

17th International Conference in Knowledge Based and Intelligent Information and
Engineering Systems - KES2013

Efficient Maximum Range Search on Remote Spatial Databases Using k-Nearest Neighbor Queries

Hideki Sato^{a,*}, Ryoichi Narita^b

^a*School of Informatics, Daido University, 10-3 Takiharu-cho, Minami-ku, Nagoya, 457-8530, Japan*

^b*Aichi Toho University, 3-11 Heiwagaoka, Meito-ku, Nagoya, 465-8515, Japan*

Abstract

Supporting aggregate range queries on remote spatial databases suffers from 1) huge and/or large numbers of databases, and 2) limited type of access interfaces. This paper applies the Regular Polygon based Search Algorithm (*RPSA*) to effectively addressing these problems. This algorithm requests a series of *k*-NN queries to obtain approximate aggregate range query results. The query point of a subsequent *k*-NN query is chosen from among the vertices of a regular polygon inscribed in a previously searched circle. Experimental results for *maximum* range query searches show that *Precision* is over 0.87 for a uniformly distributed dataset, over 0.92 for a skew-distributed dataset, and over 0.90 for a real dataset. Also, *Number of Requests (NOR)* ranges between 3.2 and 4.3, between 3.9 and 4.9, and between 3.0 and 4.2, respectively.

© 2013 The Authors. Published by Elsevier B.V. Open access under [CC BY-NC-ND license](#).

Selection and peer-review under responsibility of KES International

Keywords: Aggregate range query; Maximum range query; Regular polygon based search algorithm; Precision; Number of requests

1. Introduction

Recently, mobile computing has become a reality with the emergence of powerful mobile terminals, advances in wireless communication technologies, and the proliferation of location positioning equipment. In addition, the World Wide Web (WWW) is full of Web services disseminating their own information (i.e., digital maps, Points of Interests (POIs), etc.). Among them, Location Based Services (LBS) are major mobile computing applications that provide mobile users with location-dependent information and/or services. An exemplary LBS enables a single user at a specific location to obtain POI information in his/her neighborhood.

Another potentially promising LBS is one for supporting a group of mobile users. Consider, for example, a group of mobile users, each at a different location (query point), wishing to obtain information about a POI to enable them to meet there together. For such a group, Aggregate *k*-Nearest Neighbor (*k*-ANN) queries ([1],[2],[3]) and aggregate range queries are helpful. While the former finds POIs whose maximum (sum) of distances from each query point is top-*k* minimum, the latter finds POIs whose maximum (sum) of distances from each query point is within a certain distance.

*Corresponding author. Tel.: +81-52-612-6111 ; fax: +81-52-612-5623.

E-mail address: hsato@daido-it.ac.jp.

In this example, it can be assumed that the location data for each mobile user is obtainable from a location management server and that the POI information is accessible via other Web services. However, there are difficulties in answering these queries by mashing up existing Web services for two reasons. First, the LBS has to access its spatial databases for retrieving query results. If the databases are local and a query processing algorithm has direct access to spatial indices (i.e., R-trees [4] and its variants), it can retrieve them efficiently. It cannot, however, when queries are processed by accessing remote spatial databases that operate autonomously. Although some or all of the data from remote databases can be replicated in a local database and a separate index structure can be built, this is unfeasible when the database is huge or a large number of remote databases are accessed.

A further problem is access to spatial data on the WWW, which is limited to certain types of queries, due to simple and restrictive Web API interfaces. A typical scenario is one of searching for the POI nearest to the address given as a query point through a Web API interface. Unfortunately, Web API interfaces are not supported for processing either k -ANN queries or aggregate range queries on remote spatial databases. In other words, a new strategy for efficiently processing these queries is required.

This paper applies Regular Polygon based Search Algorithm (*RPSA*) to efficiently searching approximate aggregate range query results. Assuming k -Nearest Neighbor (k -NN) queries [5],[6] are Web API interfaces available for processing aggregate range queries, *RPSA* requests a series of k -NN queries to obtain aggregate range query results. The query point of a subsequent k -NN query is chosen from among the vertices of a regular polygon inscribed in a previously searched circle. We experimentally evaluated the algorithm in terms of *Precision* and *Number of Requests (NOR)* for *max* range queries, by using both synthetic and real datasets.

The rest of this paper is organized as follows. Section 2 mentions related work. Section 3 describes *max* range queries and the difficulties in processing them for the later discussion. In Section 4 we present our *RPSA* for aggregate range queries and in Section 5 experimentally evaluate it by applying to *max* range query searches, using both synthetic and real datasets. Section 6 concludes the paper with a summary of key points and a mention of future work.

2. Related Work

The existing literature in the field of location-dependent queries is extensively surveyed in the article [7]. The many types of location-dependent queries include NN queries and range queries. While NN queries [5], [6] retrieve the objects of a certain class that are the closest to a certain object or location, range queries [8], [9], [10] retrieve the objects within a certain range/region. Range queries can be static or moving/mobile, depending on whether the interesting region is fixed or moves. Range queries are also called *window queries* when the range is a rectangular window [11]. Similarly, *within-distance queries* [12] can be considered a variant of range queries where the range is a circle.

Since Group NN queries find ANN objects, the studies [13],[14] are closely related to ours. Papadias et al. [13] focused on *Euclidean* distance and the *sum* function. The work done by Yiu et al. [14] was more generalized and dealt with network distance. However, their work differs from ours in two respects. First, it deals with local spatial databases, while ours deals with remote ones. Second, it deals with ANN queries, while ours deals with aggregate range queries.

The studies [15],[16] are also closely related to ours, because they deal with providing users with location-dependent query results by using Web API interfaces to remote databases. The former [15] proposes a k -NN query processing algorithm that uses one or more range queries to search for the nearest neighbors of a given query point. The latter [16] proposes two range query processing algorithms that use k -NN searches. However, our algorithm differs from theirs in that it deals with aggregate range queries by using k -NN searches, while theirs don't deal with queries of this type.

3. Preliminaries

Aggregate range queries are an extension of range queries [8], [9], [10]. Let p be a point and Q be a set of query points. Then, aggregate distance function $d_{agg}(p, Q)$ is defined to be $agg(\{d(p, q) | q \in Q\})$, where $agg(\)$ is an aggregate function (e.g., *sum*, *max*, *min*) and $d(\)$ is a distance function. Given set P of data objects, set Q of

query points, and range distance l , an aggregate range query finds all the data objects p in P , such that $d_{agg}(p, Q)$ is within l , that is to say $\{p|p \in P, d_{agg}(p, Q) \leq l\}$.

Consider the example of Fig.1, where $P = \{p_1, p_2, p_3, p_4\}$ is a set of data objects and $Q = \{q_1, q_2\}$ is a set of query points (e.g., locations of mobile users). The number on each edge connecting a data object and a query point represents any distance cost between them. Table 1 presents $d_{agg}(p, Q)$ for each p in P . Here, the set $\{p_1, p_3, p_4\}$ is *sum* range query results within 820, while the set $\{p_1, p_3\}$ is the *maximum* range query results within 500.

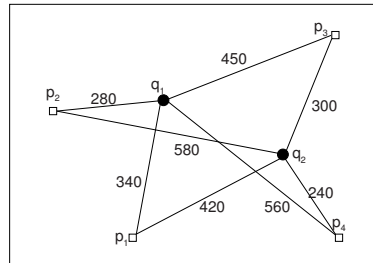


Fig. 1. Example query points (solid circles) and data points (hollow squares).

Table 1. Aggregate distance function applied to query points and data points shown in Fig.1.

aggregate function	<i>sum</i>	<i>maximum</i>	<i>minimum</i>
$d_{agg}(p_1, Q)$	760	420	340
$d_{agg}(p_2, Q)$	860	580	280
$d_{agg}(p_3, Q)$	750	450	300
$d_{agg}(p_4, Q)$	800	560	240

In the rest of the paper, we confine the distance function to a *Euclidean* one and the aggregate function to *maximum*. Let p be a point (x, y) and Q be a set of query points. The *maximum* distance function over Q of p , $d_{max, Q}(x, y)$ is defined in Eq.1. Figure 2(a) plots a contour graph of the *maximum* distance function $d_{max, Q}(x, y)$ over a set of 10 query points whose locations are randomly generated. Figure 2(b) presents contour lines that are projected on the x - y plane of the contour graph shown in Fig.2(a). Since $d_{max, Q}(x, y)$ is a convex function, there certainly exists a single point at which the function value is the lowest.

$$d_{max, Q}(x, y) = \max(\{\sqrt{(x - x_i)^2 + (y - y_i)^2} | (x_i, y_i) \in Q\}) \quad (1)$$

A contour graph of $d_{max, Q}(x, y)$ slopes a little complicatedly (See Fig.2). For a point (x, y) , $d_{max, Q}(x, y)$ computes its distance from query point q_i , if it resides in the furthest point Voronoi region $V(q_i)$ with regard to q_i [17]. For $(x_i, y_i) \in V(q_i)$ and $(x_j, y_j) \in V(q_j)$ ($i \neq j$), the distinct query points q_i and q_j are used to compute $d_{max, Q}(x_i, y_i)$ and $d_{max, Q}(x_j, y_j)$ respectively, even if $V(q_i)$ is adjacent to $V(q_j)$. This is the reason a contour graph of $d_{max, Q}(x, y)$ slopes a little complicatedly.

4. Search Algorithm For Aggregate Range Query

In this section, *RPSA* for aggregate range queries is described.

4.1. Consideration

k -NN queries are used to process *max* range queries. The circle of Fig.3 is the searched region of a k -NN query, where k is 5. In the circle, q is a query point and the set $\{p_1, p_2, p_3, p_4, p_5\}$ is the query results. The radius of the circle equals the distance r between the 5th nearest neighbor p_5 and q . A *max* range query imposes upon its result the condition that each *max* distance over a set of query points is within the range distance *limit*. A

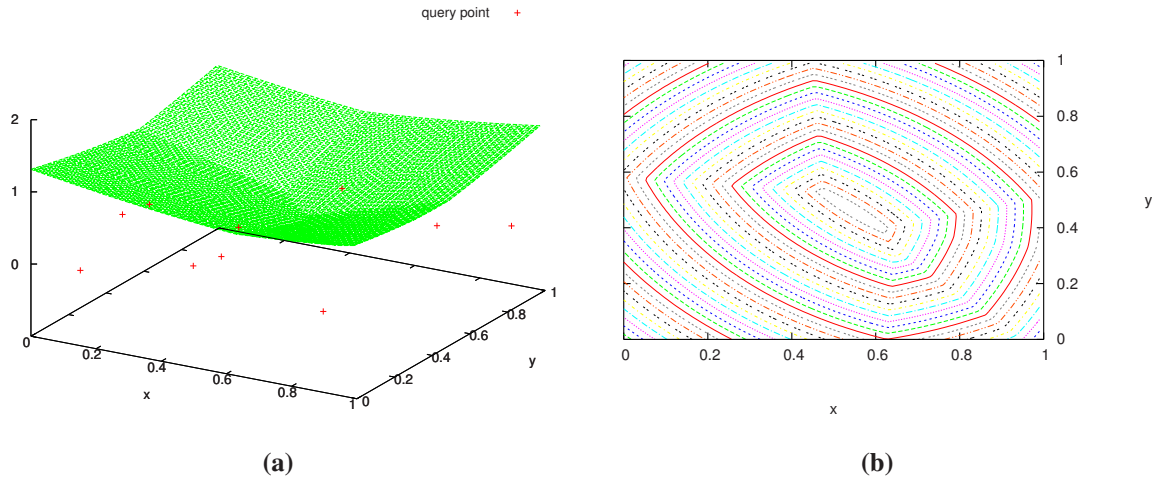


Fig. 2. Maximum distance function $d_{\max,Q}(x,y)$ (Euclidean distance, number of query points=10).

max range query result contains all the elements of $\{p_1, p_2, p_3, p_4, p_5\}$ (See Fig.3), if each satisfies this condition. Additionally, other spatial data to be answered may reside outside the circle. However, the region where such data reside cannot be analytically computed. Instead, a heuristic method is chosen for searching the regions.

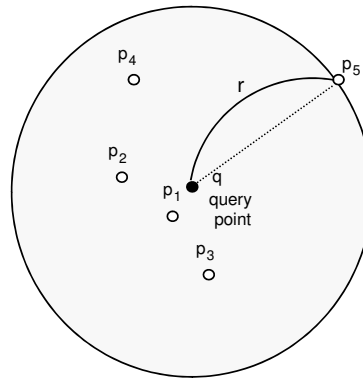


Fig. 3. Searched circle of 5-NN query (query point (solid circles) and data objects (hollow circles)).

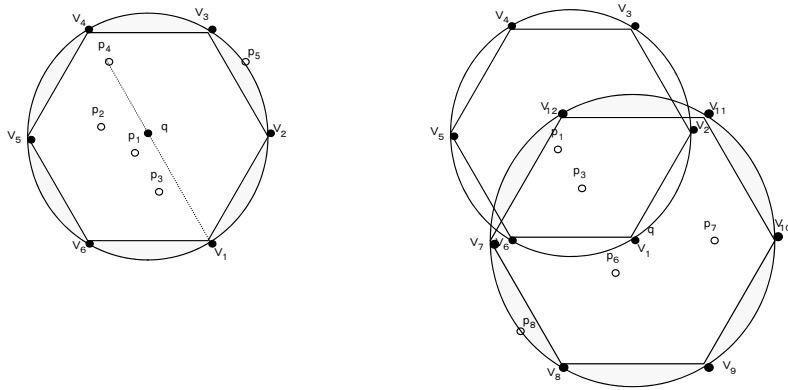
Let q be a query point of a k -NN query, p be a point outside a searched circle, and v be the point at which line segment \overline{pq} and the circumference of the circle cross. Since $d_{\max}(\cdot)$ is a convex function, Eq.2 holds for q, p, v and set Q of query points of a *max* range query, where $0 \leq \alpha \leq 1, \beta = 1 - \alpha, v = \alpha q + \beta p$. In case that $d_{\max}(q, Q) \leq d_{\max}(v, Q)$ holds, it is derived from Eq.2 where $d_{\max}(v, Q) \leq d_{\max}(p, Q)$ holds. Consequently, $d_{\max}(p, Q) \leq \text{limit}$ may hold under $d_{\max}(v, Q) \leq \text{limit}$. In the opposite case where $d_{\max}(q, Q) > d_{\max}(v, Q)$ holds, $d_{\max}(p, Q) \leq \text{limit}$ may hold. Eq.3 is a logical formula regarding whether p may exist such that $d_{\max}(p, Q) \leq \text{limit}$ holds, which is derived by merging both cases.

$$d_{\max}(v, Q) \leq \alpha d_{\max}(q, Q) + \beta d_{\max}(p, Q) \quad (2)$$

$$(d_{\max}(q, Q) \leq d_{\max}(v, Q) \wedge d_{\max}(v, Q) \leq \text{limit}) \vee (d_{\max}(q, Q) > d_{\max}(v, Q)) \quad (3)$$

Since an infinite number of points exist continuously on the circumference of a circle, which point on the circumference should be chosen as v is problematic in searching the region for spatial data to be answered. A regular polygon inscribed in a circle can be used to provide its vertices as candidates. Figure 4(a) shows an inscribed 6-regular polygon. Assume that $d_{\max}(p_4, Q)$ equals $\max(\{d_{\max}(p_i, Q) | 1 \leq i \leq 5\})$. The first vertex v_1 of

the polygon is set to be a point at which the line extending the line segment $\overline{p_4q}$ ahead of q and the circumference of the circle cross¹. Each element of $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ can be chosen as a point for searching regions for spatial data to be answered, if it satisfies Eq.3. A list of these vertices is called *CQPlist* (*Candidate Query Point list*). Any k -NN query result whose query point is on *CQPlist* can be added to the *max* range query result, if it satisfies the condition that its *max* distance over Q is within the *limit*.



(a) Inscribed 6-regular polygon (b) Subsequent 5-NN query search

Fig. 4. Additional k -NN query search whose query point is a vertex of n -regular polygon.

Figure 4(b) shows a newly searched circle of a 5-NN query with v_1 as a query point. The set $\{p_1, p_3, p_6, p_7, p_8\}$ is the query results. While p_1 and p_3 are re-searched points, p_6, p_7 , and p_8 are newly searched points. Any of the latter three may be added to the *max* range query result, if it satisfies the condition that its *max* distance over Q is within *limit*. The new 6-regular polygon inscribed in the new circle provides its vertices. Each element of $\{v_7, v_8, v_9, v_{10}, v_{11}\}$ can be added to *CQPlist*, if it satisfies Eq.3. However, v_{12} is not added to the list, because it resides inside the previously searched region. For the same reason, if either v_2 or v_6 is on the list, it is removed from the list because it has been used as a query point.

4.2. Regular polygon based search algorithm

Figure 5 shows *RPSA*. A k -NN query with fqp as a query point is requested (line 1). Each element of the query result is examined regarding whether it satisfies the condition that its *max* distance over qp is within the *limit* (line 2). *Clist* maintains previously searched circles and is initialized (line 3). A circle with fqp as the center is created (line 4) and the vertices of the regular polygon inscribed in the circle are gathered (line 5). *CQPlist* is initially created (line 6), in which candidate query points are listed in order of ascending *max* distance. The same process is repeated (line 7-15), until *CQPlist* becomes empty. The candidate query point with the least *max* distance is selected as the query point (line 8) for a k -NN query (line 9). Each element of the query result is added to *Rlist*², if it satisfies the condition that its *sum* distance over qp is within the *limit* (line 10). A searched circle is created (line 12) and the vertices of the regular polygon inscribed in the circle are gathered (line 13).

CQPlist is related to the condition terminating the loop execution (line 7-15), which is initially created with at most n candidate points (line 6). A single execution of the loop necessarily consumes a single query point that is removed from *CQPlist*. In line 14, *CQPlist* is updated to be a list of elements in either *CQPlist* or a set of vertices of the regular polygon inscribed in a searched circle (line 12) and satisfying two conditions. One is that its *max* distance is the *limit* or less and the other is that it does not reside inside previously searched circles. The regions of previously searched circles increase monotonically. Accordingly, the loop execution necessarily terminates.

¹This is heuristically determined because p may reside on the line extending the line segment $\overline{qv_1}$ ahead of v_1 such that $d_{\max}(p, Q) \leq \text{limit}$.

²*Rlist* is a list containing identifiers of data objects retrieved and is finally returned as the aggregate range query results (See Fig.5).

<div style="display: flex; justify-content: space-between;"> <div style="width: 80%;"> <p>RPSA (qp, limit, k, fqp, n)</p> <p>Input:</p> <ul style="list-style-type: none"> a set of query points limit of aggregate distance over a set of query points number of data to be returned for k-NN query query point for first k-NN query number of edges of a regular polygon <p>Output:</p> <ul style="list-style-type: none"> aggregate range query result <pre> 01 Slist:=NEAREST_NEIGHBOR_SEARCH(k, fqp); 02 Rlist:=MAKE_RESULT_LIST(qp, limit, [], Slist); 03 Clist:=[]; 04 circle:=MAKE_CIRCLE(fqp, DISTANCE(fqp, Slist)); 05 Vlist:=MAKE_VERTEX_LIST(circle, n, MAX_AGGREGATE_DISTANCE_POINT(Slist, qp)); 06 CQPlist:=MAKE_CANDIDATE_QUERY_POINT_LIST([], Vlist, qp, limit, Clist); 07 while(not(CQPlist=[])){ 08 let search_point be the head element of CQPlist and CQPlist be the remaining list of CQPlist; 09 Slist:=NEAREST_NEIGHBOR_SEARCH(k, search_point); 10 Rlist:=MAKE_AGGREGATE_DISTANCE_LIST(qp, limit, Rlist, Slist); 11 Clist:=APPEND(Clist, circle); 12 circle:=MAKE_CIRCLE(search_point, DISTANCE(search_point, Slist)); 13 Vlist:=MAKE_VERTEX_LIST(circle, n, MAX_AGGREGATE_DISTANCE_POINT(Slist, qp)); 14 CQPlist:=MAKE_CANDIDATE_QUERY_POINT_LIST(CQPlist, Vlist, qp, limit, Clist); 15 } 16 return Rlist; </pre> </div> <div style="width: 15%; text-align: right; vertical-align: top;"> <p>qp</p> <p>limit</p> <p>k</p> <p>fqp</p> <p>n</p> <p>Rlist</p> </div> </div>	
--	--

Fig. 5. Regular Polygon based Search Algorithm RPSA.

5. Experimental Evaluation

We experimentally evaluated the performance of RPSA by measuring *Precision* and *NOR* over *max k*-th range queries. Given set P of data objects, set Q of query points, and $k (\leq |Q|)$, the range distance of a *max k*-th range query is set to be $\frac{d_{\max}(p, Q) + d_{\max}(o, Q)}{2}$, where p is the top- k minimum data object and o is the top- $(k+1)$ minimum data object³. *Precision* is used as the criteria to specify the accuracy of *max k*-th range query results. It is defined in Eq.4, where $R_{\max k\text{-th range}}$ is the original *max k*-th range query result and $R_{RPSA(\max k\text{-th range})}$ is the query result retrieved with RPSA. *NOR* is the requested number of k' -NN queries. Since the processing time for answering a *sum k*-th range query with RPSA is approximately proportional to *NOR* (See Fig.5), *NOR* can be used as another criteria for measuring performance. As the k of a *max k*-th range query increases, the region where the query result resides becomes larger. Therefore, k' is set to be equal to k to enable comparison among queries of different k . Each experimental result is the average of 100 trials conducted for each setting. All the query point locations are uniformly distributed. The minimal point⁴ is used as a query point of the first k -NN search to retrieve a *max k*-th range query.

$$Precision(\max k - th \text{ range}) = \frac{|R_{\max k\text{-th range}} \cap R_{RPSA(\max k\text{-th range})}|}{|R_{\max k\text{-th range}}|} \quad (4)$$

³ o is formally defined as follows. Let O be $\{o | o \in P, d_{\max}(p, Q) \leq d_{\max}(o, Q)\}$. In case that O is empty, o is set to be p . Otherwise, $o \in O$ is the data object such that $d_{\max}(o, Q) \leq d_{\max}(o', Q)$ holds for all $o' \in O$.

⁴ Given set Q of query points, the minimal point corresponds exactly to the center of the Minimum Covering Circle (MCC) over Q . Accordingly, it can be obtained with a computational geometric algorithm [17].

5.1. Effects of regular polygon on performance

We measured how the number of edges of a regular polygon, one of the *RPSA* parameters (See Fig.5), affects *RPSA* performance. Experiments regarding *max* 10th range queries were conducted by varying the value from 3 to 30, with 10,000 data points whose locations are uniformly distributed. *Precision* is over 0.9 when the value is 26 or more (See Fig.6(a)). *NOR* does not increase in proportion to the value (See Fig.6(b)). These results suggest that a regular polygon with 26 or more edges is sufficient to search for query results. We consider the reason for this is as follows. A regular polygon inscribed in a searched circle is used to provide its vertices as query point candidates for searching regions where spatial data objects to be answered may reside. In Section 3, it is mentioned that a contour graph of $d_{max,Q}(x, y)$ slopes a little complicatedly (See Fig.2). To find proper query points it is necessary to minutely follow the contour surface of the complicated slopes of $d_{max,Q}(x, y)$. This leads to compact space intervals between the query point candidates.

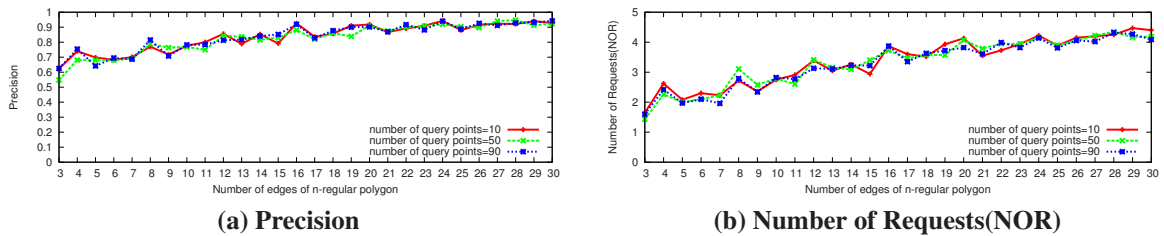


Fig. 6. Performance of *max* 10th range search for varying number of edges of a regular polygon.

5.2. RPSA performance for several types of datasets

We measured *RPSA* performance for several types of datasets. First, we conducted experiments by varying the number of query points and the k of *max* k -th range queries, using 26 regular polygons and 10,000 data points whose locations were uniformly distributed. In these experiments, we found that *Precision* was over 0.87 (See Fig.7(a)) and that *NOR* ranged between 3.2 and 4.3 (See Fig.7(b)).

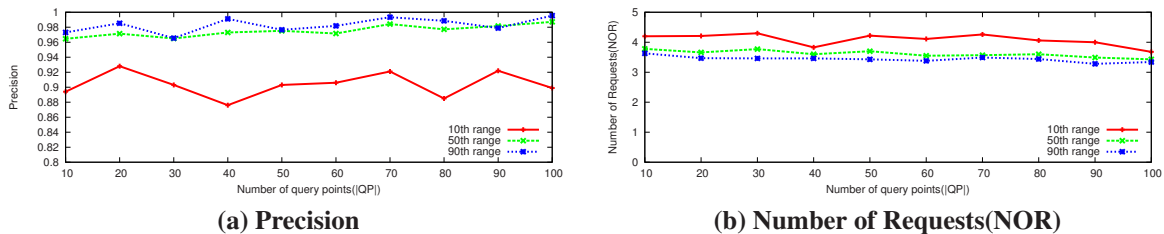
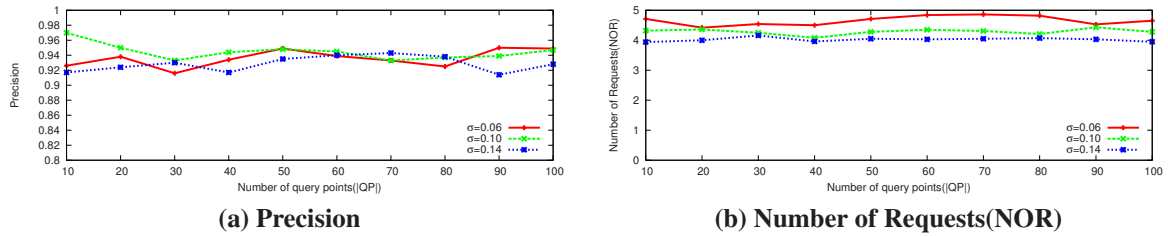


Fig. 7. Performance of *max* k -th range search over data points of uniform distribution.

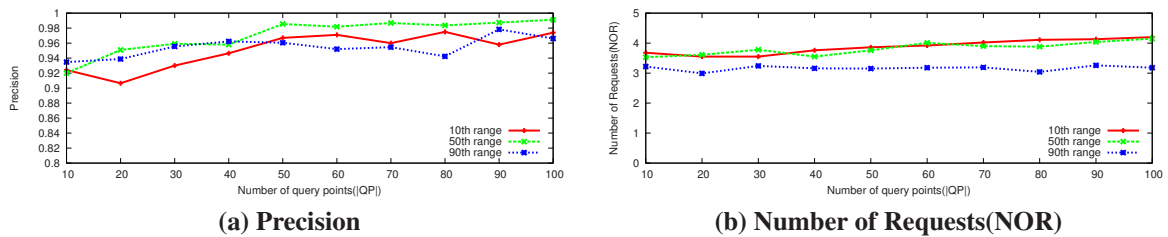
Second, we measured performance by using skew-distributed data points. We conducted experiments regarding *max* 10th range queries by varying the number of query points, using 26 regular polygons and 10,000 data points whose locations were generated from two-dimensional Gaussian distribution. We let the location of a data point be (x, y) ($x \in [0, 1)$, $y \in [0, 1)$). The mean point of Gaussian distribution was randomly generated and the standard deviation (σ) was changed. Measured results showed *Precision* was over 0.92 (See Fig.8(a)) and *NOR* ranged between 3.9 and 4.9 (See Fig.8(b)). The larger σ is, the more performance is like that for uniform distribution. This is because the Gaussian distribution of large σ is similar to uniform distribution.

Third, we measured performance by using real data points. The points concerned restaurants located in Nagoya and were available at the Web site and accessible via the Web API⁵. There were 2003 corresponding restaurants,

⁵<http://webservice.recruit.co.jp/hotpepper/gourmet/v1/>

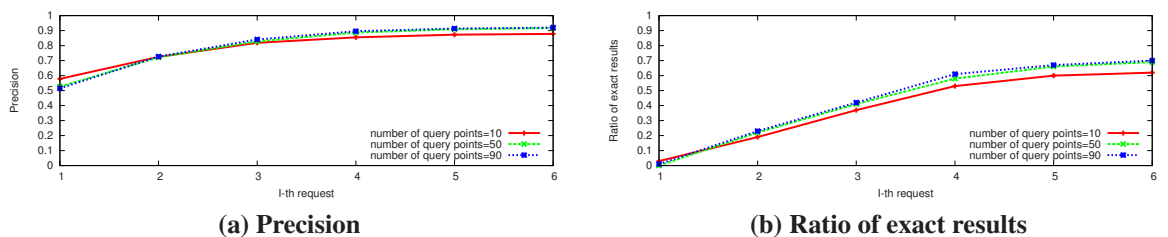
Fig. 8. Performance of *max* 10th range search over data points of Gaussian distribution.

most of them in downtown Nagoya. We conducted experiments by varying the number of query points and the *k* of *max k*-th range queries, using 26 regular polygons and real data points. Measurement results showed *Precision* was over 0.9 (See Fig.9(a)) and *NOR* ranged between 3.0 and 4.2 (See Fig.9(b)).

Fig. 9. Performance of *max k*-th range search over real data points.

5.3. Precision improvement process

In this subsection, we experimentally clarify a *Precision* improvement process. Let $\langle p_1, p_2, \dots, p_n \rangle$ be a sequence of *Precision* values regarding *max k*-th range query results, where $p_i (1 \leq i \leq n)$ is *Precision* after requesting the *i*-th *k*-NN query. Note that $p_i (f \leq i \leq n)$ is set to p_f when *NOR* *f* is less than *n*. We conducted experiments to compute a sequence of average *Precision* values for *max* 10th range queries by varying the number of query points, using 26 regular polygons and 10,000 data points whose locations are uniformly distributed. Figure 10(a) shows that *Precision* is over 0.87 after requesting the 5th 10-NN queries. We also measured the ratio of exact results after requesting the *i*-th 10-NN query. However, exact results do not necessarily lead to immediate termination of *RPSA*, because the algorithm cannot completely determine whether exact results are obtained. It therefore continues to execute until all available query points are exhausted. Conversely, it terminates its execution when no available query points remain, even if it does not obtain exact results. Figure 10(b) shows that the ratio of exact results was found to be over 0.6 after requesting the 5th 10-NN queries.

Fig. 10. Precision improvement process of *max* 10th range query search.

5.4. Effects of first query point on performance

We measure how the query point of the first *k*-NN search, one of the *RPSA* parameters, affects *RPSA* performance. We conducted experiments by varying the number of query points, using 26 regular polygons and 10,000

data points whose locations are uniformly distributed. Figure 11 shows *Precision* and *NOR* of *max* 10th range query results searched for by *RPSA* using each distinct query points. The minimal point is definitely superior to the middle point⁶ in *Precision* (See Fig.11(a)), although it is inferior to the middle point in *NOR* (See Fig.11(b)). We consider that this is because *RPSA* tends to not find candidate query points for conducting subsequent *k*-NN searches to achieve higher *Precision*, if the middle point is used as the query point of the first *k*-NN search. This probably leads to a smaller *NOR* than that of the minimal point for the first *k*-NN query.

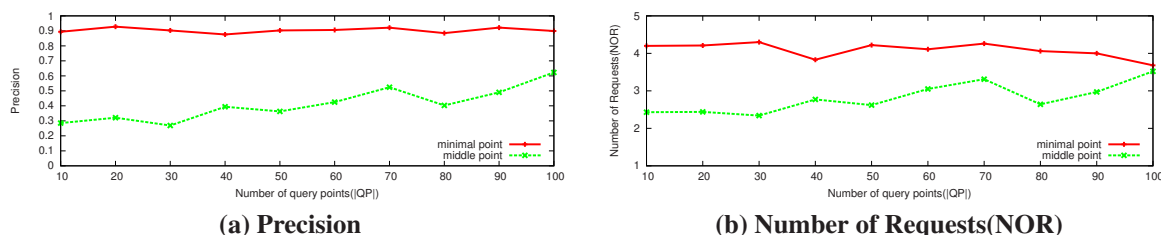


Fig. 11. Performance of *max* 10th range search with each distinct first query points.

6. Conclusion

This paper described the Regular Polygon based Search Algorithm (*RPSA*) as a means of efficiently searching for approximate aggregate range query results. The algorithm requests a series of *k*-NN queries to obtain aggregate range query results. The query point of a subsequent *k*-NN query is chosen among the vertices of a regular polygon inscribed in a previously searched circle. Experimental *max* range query search results were as follows. 1) A regular polygon with 26 or more edges is sufficient to search for query results. 2) *Precision* is over 0.87 for a uniformly distributed dataset, over 0.92 for a skew-distributed dataset, and over 0.9 for a real dataset. The respective *Number of Requests (NOR)* ranges between 3.2 and 4.3, between 3.9 and 4.9, and between 3.0 and 4.2. 3) *Precision* is over 0.87 after requesting the 5th *k*-NN queries and over 60% of the queries are exact results. 4) As a query point of the first *k*-NN query, the minimal point is superior to the middle point in *Precision*. Our future work will include improving *RPSA* performance for *max* range query searches.

References

- [1] Sato, H., 2010. "Approximately solving aggregate k-nearest neighbor queries over web services", In: Proceedings of 2nd International Symposium on Intelligent Decision Technologies. Baltimore, USA, p. 445–454
- [2] Sato, H. Approximately searching aggregate k-nearest neighbors on remote spatial databases using representative query points. In: In: Watanabe, T., Jain LC, editors. *Innovations in Intelligent Machines-2: Intelligent Paradigms and Applications*, New York: Springer-Verlag; 2011, p. 91-102
- [3] Sato, H., Narita, R., 2012. "Multistep search algorithm for sum k-nearest neighbor queries on remote spatial databases", In: Proceedings of 5th International Conference on Interactive Multimedia Systems and Services. Gifu, Japan, p. 385–397
- [4] Guttman, A., 1984. "R-trees: a dynamic index structure for spatial searching", In: Proceedings of ACM SIGMOD International Conference on Management of Data., p. 47–57
- [5] Roussopoulos, N., Kelly, S., Vincent, F., 1995. "Nearest neighbor queries", In: Proceedings of ACM SIGMOD International Conference on Management of Data., p. 71–79
- [6] Hjaltason GR, Samet, H. Distance browsing in spatial databases. *ACM Trans. Database Systems* 1999;**24**(2): p. 265-318
- [7] Ilarri, S., Menna, E., Illarramendi, A. Location-dependent query processing: where we are and where we are heading. *ACM Computing Survey* 2010;**42**(3), Article 12
- [8] Xu, B., Wolfson, O., 2003. "Time-series prediction with applications to traffic and moving objects databases", In: Proceedings of 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access., p. 56–60
- [9] Trajcevski, G., Wolfson, O., Hinrichs, K., Chamberlain, S. Managing uncertainty in moving objects databases, *ACM Trans. Database Systems* 2004;**29**(3): p. 463-507
- [10] Yu PS, Chen SK, Wu KL, Incremental processing of continual range queries over moving objects, *IEEE Trans. on Knowledge and Data Engineering* 2006;**18**(11): p. 1560-1575

⁶Given set *Q* of query points, the middle point is the center of the Minimum Bounding Rectangle (MBR) over *Q*.

- [11] Tao, Y., Sun, J., Papadias, D., 2003. "Selectivity estimation for predictive spatio-temporal queries", In: Proceedings of 19th International Conference on Data Engineering., p. 417–428
- [12] Trajcevski, G., Scheuermann, P., 2003. "Triggers and continuous queries in moving objects database", In: Proceedings of 6th International DEXA Workshop on Mobility in Databases and Distributed Systems., p. 905–910
- [13] Papadias, D., Shen, Q., Tao, Y., Mouratidis, K., 2004. "Group nearest neighbor queries", In: Proceedings of International Conference on Data Engineering., p. 301–312
- [14] Yiu ML, Mamoulis, M., Papadias, D. Aggregate nearest neighbor queries in road networks. *IEEE Trans. on Knowledge and Data Engineering* 2005;**17**(6): p. 820-833
- [15] Liu, D., Lim, E., Ng, W., 2002. "Efficient k-nearest neighbor queries on remote spatial databases using range estimation", In: Proceedings of SSDBM., p. 121–130
- [16] Bae WD, Alkobaisi, S., Kim SH, 2007. "Supporting range queries on web data using k-nearest neighbor search", In: Proceedings of W2GIS., p. 61–75
- [17] Berg MD, Kreveld MV, Overmars, M., Schwarzkopf, O., *Computational geometry: algorithms and applications*. Berlin Heidelberg: Springer-Verlag; 1997.