{t \in M_{U \prec V}} t_p = 1$. Based on the fact that, for any two points p and q, $\sum_{1 \leq i \leq d} p.D_i < \sum_{1 \leq i \leq d} q.D_i$ if $p \prec q$, we have $\mu(U) \neq \mu(V)$. Therefore, the theorem holds. □

EXAMPLE 6. *In Figure 5, we have $W \not\prec_M V$ according to Theorem 3 since $\mu(W) \not\prec \mu(V)$, i.e., $\mu(W)$ is not in the shaded region.*

Validation Rule. Suppose the minimal bounding boxes (MBBs) of the uncertain objects are available, we can immediately claim that an object *matching dominates* another one simply based on their MBBs.

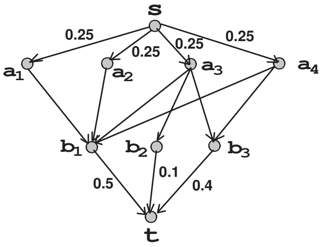


Figure 7: The network constructed based on Figure 4(a)

THEOREM 4. *Given two objects U and V, we have $U \prec_M V$ if $U^+_{mbb} \prec V^-_{mbb}$.*

PROOF. Since $U^-_{mbb} \preceq p$ and $p \preceq U^+_{mbb}$ for an object U and any point $p \in U_{mbb}$, we have $u \prec v$ for any $u \in U$ and $v \in V$ if $U^+_{mbb} \prec V^-_{mbb}$; that is, U *fully dominates* V as mentioned in Section 1. Therefore, we can easily come up with a dominance match $M_{U \prec V}$ with $P(M_{U \prec V}) = 1$. □

EXAMPLE 7. *In Figure 6, we have $U \prec_M V$ according to Theorem 4 because $U^+_{mbb} \prec V^-_{mbb}$, i.e., V^-_{mbb} is in the shaded region.*

3.3 Matching Dominance Verification

In this subsection, we show that the calculation of the matching dominance probability can be converted to the max-flow problem.

The max-flow problem is to find a feasible flow through a single-source, single-sink flow network that is maximum. Let N (E) denote a set of vertices (edges) and $G =< N, E >$ be a network with s, $t \in N$ being *source* and *sink* respectively. The capacity of an edge $<u, v>$ is the maximum amount of flow that can pass through $<u, v>$, denoted by $c_{u,v}$. A feasible flow g of $G_{U,V}$ maps each edge $<u, v>$ to a non-negative value $g_{u,v}$ following two constraints: (i) $g_{u,v} \leq c_{u,v}$ for each edge $<u, v> \in E$ where $g_{u,v}$ is the amount of flow through edge $<u, v>$ (**capacity constraint**); (ii) for each node $u \in N \setminus \{s, t\}$, $\sum_{<v,u> \in E} g_{v,u} = \sum_{<u,v> \in E} g_{u,v}$ (**conservation of flows**). The max-flow problem is to maximize $|g|$ where $|g| = \sum_{<s,v> \in E} g_{s,v}$.

Same as [21], we construct a network $G_{U,V}$ regarding two objects U and V as follows.

- Vertices s and t for source and sink respectively.
- Each instance $u \in U$ contributes a vertex u and an edge $<s, u>$ with $c_{s,u} = p_u$.
- Each instance $v \in V$ contributes a vertex v and an edge $<v, t>$ with $c_{v,t} = p_v$.
- If $u \prec v$ for two instances $u \in U$ and $v \in V$, there is an edge $<u, v>$ with $c_{u,v} = \infty$.

EXAMPLE 8. *In Figure 7, we show the network constructed based on two objects A and B in Figure 4(a). Note that all instances from A have the occurrence probability 0.25, and $p_{b_1} = 0.5$, $p_{b_2} = 0.1$ and $p_{b_3} = 0.4$. We label the capacities of the edges which start from s or end up at t. All other edges unlabeled have the capacity ∞.*

Following theorem indicates that the problem of calculating the *matching dominance probability* can be converted to the max-flow problem.

Theorem 5. *Let $G_{U,V}$ be the network constructed based on two object U and V, we have $Pr(U \prec_M V) = |g^*|$ where g^* is a maximum flow in $G_{U,V}$.*

Proof. For presentation simplicity, we use E_m to denote the edges $<u, v> \in E$ with $u \in U$ and $v \in V$. And the edges $<s, u>$ for $u \in U$ belong to E_s, and edges $<v, t>$ for $v \in V$ are assigned to E_t.

We first show that $Pr(U \prec_M V) \leq |g^*|$. Let $M_{U \prec V}$ be the dominance match with $P(M_{U \prec V}) = Pr(U \prec_M V)$, we have $\sum_{t \in M_{U \prec V}} t.p = Pr(U \prec_M V)$. Then we construct a flow g as follows. For each tuple $t \in M_{U \prec V}$, we set $g_{t.u,t.v} = t.p$. While $g_{s,u}$ is set to $\sum_{t \in M_{U \prec V} \wedge t.u=u} t_p$, and $g_{v,t}$ is set to $\sum_{t \in M_{U \prec V} \wedge t.v=v} t_p$. Clearly, all vertices except s and t satisfy the conservation constraint. The edges $<u, v> \in E_m$ always meet the capacity constraint. Since $\sum_{t \in M_{U \prec V} \wedge t.u=u} t.p \leq p_u$ according to Definition 2, all edges from E_s satisfy the capacity constraint. With the same argument, all edges from E_t also meet the capacity constraint. Therefore, g is a feasible flow on the network $G_{U,V}$, and hence $Pr(U \prec_M V) = \sum_{t \in M_{U \prec V}} t.p = |g| \leq |g^*|$.

Now we prove that $|g^*| \leq Pr(U \prec_M V)$. Let g^* be a maximum flow of $G_{U,V}$, we construct a dominance match $M_{U \prec V}$ as follows. For each edge $<u, v> \in E_m$, we build a tuple $<u$, v, $g_{u,v}>$. With similar rationale to the above proof, we can show that $M_{U \prec V}$ is a valid dominance match for U and V since g^* satisfies the capacity and conservation constraints. Therefore, we have $|g^*| = P(M_{U \prec V}) \leq Pr(U \prec_M V)$. □

Based on Theorem 5 and Definition 4, the following theorem is immediate.

Theorem 6. *Let $G_{U,V}$ be the network constructed based on two object U and V, we have $U \prec_M V$ if and only if $|g^*| = 1$ where g^* is a maximum flow of $G_{U,V}$.*

3.4 Compared with Usual Stochastic Order

As shown in [21], the dominance verification of the usual stochastic order can also be mapped to the problem of max-flow with the same network structure, the following theorem is immediate.

Theorem 7. *Given two objects U and V, U matching dominates V if and only if U dominates V regarding the usual stochastic order.*

In [21], the network $G_{U,V}$ is constructed with the time complexity $O(m^2)$ in the worst case where m is the average number of instances for each object, the state-of-art techniques for max-flow problem can be applied to check *matching dominance* operator for two objects U and V. In the implementation, the open source code from [9] is employed and the time complexity of the algorithm is $O(mc \log(c))$ where m and c are the number of vertices (i.e., the number of instances in U and V) and edges in the network $G_{U,V}$.

Although the two dominance operators are equivalent to each other, the *matching dominance* operator has the following two advantages. Firstly, the definition of *matching dominance* is more simple [2] and intuitive. It is unknown if we can come up with the proof of the monotonic property of the usual stochastic order without the help of *matching dominance* operator. Secondly, as discussed in Section 5.2, the *matching dominance* probability (i.e., $Pr(U \prec_M V)$) can naturally capture the extent of the dominance between two objects, and hence can be used to reduce the size of the skyline.

4. MDB INDEXING TECHNIQUE

In this section, we introduce the Matching-Dominance based Band (**MDB** for short) indexing technique based on the *matching dominance* operator, followed by an efficient index computation algorithm.

4.1 Definition of MDB Index

We formally define the MDB index based on the *matching dominance* operator proposed in Section 3.

Definition 6 (**MDB** Index). *Given a set $\mathcal{O}$ of uncertain objects, the MDB index of $\mathcal{O}$, denoted by $\mathcal{L}(\mathcal{O})$, is a layer indexing structure where an object U is kept on the i-th layer of $\mathcal{L}(\mathcal{O})$, denoted by $\mathcal{L}_i(\mathcal{O})$, if it is matching dominated by $i - 1$ other objects in $\mathcal{O}$.*

Theorem below indicates that we can exclude the objects which do not reside on the first k layers from the top k candidate objects for any monotonic *parameterized ranking* functions. The correctness of Theorem 8 is immediate based on Theorem 1 since we have k objects $\{U\}$ with $\Upsilon(U) \geq \Upsilon(V)$ if an object V is *matching dominated* by at least k other objects in $\mathcal{O}$.

Theorem 8. *For any monotonic parameterized ranking function PRF^{ω}, we can exclude an object from the top k candidates if it is matching dominated by k other objects.*

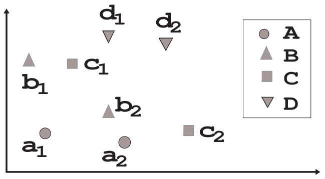


Figure 8: MDB Index Example.

Example 9. *In Figure 8, suppose $\mathcal{O} = \{A, B, C, D\}$ and each object has two instances with the same the occurrence probability* 0.5. *Since $A \prec_M C$, $A \prec_M D$ and $B \prec_M D$, we have $\mathcal{L}_1(\mathcal{O}) = \{A, B\}$, $\mathcal{L}_2(\mathcal{O}) = \{C\}$ and $\mathcal{L}_3(\mathcal{O}) = \{D\}$.*

Let K denote the maximum k value used in top k queries, in practice K is much smaller than the number of objects. Therefore, we only keep the first K layers of $\mathcal{L}(\mathcal{O})$ in our implementation. As shown in the empirical study, MDB index can significantly reduce the number of top k candidates for the top k queries, especially when k is small.

4.2 MDB Algorithm

In this subsection, we introduce an efficient MDB computation algorithm based on the filtering and verification techniques introduced in Section 3. We assume the MBBs of the objects are organized by an R-Tree. We also keep *mean* information for each entry. Specifically, $\mu(e)$ is the *mean* of an entry where $\mu(e).D_i = \min(\mu(U).D_i)$ for all objects $\{U\}$ covered by the entry e. Note that a data entry

[2] The definition of the usual stochastic order involves probabilistic computation against a infinite number of lower sets.

e corresponds to an object. We use U_{dom} (e_{dom}) to record the number of times U (e) is *matching dominated* by other objects.

```
Algorithm 1: MDB Computation($R$)
   Input : $R$ ( $R$-Tree for $\mathcal{O}$ )
   Output: $\mathcal{L}(\mathcal{O})$
1  $R_t = \emptyset$;
2  push root of $R$ into a heap $H$;
3  while $H \neq \emptyset$ do
4      $e := H.deheap()$;
5      if MD-Check ($e$, $R_t$) return false then
6          if $e$ is a data entry associated with object $U$ then
7              $\mathcal{L}_i(\mathcal{O}) \leftarrow U$, where $i = e_{dom} + 1$ ;
8              insert $\mu(U)$ into $R_t$;
9          else
10             Push all child entries of $e$ into $H$;
11 return $\mathcal{L}(\mathcal{O})$
```

Algorithm 1 outlines our implementation of MDB computation. Same as the existing skyline and skyband computation algorithms [13, 21, 15], Algorithm 1 follows the *branch-and-bound* searching paradigm. Let $SUM(p)$ denote the summation of all coordinate values of a point p, i.e., $SUM(p) = \sum_{i=1}^{d} p.D_i$, a min heap H is used in Algorithm 1 to maintain the entries to be visited and the key of each entry is $SUM(\mu(e))$. For each entry e popped from the heap H (Line 4), the procedure **MD-Check** conducts *matching dominance* verification on the objects within e (Line 5). It will return true if all objects within e (intermediate entry) or the object associated with e (data entry) are *matching dominated* by more than $K - 1$ other objects. Details of **MD-Check** will be introduced in Algorithm 2. If an entry e survives the dominance check and e corresponds to an object U, Line 7 assigns U to its corresponding layer in $\mathcal{L}(\mathcal{O})$ according to its *matching dominance* count U_{dom} (i.e.,e_{dom}). Note that, according to Theorem 3 and the definition of $\mu(e)$, an object U cannot be *matching dominated* by any objects within an entry e with $SUM(\mu(e)) \geq SUM(\mu(U))$, and hence U_{dom} will not increase once U is processed. Meanwhile, we also insert the point $\mu(U)$ into an in-memory R-Tree R_t at Line 8, which is used for *pruning* purpose in Algorithm 2. If e is an intermediate entry which survives the dominance check, we need to expand e in Line 10 and put its child entries into the heap H for further processing. The algorithm terminates when the heap H is empty, and $\mathcal{L}(\mathcal{O})$ is constructed.

How to efficiently calculate the *matching dominance* count of an object is the key issue in Algorithm 1. In Algorithm 2, we illustrate the detailed implementation of **MD-Check**. The *validation* rule (Theorem 4 in Section 3.2) can be immediately extended to intermediate entries; that is, we have $U \prec_M V$ for all objects $\{V\}$ in the entry e if $U^+_{mbb} \prec e^-_{mbb}$. Therefore, we do not need to access objects within e if the lower corner of its MBB (i.e., e^-_{mbb}) is dominated by the upper corners of K other objects in R_t (Lines 1-5). Lines 7-16 conduct dominance check against the object U if e is a data entry and V is the object associated with e. At Line 8, we apply the *pruning* rule (Theorem 3 in Section 3.2) to retrieve the objects which may *matching dominate* objects in e. For each object U in candidate objects set $\mathcal{C}$, Line 10 claims $U \prec_M V$ if $U^+_{mbb} \prec V^-_{mbb}$ (*validation* rule). Otherwise, Line 13 will conduct the max-flow computation based dominance check proposed in Section 3.3.

```
Algorithm 2: MD-Check( $e$, $R_t$ )
   Input : $e$: an intermediate or data entry of $R$-Tree
           $R_t$: $R$-Tree for survived objects seen so far
   Output: if objects in $e$ are matching dominated by
           more than $K-1$ objects
1  if $e$ is an intermediate entry then
2      if there are $K$ objects $\{U\}$ in $R_t$ such that
       $U^+_{mbb} \prec e^-_{mbb}$ (validation rule) then
3          return true
4      else
5          return false
6  else
7      $V \leftarrow$ the object associated the with $e$ ;
8      $\mathcal{C} \leftarrow$ objects $U \in R_t$ with $\mu(U) \prec \mu(V)$ (pruning
       rule) ;
9      for any object $U$ in $\mathcal{C}$ do
10         if $U^+_{mbb} \prec e^-_{mbb}$ (validation rule) then
11             $e_{dom} := e_{dom} + 1$ ;
12         else
13             if $U \prec_M V$ then
14                 $e_{dom} := e_{dom} + 1$ ;
15         if $e_{dom} \geq K$ then
16             return true
17     return false
```

Utilizing Transitivity Property. We can further speed up the computation by utilizing the transitivity property of the *matching dominance* operator (Theorem 2 in Section 3.2). Besides the dominance count, we also keep a dominance list for each object U in $\mathcal{L}(\mathcal{O})$, which keeps the objects $\{W\}$ with $W \prec_M U$. According to the transitivity property of *matching dominance*, we have $W \prec_M V$ if $U \prec_M V$. This implies that we can avoid the dominance verification for W and V if $U \prec_M V$, and hence significantly reduce the computational cost especially when K is large. It is interesting to investigate the access order of objects in $\mathcal{C}$ at Line 9. In the implementation, we calculate the *manhattan distance* between $\mu(U)$ and $\mu(V)$ and the object U with the smallest distance will be accessed first. This is called MD access order in this paper, and the empirical study shows this method is useful since it can increase the chance of applying *transitivity* property.

MDB Index Maintenance. We can dynamically maintain the MDB index based on the techniques in Algorithm 2. Specifically, when a new object U arrives, the dominance count of U can be calculated by invoking **MD-Check**(U, R_t). According to Theorem 3, only objects $\{V\}$ with $\mu(V) \prec \mu(U)$ (e.g., objects with *means* located in R_1 in Figure 9) may increase U_{dom}. Meanwhile, the dominance count of some objects may be increased by one due to the arrival of U. Only objects $\{W\}$ with $\mu(U) \prec \mu(W)$ (e.g., objects with *means* located in R_2 in Figure 9) need to be checked. The object U and other objects with updated dominance counts will be put to their corresponding layers. The *deletion* of an object can be processed in a similar way. Note that we only need to decrease the dominance counts of the objects which are *matching dominated* by U, and then update $\mathcal{L}(\mathcal{O})$.

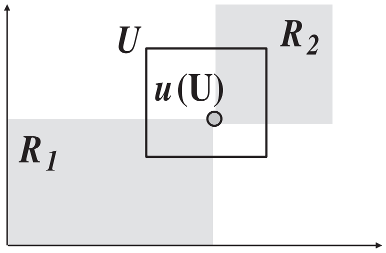


Figure 9: Update

5. EXTENSIONS

This section discusses the possible extensions of our *matching dominance* operator proposed in the paper.

5.1 Continuous PDFs

In some applications, the uncertainty of the data might be described by continuous probabilistic density functions (PDFs). We can still check the *matching dominance* operator by partitioning the PDF into a set of instances where each instance u is associated with a hyper-cube u_{mbb}, instead of a point in the discrete case. To verify if $U \prec_M V$ for two objects U and V, we can construct a network $G_{U,V}$ in the same way with *discrete* case (Section 3.3) except that an instance u cannot be mapped to another instance v unless u_{mbb} *fully dominates* v_{mbb}, i.e., $u^+_{mbb} \prec v^-_{mbb}$. Clearly, we can immediately claim that $U \prec_M V$ if a maximum flow g^* with $|g^*| = 1$ is detected. Nevertheless, we cannot claim $U \not\prec_M V$ since we may come up with a network $G'_{U,V}$ with $|g^*| = 1$ if we further partition the instances. In practice, we can discretize the distributions to an approximate level of granularity, and claim that $U \not\prec_M V$ if we cannot find a maximum flow g^* with $|g^*| = 1$. This may push some objects to higher layers and hence leads to a larger candidate size in the top k query, but we will not miss any *true* top k objects.

5.2 Probabilistic Matching Dominance based Skyline

In some scenarios, instead of enforcing the *matching dominance* (i.e., the usual stochastic order) users may relax the dominance condition to reduce the number of skyline objects retrieved for a concise result; that is, we may claim an object U dominates another object V if U is *likely* to be ranked higher than V in most of the functions in $\mathcal{F}$. It is very nature to use the *matching dominance* probability to control the extent of the dominance. Given a probabilistic threshold θ with $0.5 < \theta \leq 1$, we say an object U θ-dominates V if $Pr(U \prec_M V) \geq \theta$. The θ-skyline is a set of objects which are not θ-dominated by any other objects, which corresponds to the stochastic skyline in [21] when $\theta = 1$.

Following theorem is immediate based on the above definition.

THEOREM 9. *Let $\mathcal{S}_\theta(\mathcal{O})$ denote the θ-skyline of a set $\mathcal{O}$ of objects, for any $\theta_1 < \theta_2$ we have $\mathcal{S}_{\theta_1}(\mathcal{O}) \subseteq \mathcal{S}_{\theta_2}(\mathcal{O})$.*

Theorem 9 indicates that users can make the trade-off between the size of θ-skyline objects and the strength of the dominance condition by tuning the θ value (i.e., *matching dominance* probability).

6. PERFORMANCE EVALUATION

We present results of a comprehensive performance study to evaluate the efficiency and effectiveness of the proposed techniques in this paper.

Algorithms. We implement and evaluate the following techniques for MDB construction.

- **MDB**: the MDB computation algorithm proposed in Section 4.
- **MDB-NF**: MDB **without** *pruning* rule (Theorem 3 in Section 3.2).
- **MDB-NT**: MDB **without** utilizing *transitivity* property (Theorem 2 in Section 3.2).
- **MDB-NO**: MDB **without** utilizing Manhattan Distance (MD) access order, i.e., objects are randomly chosen from $\mathcal{C}$ at Line 9 of Algorithm 2.

REMARK 1. *Note that, since the matching dominance operator is equivalent to the usual stochastic order, the stochastic skyline computation techniques in [21] can be immediately used for the skyline computation against matching dominance operator. In the experiments, we focus on the MDB technique which can be regarded as a general case of skyline computation.*

REMARK 2. *As discussed in [21] and Section 1, the P-domination [1] and probabilistic dominance [16, 2] do not have the monotonic property for the uncertain objects with multiple instances, and hence they cannot be used as the dominance operator for MDB.*

Datasets. Three real datasets, HOUSE, CA and USA, are employed to represent the centers of the uncertain objects. There are 127,932 3-dimensional points (records) in dataset HOUSE which is available at `http://www.ipums.org`, and each record represents the percentage of an American family's annual income spent on 3 types of expenditures. CA and USA are 2-dimension spatial datasets representing $62K$ and $221K$ locations in California and United States respectively, which are available at `http://www.census.gov/geo/www/tiger`. By using methodologies in [3], we also generate synthetic data for centers of uncertain objects following the *anti-correlated* (A for short), *correlated* (C) or *independent* (E) distribution. We use **independent** (E) as the default distribution for objects' centers. The dimensionality of the synthetic data varies from 2 to 5 with default value 3. And the number of uncertain objects grows from 20K to 100K with default value 100K. All dimensions are normalized to domain $[0, 10000]$.

The minimal bounding box (MBB) of an uncertain object is a hyper-cube with expected length h varying from 50 to 400 with default value 200. For a given h, the lengths of the objects are randomly chosen between 0 and $2 \times h$. Suppose the instance of an object is described by m instances which follow two popular distributions $Normal$ (N for short) and $Uniform$ (U), where the expected m varies from 20 to 100 with default value 40. Note that, the total number of instances in the default dataset is $100K \times 40 =$ **4 million**, and

Notation	Definition (Default Values)
h	avg. MBB length (200)
n	the number of objects ($100K$)
m	avg. number of instances of each object(40)
d	dimensionality (3)
K	maximal number of layers in $\mathcal{L}(\mathcal{O})$ (40)

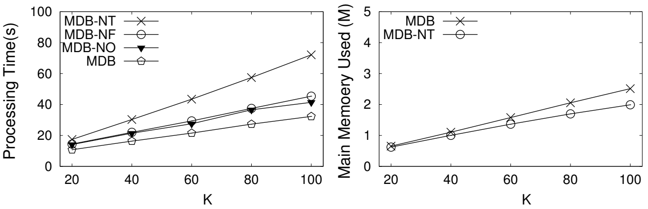
Table 2: System Parameters

it reaches **10 million** when the number of instances is set to 100. Given m, the number of instances for each object uniformly distributes between 0 and $2m$, and instances in the same object have the same occurrence probability. The $Normal$ distribution serves as default instance distribution with standard deviation $\frac{h}{2}$.

All algorithms proposed in the paper are implemented in standard C++ with STL library support and compiled with GNU GCC. Experiments are conducted on a PC with Intel Xeon 2.4GHz dual CPU and 4G memory under Debian Linux. In our implementation, MBBs of the uncertain objects are indexed by an R-Tree with page size 4096 bytes where the *mean* values of the R-Tree entries are also kept. The instances of an object are kept by an individual file. The number of layers in the MDB index ($\mathcal{L}(\mathcal{O})$) ranges from 20 to 100 with default value 40.

Table 2 lists parameters which may potentially have an impact on our performance study. In the experiments, all parameters use default values unless otherwise specified.

6.1 Evaluate MDB Computation



(a) *Construction Time* (b) Memory Usage

Figure 10: Effectiveness and Memory Usage

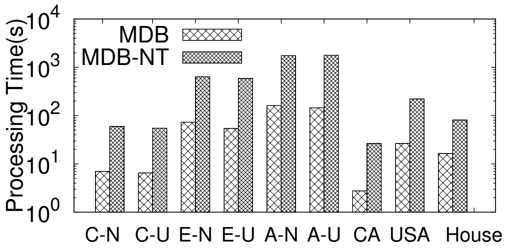
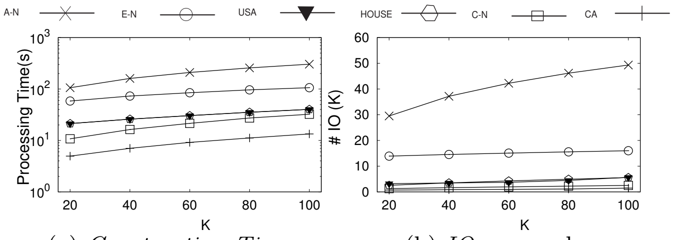


Figure 11: Different Datasets

In the first set of experiments, we evaluate the effectiveness of the *pruning* rule (Theorem 3), transitivity property (Theorem 2) and MD order[3]. Figure 10(a) illustrates the processing time of MDB, MDB-NF, MDB-NT and MDB-NO on HOUSE dataset where K varies from 20 to 100. It is reported that three techniques (pruning rule, transitivity property and MD access order) make remarkable contributions to the index construction efficiency, especially the transitivity property. Figure 10(b) reports the use of main memory in MDB and MDB-NT. As expected, the memory space used by MDB grows faster than that of MDB-NT because we need to maintain dominance lists for objects in $\mathcal{L}(\mathcal{O})$ where the sizes of the dominance lists may grow linearly with K. Nevertheless, the trend is not obvious since we only keep the identifications of the objects and there are at most K elements in each list. On the other hand, as shown in Figure 10(a), the gain of the transitivity property is significant especially when K is large.



(a) *Construction Time* (b) *IO* accessed

Figure 12: The effect of K

We evaluate the performance of MDB and MDB-NT against datasets C-N, C-U, E-N, E-U, A-N, A-U, CA, USA and HOUSE in Figure 12 where C-N denotes the 3 dimensional synthetic data whose centers and instances follow the $\underline{\text{C}}$orrelated and $\underline{\text{N}}$ormal distributions respectively, and similar definitions go to C-U, E-N, E-U, A-N and A-U. It is observed that the distribution of the instances (N and U) does not noticeably affect performance of the algorithms. On the other side, both algorithms are very sensitive to the distribution of the object centers because, as reported in experiments in Section 6.2, the distributions of the centers have a great impact on the size of $\mathcal{L}(\mathcal{O})$. As expected, MDB always outperforms MDB-NT since the former can take advantage of the *transitivity property*. Consequently, we only evaluate the performance of MDB in the following experiments.

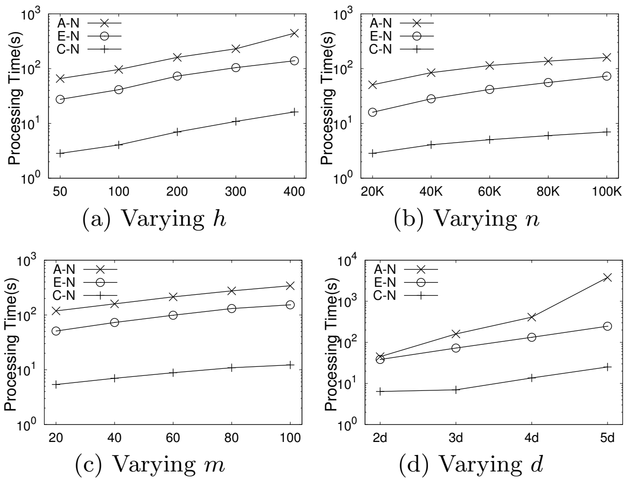
Figure 12 investigates the impact of K (i.e., the number of layers constructed for $\mathcal{L}(\mathcal{O})$) on the performance of MDB where datasets C-N, E-N, A-N, CA, USA and HOUSE are evaluated. The construction time and I/O costs are reported in Figure 12(a) and 12(b) respectively. As expected, the performance of MDB degrades against the growth of K since the number of objects in MDB increases.

We further investigate the impact of the edge length (h), the number of objects (n), the number of instances (m) and the dimensionality (d) against the performance of MDB where three datasets A-N, E-N and C-N are deployed. Figure 13 shows that the processing time of MDB increases with the growth of h, n, m and d, and dimensionality (d) has the greatest impact compared with other parameters.

6.2 Evaluate Effectiveness of MDB

In this subsection we evaluate the query performance of MDB by reporting the number of top k candidate objects (i.e., $\bigcup_{1\leq i\leq k}\mathcal{L}_i(\mathcal{O})$) and k varies from 1 to 100 with default value 40. Besides the *matching dominance* based layer index, we also evaluate another layer index, namely **FDB**,

[3] The advantage of *validation* rule (Theorem 4) is obvious and hence we do not evaluate the MDB without the *validation* rule.



(a) Varying h (b) Varying n

(c) Varying m (d) Varying d

Figure 13: Impact of Diff. Parameters

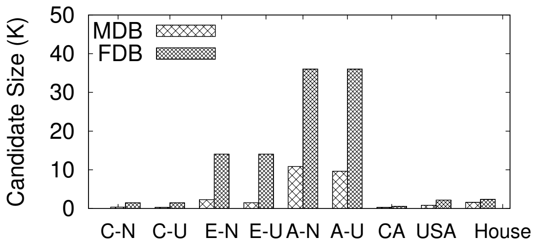
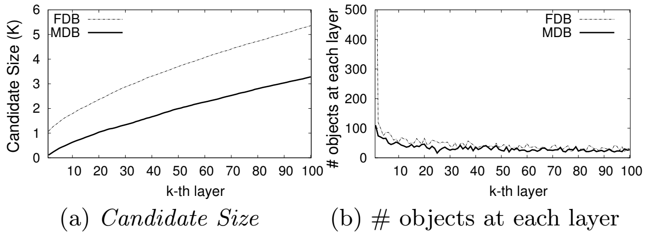


Figure 14: Different Datasets

based on the *fully dominance* operator described in Section 1.

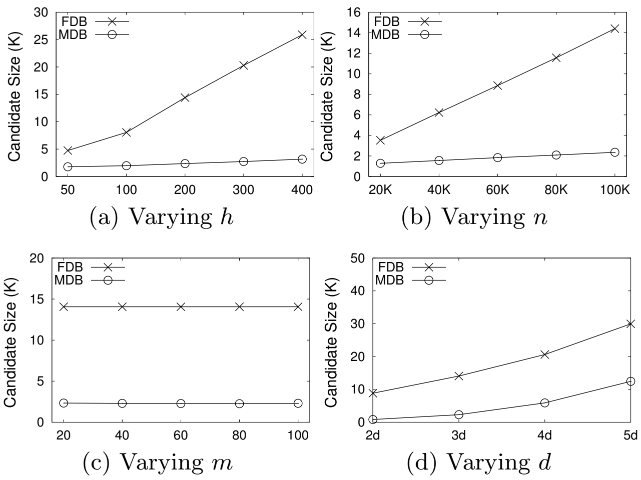
Figure 14 evaluates the effectiveness of MDB and FDB indexing techniques against dataset C-N, C-U, E-N, E-U, A-N, A-U, CA, USA and HOUSE, by measuring the number of candidate objects for top k query with $k = 40$. It is shown that MDB is much more effective than FDB since the candidate size of MDB is significantly smaller than that of FDB on all datasets. This is because the condition of *full dominance* is strict compared with *matching dominance*. Moreover, the candidate sizes of both index techniques are sensitive to the center distributions, where the worst performance comes from the *anti-correlated* data.



(a) *Candidate Size* (b) # objects at each layer

Figure 15: The effect of k

We evaluate the impact of k on HOUSE dataset where k varies from 1 to 100. Figure 15 reports the number of candidate objects for top k query (i.e., the size of $\bigcup_{1\leq i\leq k}\mathcal{L}_i(\mathcal{O})$) as well as the number of objects at the k-th layer (i.e., the size of $\mathcal{L}_k(\mathcal{O})$). As expected, Figure 15(a) shows that the number of candidates increases against k. Nevertheless, the growth rate gradually slows down when k increases because, as shown in Figure 15(b), the size of $\mathcal{L}_i(\mathcal{O})$ decreases against k. Similar trend is observed for FDB index.

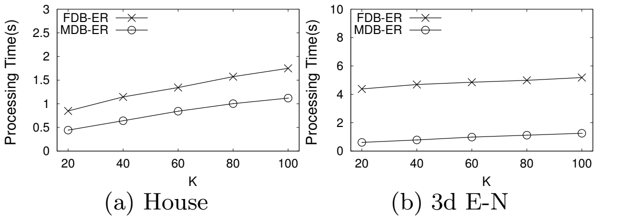


(a) Varying h (b) Varying n

(c) Varying m (d) Varying d

Figure 16: Impact of Diff. Parameters

Figure 16 investigates the impact of the edge length (h), the number of objects (n), the number of instances (m) and the dimensionality (d) against the performance of MDB and FDB where the default dataset (3 dimensional E-N) is deployed. It is shown that both methods are sensitive to the dimensionality (Figure 16(d)), and the number of instances (m) does not have noticeably affect (Figure 16(c)) on the size of both indexes. Figure 16(a) and Figure 16(b) report that MDB is much more scalable than FDB towards the growth of h and n. MDB beats FDB by a huge margin when h and n is large.



(a) House (b) 3d E-N

Figure 17: Top k Query Response Time

Although the top k candidate size and construction time which are independent of the specific *scoring* and *parameterized ranking* functions, we also implement a specific *parameterized ranking* function to show that the layer indexing based on *matching dominance* operator can significantly improve the performance. Specifically, we use PRF^{ω} with $\omega(i) = -i$. This corresponds to the *expected rank* based ranking approach [6], named **ER**, which is a simple and popular ranking approach. As to the *scoring* function, we adapt the *linear* function $f(p) = \sum_{i=1}^{d} -a_i \times p.D_i$ where a_i is randomly chosen from $[0, 1]$. In the last set of experiments, we evaluate the performance of **ER** algorithm (the *expected rank* based algorithm [6]) combined with MDB and FDB indexing techniques, denoted by MDB-ER and FDB-ER respectively. Figure 17(a) and Figure 17(b) report the average query response time of $1,000$ top k queries against HOUSE dataset and the default dataset (3 dimensional E-N) respectively, which further confirms the effectiveness of *matching dominance* operator since less objects are loaded for the rank computation.

7. RELATED WORK

In this section, we introduce the existing works which are closely related to the paper.

7.1 Layer indexing technique

Top k query processing is a very active research area and various novel techniques have been developed. Many efficient index techniques have been proposed to pre-compute and organize the candidate set based on the geometric properties of the convex and dominance relation, such as *Onion* [4] and Skyband [15, 14, 23]. However, these layer indexing techniques cannot be directly applied to top k query on uncertain objects due to the inherent difference between certain and uncertain objects.

7.2 Ranking uncertain Objects

In recent years, the inherent uncertainty of data in many applications leads to the emergence of many uncertain database models ([7, 19, 17]) most of which are developed based on either tuple-level or attribute-level uncertainty. In this paper, we model the uncertain data on attribute-level. A large amount of work has been dedicated to top-k queries with different semantics such as U-top$k$[20], U-$k$ranks [20], PT-k [10], Global-top k [22] rank, *expected rank* [6], c-Typical-Top$k$ [8], and *parameterized ranking* function based top k [11]. Particularly, as shown in [11] the *parameterized ranking* function can unify other popular ranking semantics. The ranking of multi-dimensional uncertain objects for a given *scoring* function is studied in [12] and the U-top$k$ [20] semantics is adapted for ranking.

7.3 Skyline Computation On uncertain Objects

Probabilistic skyline on uncertain data is first tackled in [16, 2]. Efficient techniques are proposed following the bounding-pruning-refining framework. Recently, the P-*domination* is proposed by Bartolini *et al.* [1] to capture the dominance among uncertain objects. However, it is defined base on the tuple-level uncertain object model and the *ranking* function is the *expected rank* [6] which is a special case of *parameterized ranking* function. The stochastic order based skyline computation for multi-dimensional uncertain objects is investigated in [13, 21]. The stochastic order can define the dominance relation between two uncertain objects with multiple instances, but objects are ranked by their *expected scores* and existing ranking functions for uncertain objects are not considered in [13, 21].

8. CONCLUSION

The dominance operator is fundamental in multi-criteria decision making applications. Although a variety of works have studied the semantics of ranking query on uncertain objects, there is no investigation on the semantics of dominance on multi-dimensional uncertain objects regarding these ranking approaches. This paper aims to fill this gap by introducing a novel *matching dominance* operator so that the new dominance operator holds the monotonic property for any monotonic *parameterized ranking* function, which can unify the popular ranking approaches proposed in the literature. We further develop effective layer indexing technique, namely MDB index, to facilitate the top k queries on multi-dimensional uncertain objects. Our comprehensive experiments demonstrate the effectiveness and efficiency of our techniques proposed in the paper.

Acknowledgments. Ying Zhang is supported by ARC DE140100679 and DP130103245. Wenjie Zhang is supported by ARC DE120102144 and DP120104168. Xuemin Lin is supported by NSFC61232006, NSFC61021004, ARC DP120 104168 and DP140103578. Muhammad Aamir Cheema is supported by ARC DE130101002 and DP130103405.

9. REFERENCES

[1] I. Bartolini, P. Ciaccia, and M. Patella. The skyline of a probabilistic relation. *TKDE*, 2012.

[2] C. Böhm, F. Fiedler, A. Oswald, C. Plant, and B. Wackersreuther. Probabilistic skyline queries. In *CIKM*, pages 651–660, 2009.

[3] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, 2001.

[4] Y.-C. Chang, L. D. Bergman, V. Castelli, C.-S. Li, M.-L. Lo, and J. R. Smith. The onion technique: Indexing for linear optimization queries. In *SIGMOD*, 2000.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (second edition)*. 2001.

[6] G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data and expected ranks. In *ICDE*, 2009.

[7] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. In *VLDB*, pages 864–875, 2004.

[8] T. Ge, S. Zdonik, and S. Madden. Top-k queries on uncertain data: On score distribution and typical answers. In *SIGMOD*, 2009.

[9] D. S. Hochbaum. The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Operations Research*, 56(4):992–1009, 2008.

[10] M. Hua, J. Pei, W. Zhang, and X. Lin. Ranking queries on uncertain data: A probabilistic threshold approach. In *SIGMOD*, 2008.

[11] J. Li, B. Saha, and A. Deshpande. A unified approach to ranking in probabilistic databases. *VLDB J.*, 20(2), 2011.

[12] X. Lian and L. Chen. Ranked query processing in uncertain databases. *IEEE Trans. Knowl. Data Eng.*, 22(3), 2010.

[13] X. Lin, Y. Zhang, W. Zhang, and M. A. Cheema. Stochastic skyline operator. In *ICDE*, pages 721–732, 2011.

[14] K. Mouratidis, S. Bakiras, and D. Papadias. Continuous monitoring of top-k queries over sliding windows. In *SIGMOD*, 2006.

[15] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. *ACM Trans. Database Syst.*, 30(1), 2005.

[16] J. Pei, B. Jiang, X. Lin, and Y. Yuan. Probabilistic skylines on uncertain data. In *VLDB*, 2007.

[17] P. Sen, A. Deshpande, and L. Getoor. Prdb: managing and exploiting rich correlations in probabilistic databases. *VLDB J.*, 18(5), 2009.

[18] M. Shaked and J. G. Shanthikumar. *Stochastic Orders and Their Applications*. Academic Press, 1997.

[19] S. Singh, C. Mayfield, S. Mittal, S. Prabhakar, S. E. Hambrusch, and R. Shah. Orion 2.0: native support for uncertain data. In *SIGMOD Conference*, 2008.

[20] M. A. Soliman, I. F. Ilyas, and K. C. Chang. Top-k query processing in uncertain databases. In *ICDE 2007*.

[21] W. Zhang, X. Lin, Y. Zhang, M. A. Cheema, and Q. Zhang. Stochastic skylines. *TODS*, 37(2), 2012.

[22] X. Zhang and J. Chomicki. On the semantics and evaluation of top-k queries in probabilistic databases. In *DBRank*, 2008.

[23] L. Zou and L. Chen. Dominant graph: An efficient indexing structure to answer top-k queries. In *ICDE*, 2008.