PROBABILISTIC COUNTING IN UNCERTAIN SPATIAL DATABASES USING GENERATING FUNCTIONS

Andreas Züfle
Emory University
azufle@emory.edu

ABSTRACT

Many applications using spatial data require counting the number of spatial objects within a region. In cases where the locations of objects are uncertain, this count becomes random variable. This spatial gem gives an efficient solution for computing the probability mass function of this random variable, that is computing for each integer n the probability of having exactly n objects within the region.

1 Introduction

Our ability to unearth valuable knowledge from large sets of spatial data is often impaired by the uncertainty of the data which geography has been named the "the Achilles heel of GIS" [1] for many reasons: 1) Imprecision is caused by physical limitations of sensing devices and connection errors, 2) Data records may be obsolete; 3) Data can be obtained from unreliable sources, such as volunteered geographic information; and 4) Data may be deliberately obfuscated to preserve the privacy of users. These issues introduce the notion of uncertainty in the context of spatio-temporal data management. Many algorithms have been proposed in the last decade to handle different spatial query predicates (such as distance range, kNN, and distance ranking) described in various tutorials [2, 3, 4, 5] and surveys [6, 7]. Many query predicates require to count the number of uncertain objects that satisfy a give query predicated, such as being located in a query region or being closer to a query object than another object. Computing the probability mass function of such as count requires to compute for each integer n the probability of having exactly n objects satisfy the query predicate. A commonly used technique that allows many of these algorithms to run efficiently leverages the technique of Generating Functions to efficiently aggregate an exponential number of possible worlds in polynomial time. This technique is described, along with examples and implementation, in this spatial gem.

An example of an uncertain (toy) database is shown in Figure 1 having six uncertain objects $\{A, B, C, D, E, F\}$. Rather than having a single unique (crisp) location, an object in an uncertain database may have multiple alternatives, each associated with a corresponding probability of being the true location. For example, object B has five possible locations and object D has four possible locations. There exist multiple models for uncertain data, either describing uncertain objects by discrete (and finite) sets of alternatives, or by describing uncertain objects by continuous distributions of (uncountably infinite) possible locations [6]. The most prominent systems for uncertain relational data management are MayBMS [8], MystiQ [9], Trio [10], and BayesStore [11] which allow to efficiently answer traditional queries that select subsets of data based on predicates or join different datasets based on conditions. While these existing systems efficiently support simple projection-selection-join queries, they offer no support for complex queries and data mining tasks. A likely reason for this gap is the theoretic result of [12] which shows that the general problem of query processing in uncertain databases is #P-hard in the number of database objects.

While this result implies that general query processing on uncertain data is hard, it does not outrule the possibility of efficient solutions for specific query types. And in fact, many important classes of spatial queries have efficient (polynomial time) solutions, including range count queries [13], nearest neighbor queries [14, 15, 16], k-nearest neighbor queries [17, 18, 19] and (similarity-) ranking queries [20, 21, 22, 23, 24, 22, 25, 26].

All these classes of spatial queries have in common that they count the number of spatial objects that fall within a region. A range count query directly returns the distribution of the number of objects within a specified query region. To decide if an object A is a kNN of a query object Q; a kNN query computes that probability that less than k objects are closer

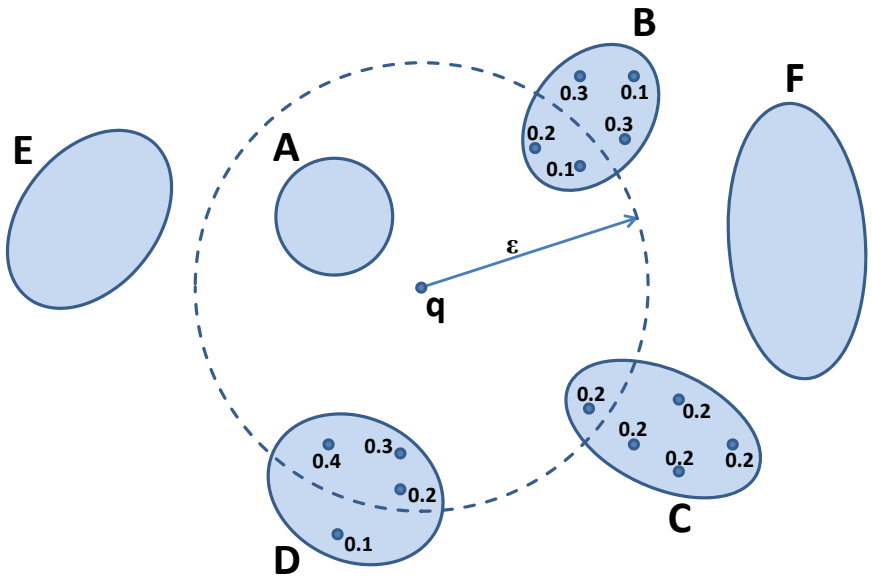

Figure 1: Example of an uncertain ϵ-range query. Object A is a true hit, objects B, C and D are possible hits.

to Q than A, thus counting a number of objects within a distance of less than the distance between Q and A; and for a distance ranking query, the probability that an object A has the k-th nearest objects of Q is the probability that exactly $k-1$ objects (other than A) have a distance to Q less than the distance between Q and A.

Example 1. *As an example of counting the number of uncertain objects within a region, reconsider Figure 1, and assume a query that counts the number of uncertain objects within a distance of ϵ from a query object q. We first note that this query answer is a random variable, which depends on the locations of the uncertain objects (which are random variables, too). We observe that objects E and F are guaranteed to be outside the range, such that we can prune them from our computation. We also note that object A is guaranteed to be in the range, allowing us to increment the query result by one without having to further consider this object. For objects B, C, and D, the events of being located inside the query range are random variables with probabilities of 0.3, 0.2, and 0.9, respectively. Thus, the result of this query is a random variable having a sample space of $\{1,2,3,4\}$, and mapping each of these possible results to their probability. For example, the probability of having exactly one object in the range is $0.7 \cdot 0.8 \cdot 0.1 = 0.056$. For the probability that exactly two objects are inside the range, we can add the probabilities on the three possible worlds where exactly one object out of $\{B,C,D\}$ is inside the range.*

In the general case of computing the probability that exactly k out of n uncertain objects are inside the query range, we need to aggregate the probabilities of $\binom{n}{k}$ combinations of objects to be inside the query range. Straightforward approaches which enumerate all the $\binom{n}{k}$ possible worlds the number of which is in $O(n^k)$.

Yet, for this problem of counting the (distribution of the) number of uncertain objects within a query range two efficient solutions based on 1) the Poisson-Binomial Recurrence [26, 25, 27] and 2) based on Generating Functions [22] have been proposed independently in the literature. These solutions allow to aggregate the probabilities of an exponential number of possible combinations of objects in polynomial time, allowing us to answer many important spatial query types efficiently. This spatial gem describes how the generating function technique, which was first presented in the context of distance ranking by Li, Saha, and Deshpande in the best paper of VLDB 2009, can be used to efficiently answer spatial queries on uncertain data.

2 Generating Functions for Probabilistic Counting

Let $\mathcal{X} = \{X_1, ..., X_N\}$ denote the set of objects having a non-zero probability of being located in the query region and let p_i denote the probability of object X_i to be located inside the query region. We can model each $X_i \models B(p_i)$ as a Bernoulli distributed random variable that has a probability of p_i of being 1 and a probability of $(1-p_i)$ of being 0. With this model, the count of objects inside the query region is the sum $\sum_{i=1}^{N} B(p_i)$. We note that since the probabilities p_i are not identical this random variable does not follow a Binomial distribution, but is instead known as

a Poisson-Bionomial distribution having parameters $\{p_1, ..., p_N\}$ [25]. Our goal is to evaluate this random variable $\sum_{i=1}^{N} B(p_i)$ efficiently, that is, for each $0 \leq k \leq N$ we want to derive the probability $P(\sum_{i=1}^{N} B(p_i) = k)$.

For this purpose, represent each random variable X_i by a polynomial $poly(X_i) = p_i \cdot x + (1 - p_i)$. Consider the generating function

$$\mathcal{F}^N = \prod_{i=1}^{N} poly(X_i) = \sum_{i=0}^{N} c_i x^i. \tag{1}$$

The coefficient c_i of x^i in the expansion of $\mathcal{F}^N$ equals the probability $P(\sum_{n=1}^{N} X_n = i)$ ([22]). For example, the monomial $0.25 \cdot x^4$ implies that with a probability of 0.25, the sum of all Bernoulli random variables equals four.

The expansion of N polynomials, each containing two monomials leads to a total of 2^N monomials, one monomial for each sequence of successful and unsuccessful Bernoulli trials, i.e., one monomial for each possible world. To reduce this complexity, an iterative computation of $\mathcal{F}^N$, can be used, by exploiting that

$$\mathcal{F}^k = \mathcal{F}^{k-1} \cdot poly(X_k). \tag{2}$$

This rewriting of Equation 1 allows to inductively compute $\mathcal{F}^k$ from $\mathcal{F}^{k-1}$. The induction is started by computing the polynomial $\mathcal{F}^0$, which is the empty product which equals 1, the neutral element of multiplication, i.e., $\mathcal{F}^0 = 1$. To understand the semantics of this polynomial, the polynomial $\mathcal{F}^0 = 1$ can be rewritten as $\mathcal{F}^0 = 1 \cdot x^0$, which we can interpret as the following tautology:"with a probability of one, the sum of all zero Bernoulli trials equals zero." After each iteration, we can unify monomials having the same exponent, leading to a total of at most $k + 1$ monomials after each iteration. This unification step allows to remove the combinatorial aspect of the problem, since any monomial x^i corresponds to a class of equivalent worlds, such that this class contains only and all of the worlds where the sum $\sum_{k=1}^{N} X_k = 1$. In each iteration, the number of these classes is at most k and the probability of each class is given by the coefficient of x^i.

Example 2. *As an example, consider again the running example of Figure 1. For each object, we first obtain the probability of being located inside the query region (which can be done in linear time using a range query and aggregating the probabilities of instances inside the query region). For the four objects A, B, C, D, E, and F, we obtain probabilities of being inside the query region of* 1.0, $p_1 := 0.3$, $p_2 := 0.2$, $p_3 := 0.9$, 0, *and* 0, *respectively. We can safely prune objects E and F since they cannot affect the query result. We can also prune object A by increasing the result by* 1 *since we know A must be inside the query range. Given the probabilities* $p_1 = 0.3$, $p_2 = 0.2$, *and* $p_3 = 0.9$ *we obtain the three generating polynomials* $poly(X_1) = (0.3x + 0.7)$, $poly(X_2) = (0.2x + 0.8)$, *and* $poly(X_3) = (0.9x + 0.1)$. *We trivially obtain* $\mathcal{F}^0 = 1$. *Using Equation 2 we get*

$$\mathcal{F}^1 = \mathcal{F}^0 \cdot poly(X_1) = 1 \cdot (0.3x + 0.7) = 0.3x + 0.7.$$

Semantically, this polynomial implies that out of the first one Bernoulli trials, the probability of having a sum of one is 0.3 *(according to monomial* $0.3x = 0.3x^1$*), and the probability of having a sum of zero is* 0.7 *(according to monomial* $0.7 = 0.7x^0$*. Next, we compute* F^2*, again using Equation 2:*

$$\mathcal{F}^2 = \mathcal{F}^1 \cdot poly(X_2) = (0.3x^1 + 0.7^0) \cdot (0.2x^1 + 0.8x^0) =$$
$$0.06x^1x^1 + 0.24x^1x^0 + 0.14x^0x^1 + 0.56x^0x^0$$

In this expansion, the monomials have deliberately not been unified to give an intuition of how the generating function technique is able to identify and unify equivalent worlds. In the above expansion, there is one monomial for each possible world. For example, the monomial $0.14x^0x^1$ *represents the world where the first trial was unsuccessful (represented by the* 0 *in the first exponent) and the second trial was successful (represented by the* 1 *in the second exponent). The above notation allows to identify the sequence of successful and unsuccessful Bernoulli trials, clearly leading to a total of* 2^k *possible worlds for* $\mathcal{F}^k$*. However, we know that we only need to compute the total number of successful trials, we do not need to know the sequence of successful trials. Thus, we may treat worlds having the same number of successful Bernoulli trials equivalently, to avoid the enumeration of an exponential number of sequences. This is done implicitly by polynomial multiplication, exploiting that*

$$0.06x^1x^1 + 0.24x^1x^0 + 0.14x^0x^1 + 0.56x^0x^0 = 0.06x^2 + 0.24x^1 + 0.14x^1 + 0.56x^0$$

This representation no longer allows to distinguish the sequence of successful Bernoulli trials. This loss of information is beneficial, as it allows to unify possible worlds having the same sum of Bernoulli trials

$$0.06x^2 + 0.24x^1 + 0.14x^1 + 0.56x^0 = 0.06x^2 + 0.38x^1 + 0.56x^0$$

The remaining monomials represent an equivalence class of possible worlds. For example, monomial $0.38x^1$ *represents all worlds having a total of one successful Bernoulli trial out of the first two trials. This is evident since the coefficient*

of this monomial was derived from the sum of both worlds having a total of one successful Bernoulli trial. In the next iteration, we compute:

$$\mathcal{F}^3 = \mathcal{F}^2 \cdot poly(X_3) = (0.06x^2 + 0.38x^1 + 0.56x^0) \cdot (0.9x + 0.1)$$

$$= 0.054x^2x^1 + 0.006x^2x^0 + 0.342x^1x^1 + 0.038x^1x^0 + 0.504x^0x^1 + 0.056x^0x^0$$

This polynomial represents the three classes of possible worlds in $\mathcal{F}^2$ combined with the two possible results of the third Bernoulli trial, yielding a total of $3\dot{2}$ monomials. Unification yields

$$0.054x^2x^1 + 0.006x^2x^0 + 0.342x^1x^1 + 0.038x^1x^0 + 0.504x^0x^1 + 0.056x^0x^0 =$$

$$0.054x^3 + 0.348x^2 + 0.542x^1 + 0.056x^0$$

This polynomial describes the PDF of $\sum_{i=1}^{3} X_i$ (having $X_1 = B, X_2 = C, X_3 = D$), since each monomial $c_i x^i$ implies that the probability, that out of all three Bernoulli trials, the total number of successful events equals i, is c_i. Thus, we get $P(\sum_{i=1}^{3} X_i = 0) = 0.0056$, $P(\sum_{i=1}^{3} X_i = 1) = 0.542$, $P(\sum_{i=1}^{3} X_i = 2) = 0.348$, and $P(\sum_{i=1}^{3} X_i = 3) = 0.054$.

3 Complexity Analysis

The generating function technique requires a total of N iterations (as in the worst case, all uncertain objects have a non-zero non-one probability of being in the query region). In each iteration $1 \leq k \leq N$, a polynomial of degree $k-1$, and thus of maximum length k, is multiplied with a polynomial of degree 1, thus having a length of 2. This requires to compute a total of $(k+1) \cdot 2$ monomials in each iteration, each requiring a scalar multiplication. This leads to a total time complexity of $\sum_{i=1}^{N} 2k + 2 \in O(N^2)$ for the polynomial expansions. Unification of a polynomial of length k can be done in $O(k)$ time, exploiting that the polynomials are sorted by the exponent after expansion. Unification at each iteration leads to a $O(n^2)$ complexity for the unification step. This results in a total complexity of $O(n^2)$, similar to the Poisson binomial recurrence approach.

Many spatial query predicates do not require to compute the full probability mass of the distribution of the number of objects within the query range but only require the probability of having less or equal than a specified parameter K of objects in the query range. For example, to find the probability that an object is among the K-nearest neighbors of a query object, it is sufficient to compute the probability that at most $K-1$ objects are closer (within a shorter range). In this case, all monomials having an exponent greater or equal to K can be pruned from the computation. In this case, where the length of the expanded polynomial in each iteration is bounded by K, thus yielding a run-time complexity of $O(k \cdot n)$. This efficient computation can also be leveraged for the case of distance ranking, where the challenge is to find the probability that exactly $K-1$ other objects are closer to a query object for an object to be exactly the K'th nearest neighbor (i.e., having a distance rank of K). This task requires only to find the coefficient c_{K-1} of the expanded monomial $c_{K-1}x^{K-1}$ having an exponent of $K-1$. Since in each iteration of multiplying and expanding monomials the coefficient c_{K-1} only depends on the coefficients c_{K-2} and c_{K-1} of previous iterations, we may also discard monomials having an exponent of K or greater to answer distance ranking queries.

To summarize, using generating functions we can compute the distribution of the number of objects within a query range in $O(n^2)$, where n is the number of database objects. In cases such as KNN or distance ranking queries where we only need to know the probability of having at most or exactly K objects within the query range, we can reduce this complexitiy to $O(K \cdot n)$ by truncating intermediate polynomials.

4 Implementation

A Python implementation can be found in the following GitHub repository `https://github.com/azufle/generating_functions`. This implementation, both as a Jupyter notebook and a classic .py script defines a function that efficiently computes the probability mass function of a probabilistic count given a list of probabilities. Additional documentation can be found in the repository.

5 Variants, Extensions, and Improvements

This section surveys a variety of extensions and improvements of the classic generating functions.

5.1 Acceleration using Discrete Fourier Transform

An advantage of the generating function approach is that this naive polynomial multiplication can be accelerated using Discrete Fourier Transform (DFT). This technique allows to reduce to total complexity of computing the sum of N Bernoulli random variables to $O(Nlog^2N)$ ([28]). This acceleration is achieved by exploiting that DFT allows to expand two polynomials of size k in $O(klogk)$ time. Equi-sized polynomials are obtained in the approach of [28], by using a divide and conquer approach, that iteratively divides the set of N Bernoulli trials into two equi-sized sets. Their recursive algorithm then combines these results by performing a polynomial multiplication of the generating polynomials of each set. More details of this algorithm can be found in [28].

5.2 Extension to Uncertain Counts

In many applications, a probabilistic event may not only have two possible outcomes (Yes/No, Success/Failure), but may have a third outcome that represents an unknown/undecided/uncertain state. For example, given incomplete trajectories of objects and their resulting uncertainty regions at a time (for example, described by a bounding box of possible locations), there may be possible worlds where 1) an object is within the query region, 2) an object is outside the query region, and 3) containment of the object within the query region cannot be decided due to uncertainty.

For such cases, the addition of an unknown state has been proposed to be represented by generating function in [29]. For each object $X_i \in \mathcal{X} = \{X_1, ..., X_N\}$ having a probability of p_i to satisfy the query condition (such as being located within the query region), a probability of $\bar{p}_i$ to not satisfy the query condition, and a probability of $1 - p_i - \bar{p}_i$ of being in an unknown state, we can consider the generating function:

$$\mathcal{F}^N = \prod_{i=1}^{N} p_i \cdot x + (1 - p_i - \bar{p}_i) \cdot y + \bar{p}_i$$

Intuitively, the anonymous variable x denotes an event of satisfying the query condition and the anonymous variable y denotes the event of an undecided satisfaction of the query condition. In the expanded polynomial, a monomial such as $c_{i,j}x^iy^j$ corresponds to a possible world having a probability of $c_{i,j}$ of having i objects guaranteed to satisfy the query condition and j additional objects possibly satisfying the query condition. For example, a monomial $0.13x^3y^2$ corresponds to a possible world having a probability of 0.13 and having at least 3 but no more than $3 + 2 = 5$ objects satisfy the query predicate.

5.3 Dynamic Polynomials

In many applications, the probability of an object to be inside a query range may change dynamically. For example, mobile objects may send update location information to change their uncertainty region. In this case, the probability distribution of the number of objects inside a query range changes as well. To update the probability distribution, we may recompute from scratch, using all objects having non-zero probability of being inside the query range. However, such an approach may be inefficient when there is a large number of such objects having frequent updates. To update the probability distribution of a probabilistic count, we can use polynomial division as described in [30]. Thus, having a database $\mathcal{X} = \{X_1, ..., X_N\}$ of objects each having a probability of p_i to be inside the query and given the polynomial $\mathcal{F}^N$ that describes the probability distribution of the number of objects inside the query range, assume that an object X_j changes it's probability from p_j to p'_j. We can update the $\mathcal{F}^N$ by removing the old effect of the old probability p_i through polynomial division of polynomial $poly(X_j) = p_i \cdot x + 1 - p_i$ and by including the effect of the effect of the new probability through multiplication with polynomial $poly(X'_j) = p'_i \cdot x + 1 - p'_i$. Combining both steps, we obtain the updated polynomial $\mathcal{F}'^N$:

$$\mathcal{F}'^N = \mathcal{F}^N \frac{poly(X'_j)}{poly(X_j)} = \mathcal{F}^N \frac{p'_i \cdot x + 1 - p'_i}{p_i \cdot x + 1 - p_i}.$$

References

[1] Michael F Goodchild. Uncertainty: The Achilles Heel of GIS. *Geo Info Systems*, 8(11):50–52, 1998.

[2] Matthias Renz, Reynold Cheng, and Hans-Peter Kriegel. Similarity search and mining in uncertain databases. *Proceedings of the VLDB Endowment*, 3(1-2):1653–1654, 2010.

[3] Reynold Cheng, Tobias Emrich, Hans-Peter Kriegel, Nikos Mamoulis, Matthias Renz, Goce Trajcevski, and Andreas Züfle. Managing uncertainty in spatial and spatio-temporal data. In *2014 IEEE 30th International Conference on Data Engineering*, pages 1302–1305. IEEE, 2014.

[4] Andreas Züfle, Goce Trajcevski, Dieter Pfoser, Matthias Renz, Matthew T Rice, Timothy Leslie, Paul Delamater, and Tobias Emrich. Handling uncertainty in geo-spatial data. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1467–1470. IEEE, 2017.

[5] Andreas Züfle, Goce Trajcevski, Dieter Pfoser, and Joon-Seok Kim. Managing uncertainty in evolving geo-spatial data. In *2020 21st IEEE International Conference on Mobile Data Management (MDM)*, pages 5–8. IEEE, 2020.

[6] Andreas Züfle. Uncertain spatial data management: An overview. *Handbook of Big Geospatial Data*, pages 355–397, 2021.

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

[8] Lyublena Antova, Thomas Jansen, Christoph Koch, and Dan Olteanu. Fast and Simple Relational Processing of Uncertain Data. In *2008 IEEE 24th International Conference on Data Engineering*, pages 983–992. IEEE, 2008.

[9] Jihad Boulos, Nilesh Dalvi, Bhushan Mandhani, Shobhit Mathur, Chris Re, and Dan Suciu. MYSTIQ: A system for finding more answers by using probabilities. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 891–893. ACM, 2005.

[10] Parag Agrawal, Omar Benjelloun, Anish Das Sarma, Chris Hayworth, Shubha Nabar, Tomoe Sugihara, and Jennifer Widom. Trio: A System for Data, Uncertainty, and Lineage. *Proc. of VLDB 2006 (demonstration description)*, 2006.

[11] Daisy Zhe Wang, Eirinaios Michelakis, Minos Garofalakis, and Joseph M Hellerstein. BAYESSTORE: Managing Large, Uncertain Data Repositories with Probabilistic Graphical Models. *Proceedings of the VLDB Endowment*, 1(1):340–351, 2008.

[12] Nilesh Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, 2007.

[13] Anna Follmann, Mario A Nascimento, Andreas Züfle, Matthias Renz, Peer Kröger, and Hans-Peter Kriegel. Continuous probabilistic count queries in wireless sensor networks. In *International Symposium on Spatial and Temporal Databases*, pages 279–296. Springer, 2011.

[14] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Querying imprecise data in moving object environments. *IEEE Trans. Knowl. Data Eng.*, 16(9):1112–1127, 2004.

[15] Yuichi Iijima and Yoshiharu Ishikawa. Finding probabilistic nearest neighbors for query objects with imprecise locations. In *MDM*, pages 52–61, 2009.

[16] Reynold Cheng, Jinchuan Chen, Mohamed F. Mokbel, and Chi-Yin Chow. Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In *ICDE*, pages 973–982, 2008.

[17] George Beskales, Mohamed A. Soliman, and Ihab F. IIyas. Efficient search for the top-k probable nearest neighbors in uncertain databases. *Proc. VLDB Endow.*, 1(1):326–339, 2008.

[18] Vebjorn Ljosa and Ambuj K. Singh. Apla: Indexing arbitrary probability distributions. In *ICDE*, pages 946–955, 2007.

[19] Reynold Cheng, Lei Chen, Jinchuan Chen, and Xike Xie. Evaluating probability threshold k-nearest-neighbor queries over uncertain data. In *EDBT*, pages 672–683, 2009.

[20] G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data and expected results. In *2009 IEEE 25th International Conference on Data Engineering*, pages 305–316, 2009.

[21] M.A. Soliman and I.F. Ilyas. Ranking with uncertain scores. In *2009 IEEE 25th International Conference on Data Engineering*, pages 317–328, 2009.

[22] Jian Li, Barna Saha, and Amol Deshpande. A Unified Approach to Ranking in Probabilistic Databases. *Proceedings of the VLDB Endowment*, 2(1):502–513, 2009.

[23] Graham Cormode, Feifei Li, and Ke Yi. Semantics of Ranking Queries for Probabilistic Data and Expected Ranks. In *2009 IEEE 25th International Conference on Data Engineering*, pages 305–316. IEEE, 2009.

[24] Jian Li and Amol Deshpande. Ranking Continuous Probabilistic Datasets. *Proceedings of the VLDB Endowment*, 3(1-2):638–649, 2010.

[25] Ming Hua, Jian Pei, Wenjie Zhang, and Xuemin Lin. Ranking queries on uncertain data: a probabilistic threshold approach. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 673–686, 2008.

[26] Ke Yi, Feifei Li, George Kollios, and Divesh Srivastava. Efficient processing of top-k queries in uncertain databases with x-relations. *IEEE transactions on knowledge and data engineering*, 20(12):1669–1682, 2008.

[27] Thomas Bernecker, Hans-Peter Kriegel, Nikos Mamoulis, Matthias Renz, and Andreas Zuefle. Scalable Probabilistic Similarity Ranking in Uncertain Databases. *IEEE Transactions on Knowledge and Data Engineering*, 22(9):1234–1246, 2010.

[28] Jian Li, Barna Saha, and Amol Deshpande. A unified approach to ranking in probabilistic databases. *VLDB Journal*, 20(2):249–275, 2011.

[29] Thomas Bernecker, Tobias Emrich, Hans-Peter Kriegel, Nikos Mamoulis, Matthias Renz, and Andreas Züfle. A novel probabilistic pruning approach to speed up similarity queries in uncertain databases. In *2011 IEEE 27th International Conference on Data Engineering*, pages 339–350. IEEE, 2011.

[30] Nina Hubig, Andreas Züfle, Tobias Emrich, Mario A Nascimento, Matthias Renz, and Hans-Peter Kriegel. Continuous probabilistic sum queries in wireless sensor networks with ranges. In *International Conference on Scientific and Statistical Database Management*, pages 96–105. Springer, 2012.