Cost Models and Efficient Query Processing over Existentially Uncertain Spatial Data

Elias Frentzos, Nikos Pelekis, Yannis Theodoridis

Laboratory of Information Systems

Department of Informatics

University of Piraeus

Hellas

Technical Report Series

UNIPI-ISL-TR-2008-01

May 2008

Cost Models and Efficient Query Processing over Existentially Uncertain Spatial Data

Elias Frentzos, Nikos Pelekis, Yannis Theodoridis

Abstract— *The domain of existentially uncertain spatial data refers to objects that are modelled using an existential probability accompanying spatial data values. An interesting and challenging query type over existentially uncertain data is the search of the Nearest Neighbor (NN), since the probability of a potential dataset object to be the NN of the query object depends on the locations and probabilities of other points in the same dataset. In this paper, following a statistical approach, we estimate the average number of the NNs required to answer probabilistic thresholding NN (PTNN) queries as function of the threshold* t*, allowing us to utilize existing approaches and propose a cost model for such queries. Based on the same statistical approach, we propose an efficient algorithm for PTNN queries over arbitrarily structured existentially uncertain spatial data. Our experimental study demonstrates the accuracy and efficiency of the proposed techniques.*

I. Introduction

A major challenge posed by real-world applications involving spatial information deals with the uncertainty inherent in the data. In the literature, two types of uncertainty have gained the interest of the research community, namely the *locational* and the *existential* uncertainty. *Locationally* uncertain are the objects that do exist but their location is uncertain; as such, this kind of uncertainty is described by a probability density function. On the other hand, *existentially* uncertain objects are those that their uncertainty emanates from their existence, and this is expressed by a probability E_x accompanying the spatial value of object x and reflecting the confidence of x's existence. As a motivating example, consider the case where an image processing tool extracts some interesting formations of pixels that may or may not correspond to a predefined type of objects due to low image resolution; the presence of existential uncertainty is also natural in the case of fuzzy classification [2], while it can be used to represented a confidence factor of the presence of historical events in the past [3].

The single related work on existentially uncertain data [2] focuses on two probabilistic versions of spatial queries. A *thresholding* query returns the objects that satisfy some spatial condition with probability more than a given threshold t, while a *ranking* query returns the objects that satisfy a spatial

This work is supported by the Diachoron project, founded by the Greek Ministry of Development, General Secretariat for Research and Technology, co-funded by the European Union. Elias Frentzos is with the University of Piraeus, e-mail: efrentzo@unipi.gr. Nikos Pelekis is with the University of Piraeus, e-mail: npelekis@unipi.gr. Yannis Theodoridis is with the University of Piraeus, e-mail: ytheod@ unipi.gr.

condition in order of their confidence, applying the number of objects requested as threshold. Dai et al. [2] proposed search algorithms for the above two types of spatial range and nearest neighbor (NN) queries, where the existentially uncertain data are indexed by 2-dimensional R-trees [5] or appropriate augmented variants of them.

In this paper, we focus on the *probabilistic thresholding nearest neighbor* (PTNN) query on existentially uncertain data. The motivation is that, this type of query presents a quite involved search complexity, as the probability of an object to be the NN depends not only on the location, but also on the existential probability of other objects. Outlining the major contributions of this paper, we first present a statistical-based analysis for the determination of the *discrete distribution probability density function* (*dpdf*) that PTNN query terminates after having retrieved exactly n objects; then we present a cost model for the forecasting of the number of disk accesses required to process a PTNN query, given that the dataset is indexed by R-trees [MNPT05]. Finally, we present an optimal algorithm for the execution of PTNN queries over arbitrarily structured data. To the best of our knowledge, our work is the first on these topics.

The rest of the paper is structured as follows: Section 2 overviews the background work which ours has built upon. Section 3 describes the statistical analysis of PTNN queries, while Sections 4 and 5 present the cost model and an efficient algorithm, respectively, for PTNN queries over arbitrary structured datasets. Section 6 presents the results of our experimental study, while Section 7 concludes the paper and provides directions for future work. The notation used in this paper is summarized in Table I.

II. Background

Formally, a PTNN query takes as input a query object q and a threshold probability t, while the data are represented as tuples of the form (x, E_x). As proposed by Dai et al. [2], the 2-dimensional PTNN (PTNN2D) algorithm, illustrated in Figure 1, iteratively retrieves spatially nearest objects in a Best-First (BF) mode [4], and terminates only after the value of P^{fist} becomes smaller than the given threshold t. The PTNN2D algorithm iteratively calculates the value of P^{first}, which is a variable that captures the probability that no object retrieved before the current object x is the actual NN, according to [2]:

$$P_x^{first} = \prod_{i=1}^{n-1} (1 - E_i), \tag{1}$$

TABLE I
TABLE OF NOTATIONS

Notation	Description
x, E_x	A data point and its existential probability
S	A dataset of tuples (x, E_x)
q, t	The query object and threshold probability of a PTNN query
P^{first}	The probability that no object retrieved before the current object x is the actual NN
P_x	The probability that an object x is the actual NN
$P_{exact}(n)$	The probability that the algorithm terminates after having retrieved exactly n objects
$\overline{n}$	Average number of iterations
D_k	The nearest distance from the query point q to its k-th NN
CI	A user-defined confidence
F	The smallest number of NNs required by the algorithm, as to resolve a PTNN query with a user-defined confidence CI, without the need to retrieve $n > f$ NNs.

where n-1 is the number of objects that are closer to the query object than the current object x, i.e., the number of objects retrieved from the BF algorithm before object x, and E_x their existential uncertainty. Then, the probability that an object x is the actual NN, is [2]:

$$P_x = E_x \cdot P_x^{first} \quad (2)$$

The intuition behind the PTNN2D algorithm is that once $P^{fist} < t$, we are sure that the subsequent nearest objects, even if they exist with 100% probability, they cannot be the NN of q, so the algorithm can safely terminate. It is also worth to note that the PTNN2D algorithm utilizes the R-tree index only to incrementally retrieve the k-th NN; as such, the R-tree can be replaced by any other access method supporting incremental NN search.

However, the number of iterations (and, subsequently, the complexity) of the PTNN2D algorithm may be arbitrarily large; the expected cost of this particular type of query is not discussed in [2]. The lack of an analytical methodology for estimating the cost of PTNN queries over existential uncertain datasets has motivated us to use statistical methods in order to estimate the average number of NNs that one needs to search in order to be able to resolve PTNN queries. Based on our analysis, we exploit well-known work on cost models of NN queries over regular multi-dimensional datasets [6], and define cost model appropriate for PTNN queries over existential uncertain data indexed by R-trees [5].

More specifically, Tao et al. [6] present an efficient cost model for the optimization of NN queries in low- and medium-dimensional spaces. They provide a closed formula for the estimation of (a) the average nearest distance D_k from the query point q to its k-th NN and (b) the number of tree nodes whose MBRs intersect the vicinity circle $\Theta(q, D_k)$ with center q and radius D_k, which is equal with the average number of node accesses $NA(k)$ required by an R-tree to retrieve the k-th NN. Specifically, according to the analysis of [6], the average nearest distance D_k is estimated by:

$$D_k \approx \frac{2}{C_V}\left[1-\sqrt{1-\left(k/N\right)^{1/d}}\right] \quad (3)$$

where d is the dimensionality, N is the cardinality and C_V is calculated by:

$$C_V = \frac{\sqrt{\pi}}{\left[\Gamma\left(d/2+1\right)\right]^{1/d}} \quad (4)$$

In our approach, we appropriately employ these techniques so as to estimate the average number of iterations $\overline{n}$ required by the PTNN algorithm in order to terminate in the case of uniformly distributed data.

Furthermore, we utilize the above mentioned statistical model in order to estimate the number f of NNs that are to be retrieved from the database so as to be at least CI % confident – CI is a user-defined confidence (e.g. 99%) – that the PTNN search will end without the need to retrieve $n > f$ NNs. The motivation behind this approach is to provide efficient search algorithms, with predetermined cost, and with custom defined certainty (as high as required) of resolution. The applicability of such a technique is extended in many different scenarios, and mainly in the case where existentially uncertain data are not indexed by any spatial index, or when the index does not support the incremental retrieval of the spatial NNs to the query point, as required by the PTNN2D algorithm [2].

```
1.  Algorithm PTNN2D(q, 2D R-tree on S, t)
2.     P^first=1; /*Probability of no object before x*/
3.     While P^first≥t and more objects in S do
4.       x:=next NN of q in S (use BF [3]);
5.       P_x:= P^first·E_x;
6.       If P_x≥t then output (x, P_x);
7.       P^first= P^first·(1−E_x);
```

Figure 1: Probabilistic NN on a 2D R-tree with thresholding

III. Statistical Analysis of PTNN Queries

To start with, we provide a lemma from which the cost model and efficient query processing techniques introduced in this paper are straightforwardly devised. More specifically, the first step towards a cost model for the PTNN2D algorithm [2], is to determine the *discrete probability density function (dpdf)* that the algorithm terminates after exactly n iterations, i.e., the distribution of the number of objects retrieved before P^{first} becomes less than the given threshold t. Towards this goal, we employ the *uncertainty uniformity assumption*, that is, the value of existential uncertainty E_x for all objects in the dataset S is uniformly distributed inside the unit interval [0, 1]. Formally, we provide the following lemma:

Lemma 1: *The dpdf that the PTNN2D algorithm terminates after exactly n iterations, under the uncertainty uniformity assumption, is given by the following formula*:

$$P_{exact}(n) = \frac{(-1)^{n-1} t \ln(t)^{n-1}}{(n-1)!} \tag{5}$$

where t is the algorithm threshold.

Proof: Our goal is to determine the discrete distribution probability density function $P_{exact}(n)$, such that, the algorithm terminates after having retrieved exactly n objects. For this we distinguish between two cases, namely $n = 1$ and $n > 1$. In the former case, the algorithm terminates with a single iteration iff the value of $P_2^{first} = \prod_{i=1}^{1}(1-E_i) = (1-E_1)$ calculated at the end of the first iteration (i.e., line 7 in Figure 1) is less than the given threshold t. Thus, from the uncertainty uniformity assumption, it holds that $P_{exact}(1) = P(1-E_1 \le t) = P(E_1 \ge 1-t) = t$. Given that $-1^0 = (\ln(t))^0 = 0! = 1$ we have proved Lemma 1 in the case where $n = 1$.

In the latter case, i.e., $n > 1$, the algorithm terminates iff P_{n+1}^{first}, which is calculated at the end of the n^{th} iteration (i.e., line 7 in Figure 1), becomes less than t after *exactly n* iterations. In other words, we must first determine the conditional probability that P^{first} becomes less than t after n iterations, given also that it must not terminate before reaching n iterations:

$$P_{cond}(n) = P\left(\prod_{i=1}^{n}(1-E_i) \le t \mid \prod_{i=1}^{m}(1-E_i) > t, \forall m \le n-1\right) \tag{6}$$

Then, the total probability that the algorithm terminates after having retrieved exactly n objects can be obtained by multiplying P_{cond} with the probability that the algorithm has not terminated until reaching n iterations:

$$P_{exact}(n) = P_{cond}(n) \cdot P\left(\prod_{i=1}^{m} (1 - E_i) > t, \forall m \le n-1 \right) \quad (7)$$

Moreover, since $0 \le E_1 \le 1 \Leftrightarrow 0 \le 1 - E_1 \le 1$, it also holds that $(1-E_1) \ge .. \ge \prod_{i=1}^{n-2}(1-E_i) \ge \prod_{i=1}^{n-1}(1-E_i)$.

Therefore, given that $\prod_{i=1}^{n-1}(1-E_i) > t$, it stands that $\prod_{i=1}^{m}(1-E_i) > t, \forall m \le n-2$; then, (6) and (7) can be rewritten as follows:

$$P_{cond}(n) = P\left(\prod_{i=1}^{n}(1-E_i) \le t \mid \prod_{i=1}^{n-1}(1-E_i) > t \right) \quad (8)$$

$$P_{exact}(n) = P_{cond}(n) \cdot P\left(\prod_{i=1}^{n-1}(1-E_i) > t \right) \quad (9)$$

Since the values of E_x follow the uniform distribution, the same also stands for 1-E_x; as such the product of the n uniformly distributed values of 1-E_x should follow the *uniform product distribution*, i.e., the distribution of the product of n uniformly distributed uncorrelated variables x_1, x_2,.. x_n, with probability density function (pdf) given by [7]:

$$P_{x_1 x_2 \ldots x_n}(u) = \frac{(-1)^{n-1}}{(n-1)!}(\ln u)^{n-1} \quad (10)$$

where u is the product $\prod x_i$.

In our case, we first set as u the product $\prod_{i=1}^{n-1}(1-E_i)$, and then determine the amount of objects $X \in S$, such as $\prod_{i=1}^{n}(1-E_i) = (1-E_n) \cdot u \le t$ which leads to:

$$(1-E_n) \le t/u \,. \quad (11)$$

Given that $(1-E_n)$ is uniformly distributed, it should hold that the amount of objects fulfilling the above expression V_n is

$$V_n = t/u \,. \quad (12)$$

Known the above, we can calculate the probability $P_{cond}(n)$ by summing up (i.e., integrating) the amount of objects V_n for each value of u, weighted by the value of the distribution of u, and divided by the respective sum (i.e., integral) of the distribution of u. Moreover, since it is known that $u = \prod_{i=1}^{n-1}(1-E_i) \ge t$, the above integrals should involve only the values of u between t and 1. Summarizing:

$$P_{cond}(n) = \frac{\int_t^1 \frac{t}{u} P_{n-1}(u)\,du}{\int_t^1 P_{n-1}(u)\,du} \quad (13)$$

Moreover, the total probability that the algorithm has not been terminated until reaching n iterations (i.e, $\prod_{i=1}^{n-1}(1-E_i) > t$), can be easily calculated, using the pdf of the product of n-1 uniformly distributed variables:

$$P\left(\prod_{i=1}^{n-1}(1-E_i) > t \right) = \int_t^1 P_{n-1}(u)\,du \quad (14)$$

Finally, by substituting (13) and (14) into (9) and performing the necessary calculations[1], we have proved Lemma 1 in the case where $n > 1$■

Lemma 1 provides us with the dpdf that the algorithm terminates after exactly n iterations. The dpdf expressed by (5) is a closed formula, since it involves only the logarithm of the threshold t and the *factorial* of n. Obviously, the density of the probability obtained from (5) for several values of n, is dominated by the factorial of n-1; as such, it is expected that as the number of iterations grows, the respective probability density will tend to zero very fast. In the sequel we employ Lemma 1 in order to produce a cost model and efficient algorithms over arbitrarily structured (e.g., non-indexed) data for PTNN queries over existentially uncertain data.

IV. A Cost Model for PTNN Queries

In this section we present a corollary directly derived from the previously presented Lemma 1, which will help us determining the cost model for PTNN queries over existentially uncertain data.

Corollary 1: *The average number of iterations in each execution of the PTNN2D algorithm is*:

$$\bar{n} = 1 - \ln(t) \tag{15}$$

Proof: The average number of iterations needed from the PTNN2D algorithm in order to terminate can be calculated by averaging the dpdf $P_{cond}(n)$ over all possible values of n:

$$\bar{n} = \sum_{i=1}^{\infty} \frac{(-1)^{i-1} t \ln(t)^{i-1}}{(i-1)!} \cdot i \tag{16}$$

Equation (16) cannot be straightforwardly evaluated since it involves infinity; however, we may calculate its limit:

$$\sum_{i=1}^{\infty} \frac{(-1)^{i-1} t \ln(t)^{i-1}}{(i-1)!} \cdot i = \lim_{n \to \infty} \sum_{i=1}^{n} \frac{(-1)^{i-1} t \ln(t)^{i-1}}{(i-1)!} \cdot i \tag{17}$$

which after the necessary calculations turns into:

$$\sum_{i=1}^{\infty} \frac{(-1)^{i-1} t \ln(t)^{i-1}}{(i-1)!} \cdot i = 1 - \ln(t) \tag{18}$$

Finally, by substituting (18) into (16) we have proved corollary 1■

Obviously, the average number of iterations $\bar{n}$ needed from the PTNN2D in order to terminate, is equal with the number of NNs needed to be retrieved from an existentially uncertain spatial database queried with a query point and a given threshold t. Thus, we may employ the analysis presented in [6] so as to estimate the average radius D_k on which the $\bar{n}$-th NN will be found. Apparently, this model can be applied in our case where the dimensionality d is 2 and the value of $\Gamma(d/2+1)$ is $\Gamma(2/2+1)=1$; then, by substituting the average number of n produced by (15) into the number of k NNs requested, (3) can be rewritten as follows:

$$D_k \approx \frac{2}{\sqrt{\pi}} \left[1 - \sqrt{1 - \sqrt{\frac{1 - \ln(t)}{N}}} \right] \tag{19}$$

From this point on, the analysis of [6] that estimates the number of node accesses $NA(k)$ remains unaffected; the single modification to be made is to calculate D_k using (19) instead of (3); the interested reader is cited to [6] for details. Concluding, the cost model for *PTNN queries over existentially*

[1] All advanced calculations were performed using *Mathematica* software [8].

uncertain data is based on (19), which estimates the distance from the query point that has to be browsed from the database so as to answer such a query; then, the required node accesses $NA(k)$ can be straightforwardly estimated by replacing the D_k into the analysis of [6].

V. Efficient Algorithms for PTNN Queries

The algorithms for PTNN queries presented in [2] assume the presence of a spatial index with the ability to incrementally retrieve the NNs of a query point q (i.e., line 4 in Figure 1). However, this is not the single case, since the actual data may be available in a variety of underlying data structures (e.g., non-indexed data) which are unable to incrementally retrieve the k-th NN as PTNN2D does. Under such circumstances, a non-incremental NN algorithm performs redundant operations, since the retrieval of the k-th NN requires also retrieving all the NNs being before it.

```
 1.  Algorithm GPTNN(q, dataset S, t, k)
 2.     Initialize PQ(k)
 3.     While there are more objects in S do
 4.       x:=next object in S;
 5.       PQ.Add x, E_x, Distance(x, q);
 6.     Loop;
 7.     P^first=1;
 8.     While P^first≥t and more objects in PQ do
 9.       x:=next object in PQ;
10.       P_x:= P^first·E_x;
11.       If P_x≥t then OutList.Add(x, P_x);
12.       P^first= P^first·(1-E_x);
13.     Loop;
14.     If P^first≥t then
15.       GPTNN(q, S, t, 2*k);
16.     Else
17.       Output Outlist;
18.     End If;
```

Figure 2: Probabilistic NN with thresholding

The only way to overpass this obstacle and efficiently process a PTNN query over existentially uncertain spatial data, is to exhaustively scan the database and maintain a priority queue with the k NNs w.r.t. the query point; then, a post-processing step similar with the PTNN2D algorithm [2] would be used in order to determine the actual NNs with probability greater or equal than the given threshold t. Figure 2 illustrates the pseudo-code of this algorithm (named GPTNN), which takes as input a query point q, an existentially uncertain dataset S, the threshold t and an initial, arbitrary large number of k. It exhaustively scans the entire dataset (lines 3-7), maintaining a priority queue PQ (line 2) that is used to store the k NNs of q in the entire S. Then, it performs a post-processing step (lines 8-13) similar to the PTNN2D algorithm [2], which is used to determine the actual probability of each object in PQ to be the NN to q. Finally, given that there exists no guaranty that P^{first} is less than t after having retrieved k nearest objects, the algorithm may be recursively repeated doubling the number of k NNs until P^{first} becomes less than t (lines 14-18). It is clear that the main difference between the proposed GPTNN and PTNN2D is that the latter uses the BF strategy of [4] over an existing R-tree index, while our proposal utilizes for the same purpose a priority queue which is populated after an exhaustive scan; as such, GPTNN can be applied over any kind of structured or unstructured existentially uncertain data.

```
1.   Algorithm CNREQ(t, CI)
2.      While P_sum<CI do
3.        i = i + 1;
4.        Calculate P_exact; /*use Eq.(5) */
5.        P_sum:=P_sum+P_exact;
6.      Loop;
7.      Return i
```

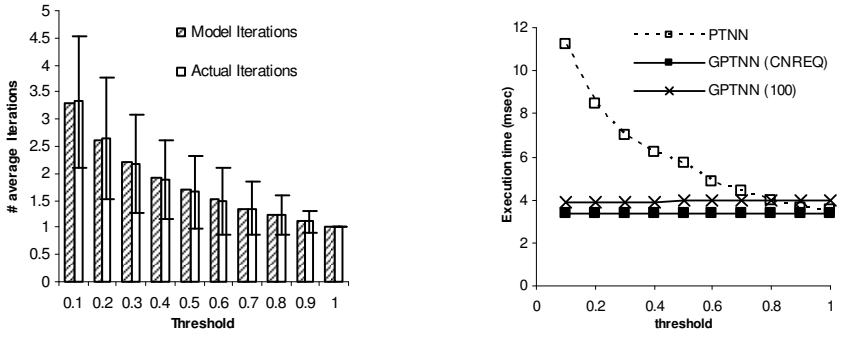
Figure 3: Calculation of n_{req} based on threshold t and confidence interval CI

The efficiency of the GPTNN algorithm is merely based on a suitable choice of k. Choosing small values of k may lead to the repeating of the exhaustive scan in cases where $P^{first} \geq t$ (line 15 in Figure 2); on the other hand, choosing large values of k may lead to decreased performance, due to the length of the priority queue employed (PQ in Figure 2). Following, based on our probabilistic analysis, we provide an effective technique to determine the number of k required to efficiently process the GPTNN algorithm. Specifically, employing the discrete probability density function obtained by Lemma 1, we can determine the required number of k NNs, that have to be retrieved from the database, so as to be sure with a confidence interval CI (typically, $CI \geq 90\%$), that the algorithm will terminate, something that happens when $P^{first} \leq t$. Formally, we aim to determine k w.r.t. the following assumption:

$$\sum_{i=1}^{k} P_{exact}(i) \geq CI \tag{20}$$

and $P_{exact}(i)$ taken from (5). While an analytic solution for this problem is hard to be found, we may easily provide an algorithm which calculates an approximate integer solution. Specifically, the proposed CNREQ algorithm (Figure 3), iteratively calculates $P_{sum} = \sum P_{exact}(i)$ using (5) for $P_{exact}(i)$, increasing i until the value of P_{sum} becomes greater than the requested confidence interval CI; then, it returns the value of i to be the k NNs required as input of the GPTNN algorithm.

Concluding, our algorithmic proposal on PTNN queries consists of the GPTNN algorithm taking as input value of k the value determined by the CNREQ algorithm, given the query threshold t and a large value of CI (e.g., 99%). Under such circumstances, the GPTNN algorithm is expected, with 99% probability, to perform a single sequential scan, demonstrating thus optimal behavior.



(a) (b)

Figure 4: (a) Number of iterations and (b) execution time scaling the threshold

VI. EXPERIMENTAL STUDY

In order to test the accuracy of the proposed model, we conducted a set of experiments, over a synthetic dataset of existentially uncertain point data; all algorithms were implemented in VB.net. We

generated a random point dataset over the unit space, and assigned on each point a value of existential uncertainty, randomly distributed in the interval [0,1]. We then performed 1000 randomly distributed PTNN queries, under various threshold values, and counted the actual number of iterations that the algorithm performed; we also compared the values gathered from the experiment with the one calculated using our model (i.e., Eq.(15)). The corresponding results are illustrated in Figure 4(a). It is clear that the values displayed in both bars (model and actual iterations) are almost identical, meaning that the estimation gathered by our model is very accurate, with an error that never exceeds 2%, regarding the average number of iterations for all 1000 queries. Moreover, the mean deviation (i.e., the average unsigned error of the estimation in each individual query), illustrated by the error bars, is between 20% and 40% in all experimental settings.

We also used the same dataset in order to demonstrate the efficiency of the proposed solution, by performing 1000 randomly distributed PTNN queries following three different strategies: the first (illustrated as PTNN in Figure 4(b)) utilizes the PTNN2D algorithm over an unstructured (i.e., stored in an array) dataset, while the retrieval of the next NN in line 4 of Figure 1 is performed by an exhaustive scan over the entire dataset. The second strategy, called GPTNN(CNREQ), uses the GPTNN algorithm, after having calculated the optimal k using the CNREQ algorithm; finally, the so-called GPTNN(100) uses the GPTNN algorithm, with an arbitrary selected initial k=100. It is clear that the proposed methodology outperforms both its competitors in all cases, while it turns to be practically independent from the value of the threshold; the later is actually an expected result, since the value of k produced by CNREQ for CI = 99% does not vary significantly (it varies between 2 and 7).

VII. Conclusions and Future Work

In this paper, we have worked with the problem of performing *probabilistic thresholding nearest neighbor* queries over existentially uncertain spatial point datasets [2]. Following a statistical approach, we estimate the average number of the nearest neighbors required for processing PTNN queries as a function of the threshold t, and then, utilizing existing approaches [6] we propose a cost model for such queries. Based on the same statistical approach, we further propose an optimal – with a user-defined confidence – algorithm for PTNN queries over arbitrary structured existentially uncertain data. Our experimental study proves the effectiveness and efficiency of the proposed techniques.

There are numerous research directions arising from this work; first of all is the extension of the model in order to support arbitrarily distributed data and existential uncertainties with the usage of spatial histograms [1]. Then, we plan to extend our model in order to support probabilistic ranking nearest neighbor (PRNN) queries [2]. Finally, our last intention is to implement all the proposed methodology on top of a commercial SDBMS and provide commercial users with the entirety of the described functionality.

References

[1] S. Acharya, V. Poosala, S. Ramaswamy, “Selectivity Estimation in Spatial Databases”, Proc. ACM SIGMOD Int’l Conference on Management of Data (SIGMOD’99), pp.13-24, 1999.

[2] X. Dai, M.L. Yiu, N., Mamoulis, Y., Tao, M., Vaitis, “Probabilistic Spatial Queries on Existentially Uncertain Data”, Proc. Int’l Symp. Spatial and Temporal Databases (SSTD’05), pp.254-272, 2005.

[3] Diachoron project [http://www.diachoron.gr]

[4] Hjaltason, G., and Samet, H., Distance Browsing in Spatial Databases, ACM Transactions in Database Systems, vol. 24(2), pp. 265-318, 1999.

[5] Y. Manolopoulos, A. Nanopoulos, A.N. Papadopoulos, Y. Theodoridis, Rtrees: Theory and Applications, Springer 2005.

[6] Y. Tao, J. Zhang, D. Papadias, N. Mamoulis, “An Efficient Cost Model for Optimization of Nearest Neighbor Search in Low and Medium Dimensional Spaces”, IEEE Trans. Knowledge and Data Eng., Vol.16, no.10, pp.1169-1184, 2004.

[7] Weisstein, Eric W. "Uniform Product Distribution." From MathWorld, A Wolfram Web Resource.

[8] Wolfram Research (2005). Mathematica Version 5.2. [http://www.wolfram.com/]