Group Nearest Neighbor Queries for Fuzzy Geo-Spatial Objects

Novia Nurain[1], Mohammed Eunus Ali[2], Tanzima Hashem[3], and Egemen Tanin[4]
[1,2,3]Dept. of CSE, Bangladesh University of Engineering Technology, Dhaka, Bangladesh
[4]Dept. of CIS, University of Melbourne, Victoria, Australia
[1]novia@cse.uiu.ac.bd, [2,3]{eunus,tanzimahashem}@cse.buet.ac.bd, [4]etanin@unimelb.edu.au

ABSTRACT

A geo-spatial object with non-deterministic boundaries and compositions is commonly known as a fuzzy geo-spatial object. The advancement of data capturing devices such as sensors and satellite imaging technologies enable us to identify fuzzy geo-spatial objects from a large and complex image of an area. The nearest neighbor (NN) query processing on fuzzy objects, which finds the nearest fuzzy object to the given query point, has been addressed recently. In this paper, we envision a new set of applications that require finding the nearest fuzzy geo-spatial object for a group of fuzzy geo-spatial *query* objects. For example, when an oil spill occurs at a sea, the primary concern of an emergency response planner is to find an environmentally sensitive area, e.g., port or harbor, that will be affected the most by the oil spill. To support such applications, in this paper, we propose a new query type, called a fuzzy group nearest neighbor (FGNN) query. Given a set of fuzzy geo-spatial data objects, and a group of fuzzy geo-spatial query objects, an FGNN query returns a fuzzy geo-spatial object that minimizes the aggregate distance to the group. To solve FGNN queries, we develop an efficient technique in this paper. Our extensive experimental study reveals the efficacy and efficiency of our proposed technique.

Categories and Subject Descriptors: H.2 [Database Management]: Spatial Database

General Terms: Algorithms, Experimentation

Keywords: Fuzzy geo-spatial objects, geographical information system, group nearest neighbor query.

1. INTRODUCTION

Recent growth in the availability of geo-spatial data due to the advancement of sensors and satellite imaging technologies has introduced a new type of object with non-deterministic boundaries and compositions. Figure 1 exhibits a satellite image of islands in a coastal area. The boundaries of these islands cannot be identified accurately.

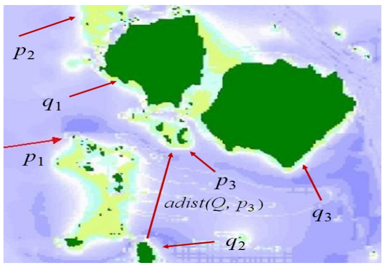


Figure 1: A set of oil spilled areas $\{q_1, q_2, q_3\}$ and a set of unaffected areas $\{p_1, p_2, p_3\}$, here, p_3 is the unaffected area that will get affected next by the group of oil spilled areas

Fuzziness lies in the composition of such objects, i.e., a point may or may not belong to an object. Moreover, not all parts of an object are equally important (e.g., kernel vs. boundary). Fuzziness may also lie in the data collected from the sensor readings as sensor readings are not always precise. In addition, measurements from multiple sensors may vary even if they are located close to each other. For example, consider a scenario where we want to determine whether an inhabited island in a coastal area is affected by a sudden oil spill at a sea. Imprecise readings of sensors of that area give blurred information about different borders of islands. Therefore, to represent fuzziness of these objects, we assign different probabilities as membership values to different points of an island, e.g., high probabilities are assigned to the points on the kernel, and low probabilities are assigned to the points on the boundary. These objects are known as *fuzzy geo-spatial objects.*

Recently, query processing, in particular nearest neighbor (NN) queries [3], on fuzzy objects has received attention due to its wide range of applications in geo-spatial and medicine domain. For a given query fuzzy object and a set of fuzzy data objects, an NN query finds the data object that minimizes the Euclidean distance from the query object.

We envision a new set of applications that require finding the nearest fuzzy geo-spatial object with respect to a group of fuzzy geo-spatial *query* objects. Let us consider the following scenario. An oil spill occurs at a sea and an emergency response planner tries to find an environmentally sensitive area, e.g., a port or a harbor, from a set D of areas that will be affected the most due to the combined impact of a group Q of oil spilled areas. This problem can be modeled as an instance of the group or aggregate nearest neighbor problem [5], i.e., finding an area p from D that minimizes the total distance, $adist(Q, p)$ with respect to a group Q of affected areas, i.e., oil spilled areas. Here, the aggregate distance is the *summation* of distances to the data object from the group of query objects. Besides, a response planner may also need to find an unaffected area that has the highest chance to get affected by *all* the oil spilled areas at the earliest time. In this case, we need to minimize the *maximum* distance from all the affected areas to an unaffected area. Such queries can enhance the functionality of an oil spill trajectory modeling tool [1].

Figure 1 exhibits an example - where the affected areas, i.e., oil spilled areas $\{q_1, q_2, q_3\}$ are query objects and the unaffected areas $\{p_1, p_2, p_3\}$ are the data objects. In this example, p_3 minimizes the total distance (i.e., SUM) with respect to all the query objects, and thus p_3 is the area that will be affected most by the combined impact of all the oil spilled areas.

Motivated by the above applications, we propose a new query type called a fuzzy group nearest neighbor (FGNN) query. Given a set of fuzzy geo-spatial data objects, a group of fuzzy geo-spatial query objects, an FGNN query returns a fuzzy geo-spatial object that minimizes the aggregate distance to the group. Besides, if a user is interested in finding k nearest data objects for the group, then we term the query as an FkGNN query.

A straightforward application of the existing group nearest neighbor algorithm for point objects [5, 6] for answering an FGNN query requires considering every point in a fuzzy geo-spatial object independently, where each fuzzy geo-spatial object is represented as a large collection of points. This will incur a higher computational and I/O overhead. Besides, the existing group nearest neighbor method for extended objects [7] considers only the query objects as extended regions instead of the entire data set. Moreover, we need to incorporate a user-specified probability threshold for our envisioned FGNN queries. This is because since all parts of a fuzzy geo-spatial object are not equally important, a user may want to select a fraction of the object for query processing by specifying a probability threshold. For example, an emergency response planner may need to find a region of a port or harbor that will be affected by specific regions of oil spilled areas.

In this paper, we propose efficient techniques for the processing of an FGNN query over fuzzy geo-spatial objects. To support the query processing approach, we first approximate each fuzzy geo-spatial object using a bounding rectangle based on a probability threshold defined by a user. Then we compute the group nearest neighbor query with respect to these rectangles to find a candidate answer set. Finally, we refine the answers by computing the actual aggregate distances of the candidate fuzzy geo-spatial objects.

In summary, the contributions of this paper are as follows.

- We formulate the problem of group nearest neighbor query for fuzzy geo-spatial objects.
- We propose efficient approaches for processing an *fuzzy group nearest neighbor (FGNN)* query, to determine the nearest neighbor with respect to a set of query objects for a specific probability threshold.
- We conduct an extensive experimental study to show the effectiveness and efficiency of our approaches.

2. MODELS AND PROBLEM DEFINITIONS

In this section, we first present the preliminaries of modeling fuzzy objects and then formulate the FGNN query.

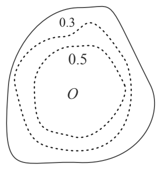


Figure 2: A fuzzy object O

2.1 Modeling of Fuzzy Objects

A fuzzy geo-spatial object O is represented as a set of points o along with a membership value $\delta_O(o)$, which indicates the probability of o belonging to O [2, 3].

$$O = \{(o, \delta_O(o)) | \delta_O(o) > 0\}$$

Since a fuzzy geo-spatial object is represented as a collection of points with different probabilities, not all parts of the object are equally important. Thus, we define the term α-cut. Given a fuzzy geo-spatial object O and a probability threshold α, the α-cut contains only those points that have membership values greater than α. For example, the set $O_\alpha = \{o \epsilon O | \delta_O(o) >= \alpha\}$ is the α-cut for a fuzzy object O. Fig. 2 demonstrates a fuzzy object O with α-cuts at two probabilities 0.3 and 0.5.

2.2 Problem Setup

When a group of query objects is involved in a query and returns the object that *minimizes* the aggregate distance for the group, the query is known as the group nearest neighbor (GNN) query. In this paper, we focus on two aggregate functions SUM and MAX as these address two key potential applications in the geo-spatial domain.

Given a set of fuzzy geo-spatial objects, and a group of fuzzy query objects, a fuzzy group nearest neighbor (FGNN) query returns a fuzzy geo-spatial object that minimizes the aggregate distance for that group. Besides, for an FGNN query, we allow the user to select parts of the fuzzy object instead of the entire object, specifying a probability threshold. We formally define an FGNN query as follows.

Definition 2.1. *Given a data set D of fuzzy geo-spatial objects, a set of n fuzzy query objects $Q = \{q_1, q_2, ...q_n\}$, a user specified probability threshold $\alpha \in [0, 1]$, an FGNN query returns a fuzzy geo-spatial object O such that for any $O^* \in D - \{O\}$, $adist_\alpha(O, Q) \leq adist_\alpha(O^*, Q)$.*

Here, $adist_\alpha$ is the aggregate distance at a probability threshold α. For a group of n fuzzy query objects $Q = \{q_1, q_2, ..., q_n\}$ and a fuzzy data object O from the data set D of fuzzy geo-spatial objects, we define the aggregate distance at a user specified probability threshold α as follows.

$$adist_\alpha(O, Q) = g_{i=1}^{n}(dist_\alpha(O, q_i)) \quad (1)$$

Here, g is an aggregate distance function and it can be SUM or MAX. Besides, $dist_\alpha$ is the distance between two fuzzy objects at a probability threshold α. For two fuzzy objects X and Y, the $dist_\alpha$ is defined as follows.

$$dist_\alpha(X, Y) = min_{(x,y) \in X_\alpha \times Y_\alpha} ||x - y|| \quad (2)$$

Here, X_α and Y_α are the α-cut of fuzzy objects X and Y respectively. $||x - y||$ is the Euclidean distance between two points x and y, where x and y belongs to X_α and Y_α respectively. In general, a user may be interested in finding k fuzzy data objects that have the k smallest aggregate distances from the group, which is known as fuzzy k group nearest neighbor (FkGNN).

3. RELATED WORK

To capitalize the potential of enriched geo-spatial data in GIS, recent research has focused on modeling and developing efficient techniques that find answers for different queries over fuzzy objects. Processing the nearest neighbor (NN) query on fuzzy objects is studied in the literature [3, 4].

In [3], the authors have proposed solutions to solve k-NN and range queries over fuzzy objects. For both types of queries, the required probability threshold of a query can be a fixed value or a range denoting a probability interval. The first one is known as a *single probability threshold* based query and the second one is known as a *continuous probability threshold* based query. Here, the authors have provided an efficient approach to compute the minimum bounding rectangle (MBR) for each fuzzy object. In addition, to reduce the number of distance computation the authors have adopted the idea of lazy probe introduced in [9]. We adopt the MBR computation and the lazy probe method for the processing of our envisioned group nearest neighbor (GNN) query for fuzzy geo-spatial objects.

The GNN query was first proposed by Papadias et. al. [5, 6] for point objects, which returns data points that minimize the aggregate distances (i.e., SUM, MAX, and MIN) with respect to a set of query points. Papadias et. al. [6] have proposed three different techniques: multiple query method (MQM), single point method (SPM), and minimum bounding method (MBM) to evaluate a GNN query with the assumption that all the points are indexed using an R-tree. MQM performs an incremental search (e.g., depth-first or best-first) to find the nearest data points for each of the query point and then compute the aggregate distance for each of the retrieved data points. On the other hand, SPM computes the centroid of the query objects and incrementally retrieves the nearest data points in order of their minimum distances from the centroid. Finally, MBM incrementally retrieves the nearest objects in order of their minimum aggregate distance from all the query points. Experimental results reveal the superiority of MBM over SPM and MQM.

A flexible version of the GNN query is proposed by the authors in [11]. Their proposed flexible aggregate similarity search FANN query finds the most similar data objects with respect to a subgroup of the query objects in both low and high dimensions.

The studies of [6, 11] consider the locations of the objects as static. However, Elmongui et.al. [10] address the GNN problem for moving objects in spatio-temporal data stream. Jaijia et. al. [12] study the problem of GNN query on the uncertain database where the location of the data object is uncertain. They propose a probabilistic GNN (PGNN) query that finds a nearest uncertain data object that has the probability to be in the GNN results exceeds a user-specified threshold. For our envisioned FGNN query, a fuzzy object is represented as a collection of points with probabilities. On the other hand, an uncertain object is represented as a single point that may be located at any point within a region with arbitrary probability distributions.

All the above studies have considered point data for GNN queries. Hashem et.al. [7] have proposed an efficient algorithm of GNN queries for extended objects that preserve location privacy of users. They have considered only the query objects as regions.

All the previous approaches for GNN cannot directly be applied to process an FGNN query. This paper is the first study to propose efficient techniques for processing an GNN query over fuzzy geo-spatial objects.

4. PROCESSING FGNN QUERIES

We assume that the data objects are indexed using an R^*-tree in the database. In the leaf node of an R^*-tree [8], we store the MBR of the α-cut at a probability threshold 1 along with a pointer referring to the actual location of the fuzzy object on the disk. We use the algorithm presented in [3] for the computation of MBRs at different user specified probability thresholds. Due to space limitation, we omit the details about the computation. To improve the performance of an FGNN query, we utilize the upper and lower bounds of the actual aggregate distance (Equation 1). For a set of n query objects, $Q = \{q_1, q_2, ..., q_n\}$, a probability threshold α, and a fuzzy geo-spatial object O, the upper and lower bounds of the aggregate distance are defined as follows.

$$LB\_adist_{\alpha}(O, Q) = g_{i=1}^{n}(MinDist(MBR_O(\alpha), MBR_{q_i}(\alpha))) \quad (3)$$

$$UB\_adist_{\alpha}(O, Q) = g_{i=1}^{n}(MaxDist(MBR_O(\alpha), MBR_{q_i}(\alpha))) \quad (4)$$

Here, g could be SUM or MAX.

4.1 Basic approach

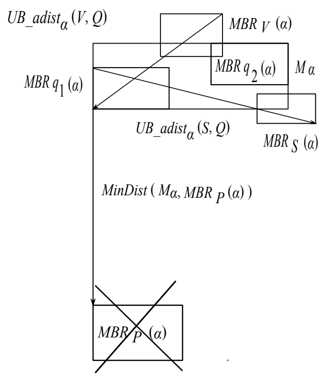
In this subsection, we illustrate our first algorithm FGNN-Basic for processing an FGNN query. Our algorithm follows the best first search technique to access the data objects incrementally and to find the results for an FkGNN query. We maintain a priority queue W_q to store the intermediate or leaf nodes. We start our search from the root node and insert it in W_q along with its lower and upper bounds of the aggregate distances, $LB\_adist_{\alpha}(Root, Q)$ and $UB\_adist_{\alpha}(Root, Q)$ respectively. The elements of W_q are stored in order of their lower bound of aggregate distance with respect to query objects.

In each iteration, we encounter any of the three different types of elements (i.e., an intermediate node, a leaf node, or an object) from the priority queue W_q and we treat each of them differently.

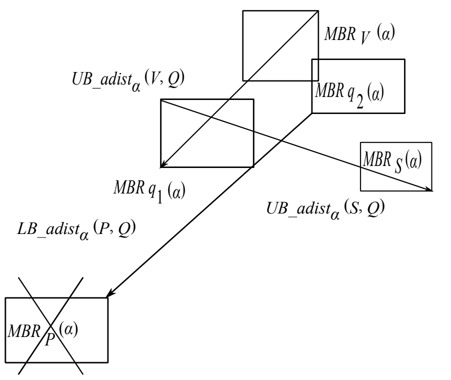
Whenever we encounter an intermediate node, we do not immediately insert its child nodes into the priority queue. Rather, for each child node, we check if the node contains the candidate answers for an FGNN query with respect to query objects. Hence, we adopt two layer pruning conditions of [7] to prune the unqualified nodes. For this purpose, we maintain an array *distmax* of size k, to store at most k maximum distances found so far, which is initialized to ∞.

CONDITION 1. *An R^*-tree node P can be pruned if it satisfies, $n \times MinDist(M_{\alpha}, MBR_P(\alpha)) > distmax[k]$ for the aggregate function $g = SUM$, and if it satisfies $MinDist(M_{\alpha}, MBR_P(\alpha)) > distmax[k]$ for the aggregate function $g = MAX$.*

Here, n is the number of query objects and M_{α} is the minimum bounding box that encloses the MBRs of the query objects. The minimum distance between the MBR of P and the MBR of the group of query objects at a probability threshold α is denoted by $MinDist(M_{\alpha}, MBR_P(\alpha))$. Fig. 3a illustrates this condition using an example for an FkGNN query with respect to a group of query objects $Q = \{q_1, q_2\}$ with k=2 and the aggregate function g = MAX at a probability threshold α. At an intermediate state of the traversal, let us consider node V and node S are dequeued from the



(a) Condition 1



(b) Condition 2

Figure 3: Pruning conditions

priority queue. For example, the lower and upper bounds of the aggregate distances of these nodes are $\{V, 30.2, 60.45\}$, $\{S, 45.5, 65.34\}$, and hence at this stage $distmax[1] = 60.45$ and $distmax[2] = 65.34$. Next, we dequeue another intermediate node P from the priority queue, we first compute the minimum distance between the MBR of node P and the MBR that encloses the group of query objects. Suppose this distance is 71.32. We then compare this minimum distance with the 2^{nd} maximum distance found so far, which is stored in $distmax[2]$. As the minimum distance is larger than the maximum distance found so far, then it is obvious that the node P cannot be in the top $k = 2$ solution. Therefore, the node P can be pruned safely.

However, for the first pruning rule, we consider the MBR that encloses the given group of query objects. Though this stage involves less distance computation, it may require visiting extra nodes that cannot contain qualifying objects for an FGNN query. Therefore, we introduce a second pruning condition, which presents a tighter bound and avoids retrieving unnecessary objects.

CONDITION 2. *An R^*-tree node P can be pruned if it satisfies, $LB\_adist_\alpha(P,Q) > distmax[k]$.*

Here, $LB\_adist_\alpha(P,Q)$ is the lower bound of the aggregate distance computed using Equation (3). Fig. 3b illustrates this pruning condition with an example. Without loss of generality, we reconsider our previous example for explaining the pruning Condition 2. At an intermediate state, when a node P is dequeued from the priority queue, we try to filter it using the first pruning rule. If P is not pruned by the first pruning condition, then we compute the lower bound of the aggregate distance between P and Q (i.e., $LB\_adist_\alpha(P,Q)$) using Equation (3). For example, $LB\_adist_\alpha(P,Q) = 85.75$. We compare this distance with $distmax[2]$ which is 65.34. The lower bound of aggregate distance is larger than the maximum distance found so far, we conclude that the node P cannot be in the top 2 solution.

Next, if we encounter a leaf node, then we retrieve the corresponding object from the disk and reinsert it into the priority queue. Again, if we encounter an object, then we insert this object in the result set. Finally, our algorithm for F*k*GNN query terminates when the priority queue is empty or the k nearest objects are found.

Following theorem proves the correctness of our algorithm.

THEOREM 1. *If k is the number of required data objects for an FkGNN query with respect to a set $Q = \{q_1, q_2, .., q_n\}$ of n query objects, then the result set of our proposed approach includes all the data objects that have the i^{th} ($1 \leq i \leq k$) minimum value for the aggregate function g (SUM or MAX) with respect to Q.*

Proof. (By contradiction) Assume that O^* is a data object that does not belong to the result set, but has the i^{th} minimum value ($1 \leq i \leq k$) for the aggregate function g (SUM or MAX) with respect to Q. O^* does not belong to the result set because it has been pruned before inserting into the priority queue. However, O^* is only pruned if it satisfies the pruning Condition 1 or Condition 2, which again indicates that the lower bound of the aggregate distance of O^* with respect to Q (i.e., $LB\_adist_\alpha(O^*, Q)$) is greater than the maximum aggregate distance found so far (i.e., $distmax[k]$). It contradicts with our assumption that the object O^* is one of the k FGNNs with respect to Q. □

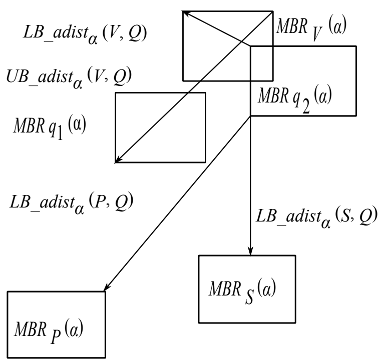


Figure 4: Delaying the probing of leaf nodes

4.2 Performance improvement using delay probe

Our proposed basic algorithm for an F*k*GNN query retrieves objects from the disk and computes the actual aggregate distance (Equation (1)) for all these objects. However, the computation of the aggregate distance is very time consuming, which results in a high query processing cost. To avoid such high computational overhead, we deploy the idea of delay probe presented by Kriegel et.al. [9], which helps us to determine a true hit without retrieving the actual object from the disk and reduces the I/O overhead significantly. We name this new algorithm as FGNN-DP. The basic idea is to include some objects in the result set based on the upper bound of aggregate distance, which is relatively easier to compute than the actual aggregate distance. Hence, we require another temporary priority queue T to store at most k leaf nodes/objects.

The idea of delay probing is illustrated in Fig. 4 with an example. Suppose an F*k*GNN query with $k = 2$ is issued for a group of $n = 2$ query objects $Q = \{q_1, q_2\}$, aggregate function $g = \text{MAX}$ at a probability threshold α. At an intermediate stage of the search, the priority queue W_q contains the leaf nodes V, S, P and the temporary queue T is empty. At first, the leaf node V is dequeued from the queue. Since it has the smallest $LB\_adist_\alpha(V,Q)$ it is reinserted into T. The next element is dequeued from the queue W_q is S. Here, the lower bound of aggregate distance of S is less than the upper bound of aggregate distance of V, hence S is inserted into the temporary queue T as well. Finally, the leaf node P is dequeued from the queue W_q and found that the lower bound of aggregate distance of P is greater than the upper bound of aggregate distance of an element V in temporary queue T. Hence, V is better than any other objects in the priority queue W_q. V is removed from the temporary queue T and inserted directly to the result set.

Table 1: Experiment setup

Parameter	Range	Default
Data set size	5K, 10K, 20K, 50K	20K
k	4, 8, 16, 32	20
Group size	4, 16, 32, 64	32
Probability threshold	0.3, 0.5, 0.7, 0.9	0.6
Point distribution	Uniform, Zipfian	Uniform
Area of query space	10%, 20%, 30%, 40%	30%

If we encounter a leaf node from the temporary queue T, only then we retrieve the corresponding object from the hard disk and compute the actual aggregate distance and finally reinsert the object again in the priority queue W_q. Again, if we encounter an object from the temporary priority queue T then we insert the object into the result set.

5. EXPERIMENTAL EVALUATION

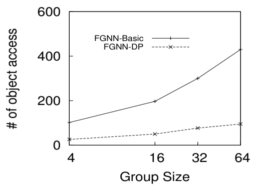
In this section, we present an extensive experimental study to evaluate the performance of our proposed algorithm. We generate synthetic data sets to resemble the scenario of our oil spill modeling applications where an emergency rescue planner needs to determine an environmentally sensitive area that would be affected most by the impact of a group of oil spilled areas during an oil spill at a sea. In this scenario, we need to model each affected/unaffected area as a fuzzy geo-spatial object. We consider each object as a circle containing n points using both uniform and Zipfian distributions. We then generate N such objects and distribute them into 100×100 square units. The different probabilities of the points are generated by the Gaussian distribution with mean at the center and standard deviation of 0.5. The values are normalized across 0 to 1. We evaluate the number of object access from the disk and the running time to measure the efficiency of our algorithms under different settings of parameters (Table 1). For each set of experiments, we evaluate 30 randomly generated groups of query objects and present the average experimental results. We implement all the algorithms in C++ and run experiments on a desktop with Core i5 2.40 GHz CPU and 2 GB memory.

5.1 Effect of varying group size

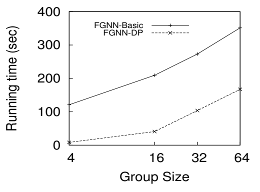
We study the impact of group size on the performance of FGNN query by varying the group size using 4, 16, 32, and 64 and measuring the required running time along with the number of object access from the disk. Fig. 5 depicts the number of objects accessed and the running time of FGNN query for MAX (a–b) and SUM (c–d). Fig. 5a and 5c depict that all the algorithms need to access more objects from the disk with an increase in the group size. For example, for an increase of the group size from 4 to 16, the number of object access increases approximately 1.6 and 1.7 times, respectively, for aggregate function MAX and SUM of all the algorithms. The reason behind such behavior: we expect that an increase in the group size increases the k^{th} maximum aggregate distance found so far in our array *distmax* and hence there is more chance that less number of nodes or data objects will be pruned. Since more objects are accessed, and more aggregate distances are computed we find that the running time also increases for larger group size (Fig. 5b, 5d). The superiority of our proposed FGNN-DP is evident for larger group size.

5.2 Effect of varying k

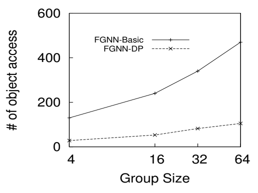
In this set of experiments, we evaluate the impact of k on the performance of FGNN algorithms by varying k from 4 to 32. Fig.6 depicts that the number of object access and the running time increase as k increases for both MAX



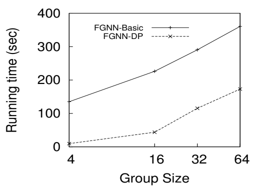
(a) FGNN Query (MAX)



(b) FGNN Query (MAX)

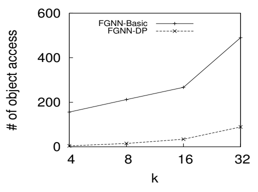


(c) FGNN Query (SUM)

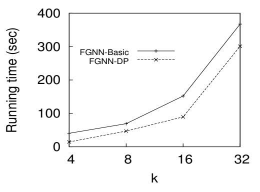


(d) FGNN Query (SUM)

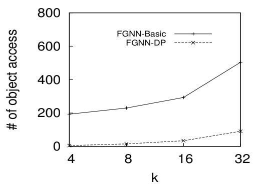
Figure 5: Effect of varying group size



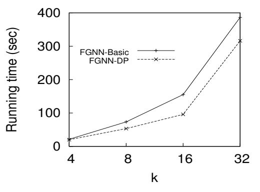
(a) FGNN Query (MAX)



(b) FGNN Query (MAX)



(c) FGNN Query (SUM)



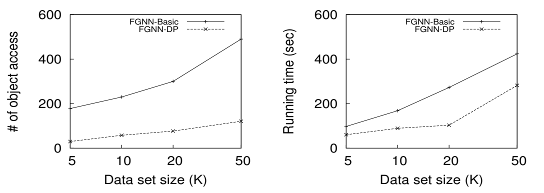
(d) FGNN Query (SUM)

Figure 6: Effect of varying k

(Fig. 6a, 6b) and SUM (Fig. 6c, 6d). The reason behind such increase: the k^{th} maximum aggregate distance stored in the array *distmax*, which we used for pruning the R^*-tree nodes and data objects increases with the increase in k. Therefore, less number of nodes and data objects will be pruned for larger value of k. We need to process more nodes or data objects. It requires more computation of aggregate distances that eventually increases the running time. Experimental results reveal that FGNN-DP is approximately 1.5 times faster and access on average 16.25 times less objects in comparison to FGNN-Basic algorithm for MAX (Fig. 6a, 6b). For the other aggregate function SUM (Fig. 6c, 6d), we see a similar trend as MAX.

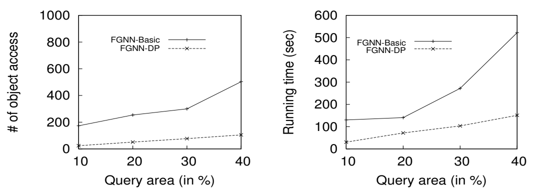
5.3 Effect of varying data set size

Fig. 7 depicts the impact of data set size on the performance of FGNN query for fuzzy objects by varying the number of objects using 5K, 10K, 20K, and 50K for MAX (a–b). Here, we see that the performance of FGNN query degrades as the data set grows. When the number of objects grows, the density of whole data space becomes higher, hence it makes more difficult to prune R^*-tree nodes or data objects. Therefore, FGNN query accesses more objects with an increase in the data set size (Fig. 7a). The running time of all the algorithms also increases as the number of aggregate distance computation increases with the growth of data set (Fig. 7b). For both aggregate functions, FGNN-DP outperforms the basic algorithm. Due to space limitation, we omit the graphs for SUM as the results are similar to MAX.



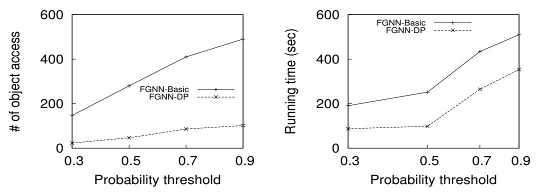
(a) FGNN Query (MAX) (b) FGNN Query (MAX)

Figure 7: Effect of varying data set size for MAX



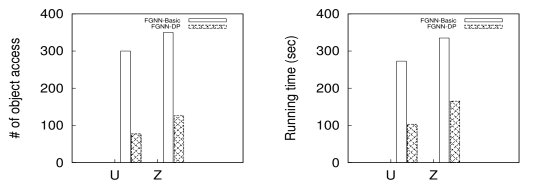
(a) FGNN Query (MAX) (b) FGNN Query (MAX)

Figure 8: Effect of varying the query space



(a) FGNN Query (MAX) (b) FGNN Query (MAX)

Figure 9: Effect of varying probability threshold for MAX



(a) FGNN Query (MAX) (b) FGNN Query (MAX)

Figure 10: Effect of varying point distribution for MAX

5.4 Effect of varying the query space

In this set of experiments, we vary the query area, i.e., the area to which the group of query objects is confined to as 10%, 20%, 30%, and 40% of the entire data space. Fig. 8 shows the required running time and object access for our proposed methods for aggregate function MAX (a–b). Here, we see that the processing time and object access increase with an increase of the query space for all the proposed methods. The reason behind such behavior: with the increase in the query space, the k^{th} maximum aggregate distance of the array *distmax* increases. Therefore, less number of R^*-tree nodes or data objects will be pruned, which result in access of more objects and aggregate distance computation and an increase in the processing time. Experimental results reveal the superiority of FGNN-DP over the basic approach, which is more obvious for a larger query space. Since the graphs for SUM exhibit similar results as MAX we omit them.

5.5 Effect of varying probability threshold

We study the impact of user specified probability threshold on the performance of FGNN query by varying the probability threshold using 0.3, 0.5, 0.7, and 0.9 for MAX. Fig. 9 depicts that the number of object access and running time increase with an increase of the probability threshold. Here, we see that FGNN-DP is 1.5–2.5 times faster and accesses 6–9 times less objects than FGNN-Basic for aggregate function MAX. Again, we omit the graphs for SUM, as the results exhibit similar trends.

5.6 Effect of varying point distribution

The impact of point distribution of a fuzzy object in the performance of FGNN query is illustrated for both uniform (U) and Zipfian (Z) point distribution for aggregate function MAX (Fig. 10). For uniform distribution the points have equal probability to locate anywhere within an object. On the other hand, in the Zipfian distribution more points are located around the kernel. Gaussian distribution of probabilities of points ensures that points around the kernel have a high probability threshold. Therefore, for computing the aggregate distance in Zipfian distribution we need to consider more points than the uniform distribution. Hence, the number of object access and the running time are high for Zipfian distribution (Fig. 10a, and 10b) compared to uniform distribution. We omit the graphs for SUM, as the results are similar to MAX.

6. CONCLUSION

In this paper, we have introduced a fuzzy group nearest neighbor (FGNN) query for fuzzy geo-spatial objects. An FGNN query has potential applications in geographical information systems. To efficiently process an FGNN query we have proposed two algorithms: FGNN-Basic and FGNN-DP. An extensive experimental study depicts the efficacy and efficiency of our algorithms. Experimental results reveal that for aggregate function MAX, FGNN-DP is approximately 2.64 times faster and accessed 3.8 times less objects compared to FGNN-Basic algorithm. On the other hand for aggregate function SUM, FGNN-DP is approximately 2.52 faster and accessed 4.14 times less objects than FGNN-Basic. In future, we plan to extend our work for other advanced queries such as reverse nearest neighbor queries, and skyline queries for fuzzy geo-spatial objects.

7. REFERENCES

[1] GNOME. http://response.restoration.noaa.gov/gnome, 2014.

[2] L. Zadeh. Fuzzy set. In *Inf. Control 8*, pp. 338–353, 1965.

[3] K. Zheng, P.C. Fung, and X. Zhou. K-Nearest Neighbor Search for Fuzzy Objects. In *SIGMOD*, pp. 699–710, 2010.

[4] K. Zheng, X. Zhou, and P.C. Fung. Spatial Query Processing for Fuzzy Objects. In *The VLDB Journal*, 21:729–751, 2012.

[5] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui. Aggregate nearest neighbor queries in spatial database. In *TODS*, 30(2):529–576, 2005.

[6] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis. Group Nearest Neighbor Queries. In *ICDE*, pp. 301–310, 2004.

[7] T. Hashem, L. Kulik, R. Zhang. Privacy preserving group nearest neighbor queries. In *EDBT*, pp. 489-500, 2010.

[8] N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger. R^*-tree: an efficient and robust access methods for points and rectangles. In *SIGMOD*, 19(2), 322–331, 1990.

[9] H. Kriegel, P. Kroger, P. Kunath, and M. Renz. Generalizing the optimality of multi-step k-nearest neighbor query processing. In *Advances in Spatial and Temporal Databases*, pp. 75–92, 2007.

[10] H. G. Elmongui, M. F. Mokbel, and W. G. Aref. Continuous aggregate nearest neighbor queries. In *GeoInformatica*, 17(1): 63–95, 2013.

[11] Y. Li, F. Li, K. Yi, B. Yao, and M. Wang. Flexible aggregate similarity search. In *the 2011 ACM SIGMOD International Conference on Management of data*, pp. 1009–1020, 2011.

[12] J. Li, B. Wang, G. Wang, and X. BiM. Efficient Processing of Probabilistic Group Nearest Neighbor Query on Uncertain Data. In *Database Systems for Advanced Applications*, pp. 436–450, 2014.