Finding One-Of Probably Nearest Neighbors with Minimum Location Updates

Mitzi McCarthy
Department of Computer Science
La Trobe University
Bundoora, VIC, Australia
Email: m.mccarthy@latrobe.edu.au

X. Sean Wang
Department of Computer Science
The University of Vermont
Burlington, VT, USA
Email: Sean.Wang@uvm.edu

Zhen He
Department of Computer Science
La Trobe University
Bundoora, VIC, Australia
Email: z.he@latrobe.edu.au

***Abstract*—Location information is necessarily uncertain when objects are constantly moving. The cost can be high to maintain precise locations at the application server for all the objects while many applications may not need all the costly precision that is technically possible. An interesting question is how to reduce the cost associated with obtaining precise locations while satisfying user requirements. A general technique of maintaining uncertain location information is using a "safe-region" for each object, a region in which the object must be in but it is not known where it exactly is. Location updates are only needed when the object moves out of its safe region, or a user query needs more precise information to answer. This paper uses the same idea for finding an object that is likely to be the nearest neighbor of a query location, a query type termed herein Probabilistic one-of Nearest Neighbor (PoNN) query. An algorithm is described that first tries to answer a given PoNN query with the known safe regions. If this fails, the algorithm selects some objects to ask for their precise locations (called location exposures). An innovative method is used by the algorithm in deciding which objects to expose in order to reduce the total number of exposures. The innovation includes an information gain formulation of the problem and careful probability calculations. The paper details the algorithm and shows an experimental study of the algorithm against more straightforward solutions in a simulated environment.**

I. Introduction

Location aware mobile devices have become prevalent in recent times. These devices often sense their locations via a global positioning system (GPS) and have the capability to report their locations. The proliferation of such devices has made location based services a fast growing application area.

An interesting query used in location based services is the nearest neighbor query. The nearest neighbor query asks which object is closest to a query location. To answer this, the current location of each object of interest needs to be determined. However, the communications overhead to discover the location of every object is very high. A popular way to minimize this overhead is to use the concept of a safe region [7], [8], [11], [6] around each object. The safe region is an efficient way for the application server to keep track of object locations by obtaining objects' locations only when they move out of their safe regions or a need arises from application requirements.

In existing safe region work the focus has been on the optimal *assignment* of safe regions to objects to minimize location updates due to objects leaving their safe regions. However, we identify that an optimal *evaluation* of queries with the *given* set of safe regions is also an important problem to solve in reducing the overall communication cost. To our knowledge none of the existing research addresses this problem. It is important to note that sometimes a query answer is not known even if the objects stay inside their safe regions, as a result, e.g., of the appearance of a new query or movement of an existing query. In this case we may need to force some of the affected objects to expose their location in order to answer the query. Note that due to our focus on optimal exposure for query evaluation, we do not propose methods for optimal assigning of safe regions in this paper.

In this paper we study the problem of answering the probabilistic, one-of, nearest neighbor (PoNN) query within the aforementioned problem context. The PoNN query returns one object which is either the same distance or closer than all other objects to the query location with a probability over a given threshold. Given a set of safe regions we aim to answer the PoNN query with the minimum number of object location exposures. We adopt an information gain-based solution which aims to progressively expose objects such that the uncertainty in query answer is reduced. For example uncertainty is likely to be reduced when exposing an object will likely result in ruling in (or out) the object being a PoNN.

To our knowledge, there is no existing work which selectively exposes object locations in order to answer probabilistic queries with minimum location updates. Most existing work on probabilistic queries [1], [4], [5], [2], [10], [9], [12], [3] assume the uncertainty in the object location is an inherent property of the problem and cannot be reduced. Therefore these works focus on reporting an answer with an associated level of confidence [4], [5], [9], answer above a certain probability threshold [1], [2], [12], [3], or give the answer in terms of a probability distribution [10]. Within this context they propose CPU and IO efficient solutions. In contrast, we assume object location uncertainty can be improved by requests to objects and thereby answer the probabilistic nearest neighbor query up to a given probability threshold (which can be 100%, reducing to the deterministic case).

The work done by Wolfson et. al. [13] on probabilistic range queries assume the uncertainty in moving object locations can be controlled. However, they effectively control the size of the

978-0-7695-4048-1/10 $26.00 © 2010 IEEE
DOI 10.1109/MDM.2010.44

safe region instead of actively requesting objects to report their locations when answering queries. The work done by Hu et. al. [6] is the only work that studies probabilistic queries within the context of safe regions. However, again they are focused on assigning safe regions to objects instead of minimizing object location exposures when answering queries.

In this paper, we give the details of our information gain-based algorithm, and report the results of a performance study. The experimental results show that our algorithm outperforms straightforward solutions in a variety of situations. At a high level, with this paper, we make the following contributions:

- Frame the problem of answering PoNN queries as a controlled location discovery problem in which the goal is to minimize the number of objects needed to report their current location;
- Introduce a novel information gain-based solution to the aforementioned problem; and
- Obtain performance results through experiments.

The rest of the paper is organized as follows. We first introduce our assumptions and basic definitions in Section II. We then outline our algorithm in Section III. We detail the information gain-based solution in Section IV. In Sections V and VI, we describe the experimental setup and results. We finally conclude our paper with Section VII with a summary statement and some thoughts on possible future directions.

II. Preliminaries

In this section, we provide our assumptions and notation. We assume a set of objects $\mathcal{O}$ in a two-dimensional space. Each object in $\mathcal{O}$ is denoted as s, possibly with a subscript. Each object is assumed to be located in a rectangular, closed, *safe region*. That is, each object can be, and must be, anywhere in a rectangular region $[(x_{min}, y_{min}), (x_{max}, y_{max})]$ denoted by the two opposite corner coordinates.

The two-dimensional space is assumed to be discretely and regularly gridded, i.e., the whole space is partitioned into a set of equal-sized and aligned squares. This is assumed to be our location information precision, i.e., an object can only report its location in terms of cells in the grid, but not the exact location within the cell. In this assumption, the safe region is also assumed to be a rectangle covering exactly some cells.

Figure 1(a) shows the gridded space with two objects s_1 and s_2 with their corresponding safe regions $s_1.SR$ and $s_2.SR$. When the safe region is only of one cell, we say the exact location of the object is known, or its location is *exposed*.

A query location, denoted q, can be anywhere in the 2D space. The query location is modeled to occupy an entire grid cell due to the gridded 2D space, as shown in Figure 1. The numbers in the safe regions correspond to the distance of the cell in the safe region to the query location.

In this paper, we are concerned with the *probabilistic, one-of* nearest neighbor query. More specifically,

Definition 1: Given a threshold value $\tau \in [0, 1]$, query location q and the set $\mathcal{O}$ of objects (with their corresponding safe regions), an object s in $\mathcal{O}$ is called a *probabilistic, one-of, nearest neighbor of* q, denoted PoNN, if the probability of s to be one of the nearest neighbors of q is no less than τ.

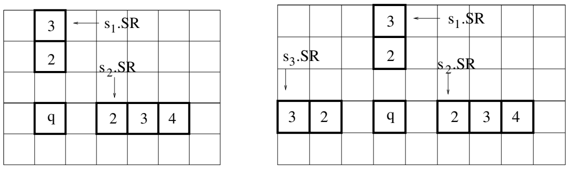
Our query is to find probabilistic, one-of, nearest neighbors of q over the threshold τ. That is, any one of these objects will be satisfactory. Note that when $\tau = 1$, the nearest neighbor we find is one of the deterministic nearest neighbors. And furthermore, when no ties exist, the query result is the nearest neighbor of the query location q.

We now need to discuss the probability of an object being a nearest neighbor. The probabilistic nature arises from the fact that an object can be in any cell in its safe region. A "point" in the probabilistic space (with dimensionality=$|\mathcal{O}|$) consists of the $|\mathcal{O}|$ locations, each corresponding to the location of an object. A *valid "point"* is one that gives non-zero probability, and such a valid "point" must consist of locations within the safe regions of the respective objects. If uniform assumption for safe region is used and objects are independent of each other with respect of their locations in their own safe regions, each valid "point" is equally likely to appear. And in each valid "point", one or more objects are the nearest neighbor of q (only one if no ties). The probability that a particular object s is a nearest neighbor is the number of valid "points" in which s is a nearest neighbor of q over the total number of valid "points" in the probabilistic space.

Given the safe regions of the objects, we can first try to answer the PoNN query. For certain values of τ, it may not be possible, i.e., no answer can be provided. The central problem of this paper is how to reduce the number of reports of object locations, or the number of *exposures*, when finding PoNN.

III. Answering PoNN queries

The task of answering PoNN queries faces two hurdles. The first is how to figure out whether a PoNN can be found with the known safe region and exact location information, and if so, which object is a PoNN. The second hurdle is that if the above fails, what we should do. For the first hurdle, some basic probabilistic calculation exercise suffices and we will describe the exact procedure in the next section. The second hurdle calls for some interesting solution. Obviously, if we know the exact locations of all the objects, then all PoNNs can be found trivially. So a trivial strategy is to ask for the exact locations from all the objects. The question, however, is if we can do better, namely if we can ask the fewest objects to expose their locations. This is a non-trivial optimization problem.



(a) Query and two objects (b) Adding one more object s_3

Fig. 1. Examples of query and object locations in gridded space

We use the two examples in Figures 1(a) and 1(b) to illustrate the problem. The numbers in the cells of the examples represent the distance the cell is from query location

q, and we assume $\tau = 1$. In Figure 1(a), no PoNN can be found with the current safe regions, so we need to expose some objects. There are two possible sequences of location exposures: $\langle s_1, s_2 \rangle$ and $\langle s_2, s_1 \rangle$. If s_1 is exposed first, there is a 50% chance that we find the nearest neighbor of q (if it exposes to be in its cell 2, and in this case s_1 is a PoNN for sure) and the other 50% chance is that it falls into its cell 3, in which case we have to expose s_2 to find the result. So the expected number of exposures is $0.5 * 1 + 0.5 * 2 = 1.5$. If the other order is considered, the expected number of exposures is $2/3 * 1 + 1/3 * 2 = 4/3$, since exposing s_2 first gives us a $2/3$ chance of finding the result with one exposure. Therefore, it is more profitable to expose s_2 first.

In Figure 1(b), with another object s_3 joining in, exposing s_2 first becomes less desirable. Indeed, now there are 6 possible exposure sequences. If s_2 is exposed first, there is 1/3 chance that the nearest neighbor of q can be found with only one exposure (i.e., s_2 falls in its cell 2), and in all other cases, another exposure (of either s_1 or s_3) will give the result (e.g., if s_1 falls in its cell 2). Hence, the expected number of exposures is $1/3*1+2/3*2 = 5/3$. If we expose s_1 first, there is a $1/2$ chance that a PoNN is found (i.e., when it falls in its cell 2), and in the other case (i.e., s_1 falls in its cell 3), we just need to expose s_2 to find one of the nearest neighbors. Hence, in this case, the expected number of exposures is $1/2*1+1/2*2 = 1.5 < 5/3$. Hence, the stochastically optimal strategy is to expose s_1 (or equivalently s_3) first.

In order to make the solution computationally feasible we take a greedy approach. Algorithm 1 describes the detailed steps. The non-trivial step is in line 4, in which we determine which object in B most likely will lead us to answering the query with the minimum number of exposures. This step is the topic for the next section. Note that the algorithm will eventually stop with a finite set $\mathcal{O}$ since once we know all the exact locations. A PoNN can be answered with any τ value, irrespective of the order in which the objects are exposed.

Algorithm 1 Algorithm for finding PoNN.

```
textbfInput: q: query location, O: all data objects, τ: minimum probability threshold
Output: a PoNN of q with τ among objects in O
Method:
1: Initialize B to be the set of objects in O without exact locations (cell level), along with their safe regions
2: Initialize E to be the set of objects in O with known exact locations (cell level), along with their exact locations
3: while PoNN in O cannot be found with information in B ∪ E do
4:     Find an object s in B and move s from B to E
5:     Ask s for its exact location (at the cell level).
6: end while
7: Return a PoNN s in O
```

IV. Entropy-based safe region exposure

In this section, we consider the method of picking the best object to expose for line 4 of Algorithm 1, which is the main contribution of this paper. The aim of this line is to expose the object which is most likely to result in answering the PoNN query with minimal exposures. This is a non-trivial stochastic optimization problem, and in this section we provide an entropy-based approximation approach. We will also provide procedures for lines 3 and 7.

The notions of entropy and information gain form the basis of our approach. Here we use the concept of entropy to mean the amount of uncertainty in the answer of the nearest neighbor query given the current set of object safe regions. Accordingly we aim to expose the object whose exposure will lead to the greatest reduction in entropy and correspondingly the greatest information gain. We define the entropy for the query q_{NN} under the current safe regions, denoted $H(q_{NN})$. Note, however, since we are concerned with the "one-of" semantics, i.e., we are interested in one of the nearest neighbors (in case of ties) of a query location, we first define the entropy in terms of an object s, intuitively meaning the uncertainty of s being a nearest neighbor, denoted $H_q(s)$, given as follows:

$$\begin{aligned} H_q(s) = &- P(s \in q_{NN}) log_2 P(s \in q_{NN}) \\ &- P(s \notin q_{NN}) log_2 P(s \notin q_{NN}) \end{aligned} \tag{1}$$

where $P(s \in q_{NN})$ is the probability that s is one of the nearest neighbors of q and $P(s \notin q_{NN}) = 1 - P(s \in q_{NN})$. Considering all the objects, we then have:

$$H(q_{NN}) = \frac{1}{|\mathcal{O}|} \sum_{s \in \mathcal{O}} H_q(s) \tag{2}$$

where $\mathcal{O}$ is the given set of objects. Intuitively, this uses the average uncertainty of the objects for the query as the uncertainty of the query.

The above entropy formulation gives us a clue as to how uncertain we are in reaching the query result (when entropy is 0, we have obtained the deterministic answer, i.e., all the objects are known either in q_{NN} or not). Using this, we consider the question of which object to expose. This boils down to the conditional entropy with the condition of only object s_i being exposed, denoted $H(q_{NN}|s_i)$. The idea of the conditional entropy is that we assume we expose s_i, and then look at the possible resulting situations. After s_i is exposed, there are three cases:

1. We know s_i is definitely a nearest neighbor of q,
2. We know s_i is definitely *not* a nearest neighbor of q, and
3. We don't have a definitive conclusion.

Here "definitely" means $\tau = 1$, i.e., 100% probability that s_i is (or is not) one of the nearest neighbors of q. Since we do not know where s_i is in its safe region, each case has a corresponding probability, denoted $P(s_i \hat{\in} q_{NN}|s_i)$, $P(s_i \hat{\notin} q_{NN}|s_i)$ and $P(s_i \hat{\in} U|s_i)$, respectively. Note the $\hat{\in}$ and $\hat{\notin}$ notation means *definitely* or *deterministically* in and not in, respectively.

We will use the above three cases to define our conditional entropy and hence the information gain. First assume the following density function for distances:

$$pdf_s(d_{s,q}), \tag{3}$$

i.e., the distance probability density of an object s to query q. Due to our safe region assumption, there is a range $[l_s, u_s]$ for each object s for $pdf_s(d_{s,q})$ to be non-zero.

We will also use the cumulative probability function:

$$cpf_s(x) = \sum_{y \leq x} pdf_s(y), \tag{4}$$

i.e., the probability of an object s to be no further than the parameter value.

Note that in this paper we assume gridded space and discrete probability functions. The continuous case can be handled similarly, but not dealt with here.

Now, in order to find $P(s\hat{\in}q_{NN}|s)$, we just need to find all the minimum distances of all other objects in $\mathcal{O} - \{s\}$, i.e., $l^s_{min} = min_{s_i \in \mathcal{O}-\{s\}}(l_{s_i})$, and then we figure out the probability of s to be no greater than this minimum distance,

$$P(s\hat{\in}q_{NN}|s) = \sum_{l_s \leq x \leq l^s_{min}} pdf_s(x). \tag{5}$$

Clearly, if $l_s > l^s_{min}$, we default to 0.

Lemma 1: Equation (5) correctly computes the probability of finding s to be a definite nearest neighbor of q when exposing s.

For $P(s\hat{\notin}q_{NN}|s)$, we do the opposite, i.e., find the minimum of the upper bounds of all other objects, and then find the probability of s to be beyond this upper bound. In formula,

$$P(s\hat{\notin}q_{NN}|s) = \sum_{x > u^s_{min}} pdf_s(x) = 1 - cpf_s(u^s_{min}), \tag{6}$$

where $u^s_{min} = min_{s_i \in \mathcal{O}-\{s\}}(u_{s_i})$.

Lemma 2: Equation (6) correctly computes the probability of definitely ruling out s to be a nearest neighbor of q when exposing s.

The last item $P(s\hat{\in}U|s)$ can be found by 1 minus the values of Equations 5 and 6.

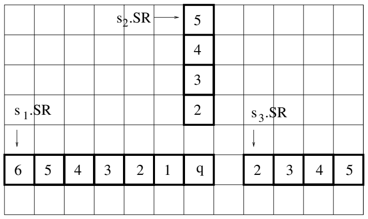


Fig. 2. Example used to intuitively show the meaning of the probabilities $P(s_i\hat{\in}q_{NN}|s_i)$, $P(s_i\hat{\notin}q_{NN}|s_i)$ and $P(s_i\hat{\in}U|s_i)$.

Using the example in Figure 2, we have $P(s_1\hat{\in}q_{NN}|s_1) = \sum_{l_{s_1} \leq i \leq 2} pdf_{s_1}(i) = 2/6$ since $l^{s_1}_{min} = 2$. Indeed, there is 2/6 probability that s_1 is in cell 1 or 2 and these are exactly the cases when s_1 is definitely one of the nearest neighbors to q. Likewise, we have $P(s_1\hat{\notin}q_{NN}|s_1) = 1/6$ since there is 1/6 probability s_1 is in cell 6 (note $u^{s_1}_{min} = 5$) and this is exactly the case there is no possibility for s_1 to be one of the nearest neighbors of q. Finally we have $P(s_1\hat{\in}U|s_1) = 1 - 2/6 - 1/6 = 3/6$.

We are now ready to define conditional entropy $H_q(s|s_i)$, i.e., the entropy of s being PoNN assuming s_i is exposed, as follows, where $s, s_i \in \mathcal{O}$ and s can be the same as s_i:

$$\begin{aligned} H_q(s|s_i) =& P(s_i\hat{\in}q_{NN}|s_i)H_q(s|s_i\hat{\in}q_{NN}) \\ &+ P(s_i\hat{\notin}q_{NN}|s_i)H_q(s|s_i\hat{\notin}q_{NN}) \\ &+ P(s_i\hat{\in}U|s_i)H_q(s|s_i\hat{\in}U), \end{aligned} \tag{7}$$

where $H_q(s|s_i\hat{\in}q_{NN})$ is the entropy of s assuming s_i is definitely a nearest neighbor of q after exposing s_i, and similarly for $H_q(s|s_i\hat{\notin}q_{NN})$, and $H_q(s|s_i\hat{\in}U)$. These entropy values are defined as in Equation (1) replacing with the appropriate probability values. Then the conditional entropy of the query is:

$$H(q_{NN}|s_i) = \frac{1}{|\mathcal{O}|} \sum_{s \in \mathcal{O}} H_q(s|s_i) \tag{8}$$

And the *information gain* of exposing s_i is then

$$G(q_{NN}, s_i) = H(q_{NN}) - H(q_{NN}|s_i) \tag{9}$$

Now we can completely describe line 4 of Algorithm 1 as follows:

Line 4: Calculate $G(q_{NN}, s_i)$ for each s_i in B and then pick the s_i with the greatest $G(q_{NN}, s_i)$ value (in case of tie, randomly pick one of). Move s_i from B to E.

Since $H(q_{NN})$ is fixed irrespective of which s_i to expose, to obtain the greatest G value is to get the smallest $H(q_{NN}|s_i)$ value among all objects s_i in B.

The probabilities needed to compute the entropies in Equation 7 are $P(s \in q_{NN}|s_i\hat{\in}q_{NN})$, $P(s \in q_{NN}|s_i\hat{\in}U)$ and $P(s \in q_{NN}|s_i\hat{\notin}q_{NN})$. These follow the same formula but with each conditional probability using a different pdf for s_i (while keeping the pdfs for all other objects untouched). Hence, our fundamental calculation is to calculate the probability $P(s \in q_{NN})$, given the pdf's of all the objects. But this is not difficult:

$$\begin{aligned} &P(s \in q_{NN}) \\ &\quad = \Sigma_{x \in [l_s, u_s]} \left(pdf_s(x) \cdot \Pi_{s_i \neq s} P((\text{distance of } s_i \text{ to } q) \geq x) \right) \\ &\quad = \Sigma_{x \in [l_s, u_s]} \left(pdf_s(x) \cdot \Pi_{s_i \neq s} (\Sigma_{y \geq x} pdf_{s_i}(y) \right) \\ &\quad = \Sigma_{x \in [l_s, u_s]} \left(pdf_s(x) \cdot \Pi_{s_i \neq s} (1 - cpf_{s_i}(x) + pdf_{s_i}(x)) \right), \end{aligned} \tag{10}$$

Following the above discussion, to calculate $P(s \in q_{NN}|s_i\hat{\in}q_{NN})$, we keep all the pdf's for $s_j \neq s_i$ and project/restrict pdf_{s_i} to the range $[l_{s_i}, l^{s_i}_{min}]$. We then use Equation (10). Other conditional probability calculations can be done similarly, and hence we obtain the conditional entropies for each object with Equation 7 and the conditional entropy for the query with Equation 8. When the safe region is a rectangle, the pdf of the distances of the cells to the query location can be derived without much difficulty.

Theorem 1: Equation (10) correctly computes the probability of s to be a PoNN, i.e., $P(s \in q_{NN})$. The computational complexity of the calculation is $O(|u_s - l_s| * |\mathcal{O}|)$.

Now we have a procedure for lines 3 and 7 in Algorithm 1:

Line 3: While (for each $s \in \mathcal{O}$, $P(s \in q_{NN}) < \tau$)

Line 7: For each $s \in \mathcal{O}$, return s if $P(s \in q_{NN}) \geq \tau$

The computational complexity of both lines 3 and 7 is $O(max_r * |\mathcal{O}|^2)$, where $max_r = max_{s\in\mathcal{O}}(|u_s - l_s|)$, following the complexity result in Theorem 1 and the fact that we need to loop through all objects in $\mathcal{O}$ in the worst case.

Theorem 2: The computation complexity of Algorithm 1 is $O(max_r * |\mathcal{O}|^3)$, where $max_r = max_{s\in\mathcal{O}}(|u_s - l_s|)$.

The complexity result above is derived directly from Theorem 1 and the analysis of lines 3, 4 and 7. Indeed, the worst case is that we iterate over all the objects in $\mathcal{O}$ with the while statement. Then in each iteration, lines 3, 4 and 7 dominate the complexity with each being $O(max_r * |\mathcal{O}|^2)$. Hence, the overall complexity is $O(max_r * |\mathcal{O}|^3)$.

This complexity result does not look promising when the size of $\mathcal{O}$ is large. However, in practice, we only need to consider the objects that are close to the query location. Indeed, we can filter out an object s if there is another one s' such that $l_s > u_{s'}$ since s cannot be a PoNN in any case. Another source of speedup is that the calculation of Equation (10) may be done more quickly when intermediate results are kept around in the algorithm because much work is repeated when considering each different s. More sophisticated speedup method may use the τ value to reduce the work of lines 3 and 7. However, we do not pursue these lines of optimization in this paper. Instead we are more interested in seeing how the information gain-based approach works in reducing the number of exposures.

V. Experimental Setup

Our experiments were conducted using one query location and randomly placed data objects with random safe regions. Each experiment was run 100 times. In the next section, the average and standard deviation of the number of exposures needed to answer the PoNN query are shown for our algorithm and three other algorithms. The default number of objects was 40 and the default probability threshold (τ) was 0.8.

The data objects were placed in the grid using a normal distribution with the query's location as the mean and a default standard deviation of 6 (this was varied in an experiment). The object's coordinates were generated independently along the two dimensions with the same distribution.

The x and y lengths of the safe regions of the data objects were generated around the object center also using a normal distribution. The default mean and standard deviation were both set to 5 (the default mean was varied in an experiment). Once safe regions were assigned, object positions within their safe regions were randomized using a uniform distribution.

In our experiments we compared our algorithm against three other algorithms. The algorithms were as follows:

IGBE This is our information gain-based exposure algorithm.

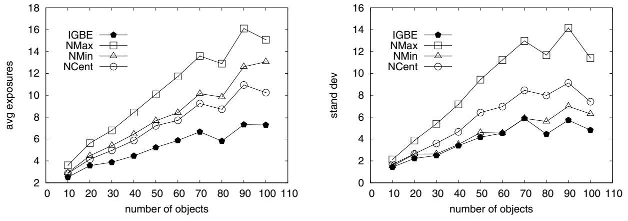
NMin This algorithm is Algorithm 1 with Line 4 changed to: Pick $s \in B$ where s has the nearest minimum distance to the query location. Move s from B to E.

NMax Similar to NMin except we pick $s \in B$ where s has the nearest maximum distance to the query location.

NCent Similar to NMin except we pick $s \in B$ where s has the nearest safe region center to the query location.

VI. Experimental Results

In this section we report the results of four experiments. The following were varied in one experiment each: the number of objects; τ; the deviation in the data object locations and the mean safe region size.

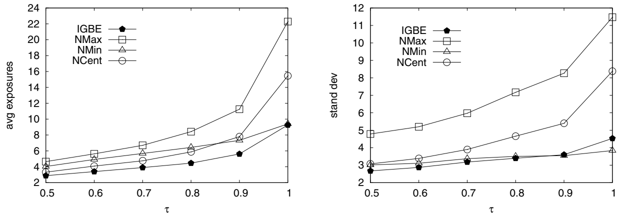


(a) Average number of exposures (b) Standard deviation of exposures

Fig. 3. Results of varying number of objects.

First, we vary the number of objects. The results are shown in Figure 3. In Figure 3(a) the average number of exposures needed to answer the query is shown. The results show IGBE outperforms the three naive algorithms, for all values tested. The performance difference between IGBE and the other algorithms increases as the number of objects increases. This is because as the number of objects increases so does the uncertainty of which object is the nearest neighbor, which the more sophisticated IGBE can handle better.

Figure 3(b) shows the standard deviation (SD) of the algorithms. Smaller SD means a more stable performance. The results show the three other algorithms have higher deviations in its effectiveness than the IGBE algorithm does. This is because, in contrast to IGBE, these algorithms are beneficial in specific scenarios (or "guessed right") but perform poorly in other scenarios, whereas IGBE is a more general approach working well in most cases. It is also interesting to note that NMin has a lower SD than NCent does, and NCent in turn has lower SD than NMax does. This shows that in most cases NMin is more stable than NCent, and that taken together with a smaller average number of exposures, NMin works better in most cases than NCent does (not just having a better average). The same can be said comparing NCent and NMax.



(a) Average number of exposures (b) Standard deviation of exposures

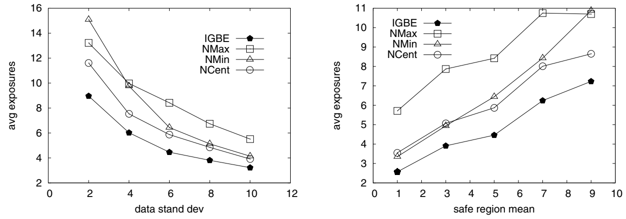
Fig. 4. Results of varying threshold.

Now we vary the probability threshold (τ) needed to give an object as the answer to the PoNN query. Figure 4(a) shows that in terms of average exposures IGBE again outperforms the other algorithms at all values. At $\tau = 1$, the result is very similar for NMin (average exposures = 9.4) and IGBE (average exposures = 9.26). When $\tau = 1$, we expected NMin to perform well. This is because in most cases, the object (s_{nmin}) with the smallest minimum distance must be exposed since otherwise we can't know for sure that no other object is closer than s_{nmin} is. As τ decreases so does the necessity to expose the object with the smallest minimum. This is why NMin begins to perform worse than NCent once $\tau < 0.9$. Note in all cases, IGBE is still the best performer.

The standard deviation of the results of the different runs is shown in Figure 4(b). Similar to Figure 3(b) IGBE and NMin are the most consistent. For $\tau = 1$, NMin is more consistent than IGBE. This is because IGBE uses a probabilistic approach so it will, on average, give better or the same number of objects to expose. However, this means there will be more variability than for NMin which uses a good selection method for $\tau = 1$ and ignores the probabilities of the different distances.

Now, we vary the standard deviation of the data objects' center locations (the mean is the query's location). The results are shown in Figure 5(a). IGBE again consistently outperforms the other algorithms. The difference between IGBE and the other algorithms is greater when the data's standard deviation is low. This is because the data objects will be more crowded around the query making it less obvious which object(s) should be exposed (in which case IGBE shines). NMin performs worse when the data's standard deviation is 2 than elsewhere because most of the objects will have a small minimum distance, so the size of the safe region becomes more important (which NMin ignores).

The standard deviation results for the experiments in Figure 5 are similar to those above and omitted here.



(a) Varying data standard deviation

(b) Varying SR mean length

Fig. 5. Results of varying data standard deviation and SR mean.

Finally, we vary the mean of safe region range. The results are shown in Figure 5(b). On average IGBE outperforms the other algorithms. NMin is the second best for the smaller safe region means and NCent is second best for the other values. This is because when the safe regions are smaller exposing the object with the smallest minimum is more likely to have its actual location closer than the other objects' safe regions.

VII. Conclusion

In this paper, we have shown a detailed algorithm that selectively exposes object locations in answering the probabilistic one-of nearest neighbor query. The interesting findings include the promise of the information gain-based approach. The experimental results show the positive effect of our approach.

From a theoretical perspective, the computational complexity of the algorithm is cubic in the number of objects. (Note that the straightforward solutions we compared with all have the same computational complexity, and hence the inherent complexity may not be in the information gain but rather in the probabilistic calculation.) However, as mentioned earlier, there are many opportunities to reduce the calculation time. In our experimental results, we focused on the number of exposures. It will be an interesting future direction to study the effect of various optimization techniques.

A general future direction, which is what we are actively pursuing, is to see how the information gain-based approach works for various other kinds of queries. Also, how to assign safe regions together with query evaluation consideration is an interesting direction.

Acknowledgment: The work of Wang was supported by the National Science Foundation, while working at the Foundation. Any opinion, finding, and conclusions or recommendations expressed in this material, are those of the authors and do not necessarily reflect the views of the Foundation.

References

[1] G. Beskales, M. A. Soliman, and I. F. IIyas, "Efficient search for the top-k probable nearest neighbors in uncertain databases," in *VLDB Conference*, 2008, pp. 326–339.

[2] J. Chen and R. Cheng, "Efficient evaluation of imprecise location-dependent queries," in *ICDE Conference*, 2007, pp. 586–595.

[3] R. Cheng, L. Chen, J. Chen, and X. Xie, "Evaluating probability threshold k-nearest-neighbor queries over uncertain data," in *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*. New York, NY, USA: ACM, 2009, pp. 672–683.

[4] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," in *SIGMOD Conference*, 2003, pp. 104–130.

[5] ——, "Querying imprecise data in moving object environments," *TKDE*, vol. 16, no. 9, pp. 1112 – 1127, 2004.

[6] H. Hu, J. Lee, and D. Lun, "PAM: an efficient and privacy-aware monitoring framework for continuously moving objects," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 3, pp. 404–419, 2010.

[7] H. Hu, J. Xu, and D. L. Lee, "A generic framework for monitoring continuous spatial queries over moving objects," in *Proceedings of SIGMOD*, 2005, pp. 479–490.

[8] D. V. Kalashnikov, S. Prabhakar, and S. E. Hambrusch, "Main memory evaluation of monitoring queries over moving objects," *Distributed and Parallel Databases*, vol. 15, no. 2, pp. 117–135, 2004.

[9] H.-P. Kriegel, P. Kunath, and M. Renz, "Probabilistic nearest-neighbor query on uncertain objects," in *Advances in Databases: Concepts, Systems and Applications*, 2008, pp. 337–348.

[10] V. Ljosa and A. Singh, "ALPA: indexing arbitrary probability distributions," in *ICDE Conference*, 2007, pp. 945–955.

[11] S. Prabhakar, Y. Xia, D. V. Kalashnikov, W. G. Aref, and S. E. Hambrusch., "Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects," *IEEE Transactions on Computers*, vol. 51, no. 10, pp. 1124–1140, 2002.

[12] Y. Tao, R. Cheng, X. Xiao, W. K. Ngai, B. Kao, and S. Prabhakar, "Indexing multi-dimensional uncertain data with arbitrary probability density functions," in *VLDB '05: Proceedings of the 31st international conference on Very large data bases*. VLDB Endowment, 2005, pp. 922–933.

[13] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha, "Updating and querying databases that track mobile units," *Distrib. Parallel Databases*, vol. 7, no. 3, pp. 257–387, 1999.