IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 23, 2020, accepted March 6, 2020, date of publication March 10, 2020, date of current version March 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2979739

A Voronoi-Based Group Reverse k Farthest Neighbor Query Method in the Obstacle Space

YONGSHAN LIU[1], XIANG GONG[1], DEHAN KONG[2], TIANBAO HAO[1], AND XIAOQI YAN[1]
[1]Department of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China
[2]Department of Information Engineering, Hebei University of Environmental Engineering, Qinhuangdao 066102, China

Corresponding author: Xiang Gong (alan0432@qq.com)

This work was supported in part by the National Natural Science Foundation under Grant 61972334, in part by the Natural Science Foundation of Hebei Province under Grant F2017203019, and in part by the 2019 Hebei University Higher Education Science and Technology Research Youth Fund Project under Grant QN2019044.

ABSTRACT With the rapid development of intelligent transportation and geographic information system, spatial data query technology has attracted the attention of many scholars. Among them, the reverse farthest neighbor query from the data point to find the target point as its farthest adjacent data points, used to obtain target set of weak influence. Its research results have been widely applied to facility location, earthquake relief, marketing and other major areas. Thus, the research of reverse furthest neighbor query technology is of great significance. However, the existing methods only deal with a single query point, and do not consider how to obtain the optimal farthest neighbor position of group reverse when the number of query points changes from one to a group. In addition, considering it is difficult to avoid some geographical location restrictions in the real situation, the existing studies are limited to road network or Euclidean space simplely, without taking the existence of obstacles into consideration. To this end, this paper proposed the V-OGRkFN(Voronoi-Obstacle Group Reverse k Farthest Neighbor) algorithm. Firstly, the algorithm gets the minimum cover circle of query points which are considered into a whole. Secondly, we use the framework based on the filtering and refining process of query by pruning strategy based on Voronoi diagram's properties. Then we get the candidate set using the theorem of transformation between k nearest neighbors and k farthest neighbors. Finally, the refining algorithm is given to get the final results. V-OGRkFN algorithm shows great performance of reverse k farthest neighbor query through the specific comparative experimental analysis.

INDEX TERMS Obstacle space, reverse farthest neighbor query, Voronoi diagram, minimum coverage circle.

I. INTRODUCTION

The popularity of smart devices and the rapid development of geographic information system, makes the spatial data query technology plays a more and more important role in real life. In recent years, the concept of reverse farthest neighbor query has attracted the extensive attention of many scholars. It is used to search the target point as the farthest neighbor data point from the data point set to obtain the weak influence set of the target point. It has important application value in the fields of facility location, earthquake relief, resource allocation, marketing and so on. For example, in the location selection of chemical treatment plants, cause toxic substances are likely to be formed in the process of treatment, endangering the normal life of nearby residents, it is necessary to arrange the location of the plants as far as possible in a safe range away from the residential area. And as for the customers, when they are shopping,they would like to the shops which are closer to them, but the mall managers who are far from these customers are also hope that more and more people go to their store shopping. Therefore, it is necessary to take effective measures for customers who are far away from the mall, such as registering monitoring query on the server to obtain its reverse farthest neighbor object in real time, and selecting effective marketing means, such as increasing propaganda and increasing shuttle bus to attract consumers. In addition, the reverse farthest neighbor query can also be used for facility site selection to effectively avoid competition

The associate editor coordinating the review of this manuscript and approving it for publication was Anna Visvizi.

among peers and to find suitable settlements to settle the victims in earthquake relief, in order to avoid potential epidemic threats caused by the mutual influence between settlements effectively. Therefore, it is of great practical significance to solve the related problems of reverse farthest neighbor query reasonably and efficiently.

However, the existing reverse farthest neighbor query technology is only limited to the road network space and the Euclidean space, and does not consider the existence of obstacles, in the real situation, it is difficult to avoid some geographical restrictions, such as buildings, rivers and mountains. In addition, existing studies are only limited to the reverse farthest neighbor query of a single point, and fail to consider how to obtain the optimal farthest neighbor position of the reverse group when the number of query points changes from one to a group. Considering that this kind of problem has many applications in real situations, such as how to effectively determine the location of a special facility (such as chemical plant, waste disposal plant, etc.) in a group of residential communities adjacent to the location under the constraint of obstacles. Therefore, this paper studies the group reverse k farthest neighbor query based on the reverse farthest neighbor query in obstacle space.

In order to solve this problem effectively, this paper first proposed the definition of group reverse k farthest neighbor query based on obstacle space, we proposed V-OGRkFN algorithm based on Voronoi diagram. This algorithm has good query performance and has certain advantages to solve the reverse k farthest neighbor query problem of multiple points in the obstacle space. The main contributions of this paper are as follows:

1) V-OGRkFN algorithm is proposed to reach the fast speed of group reverse k fartheset neighbor query. The theorems and definitions are identified before the V-OGRkFN algorithm.

2) Some related algorithms's studies are conducted to help making best use of the V-OGRkFN algorithm, including minimun coverage cicle algorithm, pruning algorithm, transofrmation algorithm and refining algorithm. These work are all made contribution to understand the main meaning of the V-OGRkFN algorithm better.

3) An analysis system is using made by us to verify the algorithm's speed and stability. At the same time, a comparative test is conducted to hold our main idea of this paper.

The remainder of the paper is organized as follows. Section 2 overviews the state of group reverse k farthest neighbor query and some related work. In section 3, the concept of group reverse k farthest neighbor query based on obstacle space is proposed and the related theorems are introduced briefly. Next section 4 describes the process of the group reverse k farthest neighbor query in the obstacle space in detail and the analysis of algorithm in the pruning and refining process. In the last section 5, experimental comparison is made with simulation data to prove the effectiveness and robustness of the proposed query algorithm.

II. RELATED WORK

A. THE REARCH OF NEAREST AND FARTHEST NEIGHBOR QUERY

In 2000, Korn and Muthukrishnan [1] first proposed the concept of reverse farthest neighbor query for obtaining weak influence set. In 2009, Li and Hao [2] proposed a query algorithm to determine RFN candidate sets by using the property of quartile neighborhood. In the same year, Yao *et al.* [3] proposed the PFC algorithm. Due to its high processing cost, Yao made improvements on it, and then proposed the CHFC algorithm, which was used to solve the problem of monochrome and two-color reverse farthest neighbor query in Euclidean space. At the same time, Tran *et al.* [4] presented a method to solve the reverse farthest neighbor query and reverse k farthest neighbor query in road network space by using Voronoi diagram related attributes and Dijkstra algorithm. In 2010, Liu *et al.* [5] proposed an optimization algorithm for reverse furthest neighbor query, which used the properties of convex hull and triangle inequality to trim, effectively shortening the query time. In 2011, Gao *et al.* [6] studied the aggregation k distant neighbor problem defined by the aggregation functions SUM, MAX and MIN, and presented the MB and BF algorithms based on R tree. In 2012, Liu *et al.* [7] proposed the reverse farthest neighbor query algorithm without location restrictions, and compared it with the PIV algorithm proposed in 2010. In 2013, Said *et al.* [8] proposed a collaborative filtering scheduling algorithm based on k distant neighborhood. In 2014, Wang *et al.* [9] discussed the problem of searching k furthest neighbors on the road network. By organizing objects into a compact cluster and calculating the network distance from the cluster to a group of reference points in advance, most of the clusters were pruned to achieve the operation of searching top k points in the data set of the aggregation network distance efficiently. In 2016, Li [10] conducted a precise pruning of the query space based on the algorithm of filter-purification framework, and proposed the dynamic reverse farthest neighbor query algorithm and probabilistic reverse farthest neighbor query algorithm. In the same year, Wang *et al.* [11] first studied the reverse farthest neighbor query problem based on any k value. In 2017, Xu *et al.* [12] proposed an algorithm for solving the monochrome and two-color reverse farthest neighbor query in the road network by using landmark and zoning technology. In the same year, Li *et al.* [13] proposed a probabilistic RFN query algorithm for uncertain moving objects in Euclidean space, which solved the weak influence set problem of uncertain moving objects. In the same year, Liu and Hao [14] proposed the farthest neighbor query problem based on MOIS- tree index structure, and gave the definition of minimum maximum distance and maximum distance, and then gave the farthest neighbor query algorithm. In 2018, Wang *et al.* [15] proposed two verification methods applicable to reverse k farthest neighbor query in outsourcing spatial database based on the existing reverse k farthest neighbor query method and mr-tree verification data structure.

The algorithms mentioned above are all for Euclidean space and road network space, and they are not completely applicable for group reverse farthest neighbor query in obstacle space. A new density-based clustering algorithm, RNN-DBSCAN, was presented which uses reverse nearest neighbor counts as an estimate of observation density by Pei *et al.* [34]. Clustering is performed using a DBSCAN-like approach based on k nearest neighbor graph traversals through dense observations. Chen *et al.* [35] developed an approach which computes kNN for only promising clients by utilising a two-level grid index (ADPGI) to find a region for setting up a new service site such that it can influence the most clients efficiently.

B. THE RESEARCH OF GROUP NEIGHBOR QUERY

In 2004, Papadias *et al.* [16] presented three algorithms for solving group nearest neighbor query, namely MQM, SPM and MBM. In 2005, Li *et al.* [17] proposed the GNN method of pruning with the ellipse formed by the farthest point in the set of query points and its MBR as the breakthrough point. In 2005, Papadias *et al.* [18] and Yiu *et al.* [19] proposed an aggregated nearest neighbor query algorithm based on a set of query points. In 2008, Lian and Chen [20] proposed a probabilistic nearest neighbor query algorithm for uncertain data. In 2010, Song *et al.* [21] proposed to obtain the reverse k nearest neighbor query algorithm in Euclidean space by calculating the minimum coverage circle of the query object and using the pruning method based on R tree. In the same year, Sun and Hao [22] discussed from the perspectives of collinear and incollinear of query point sets, and proposed a group nearest neighbor query algorithm based on Voronoi graph. In 2011, Chen *et al.* [23] proposed a method to obtain the optimal solution of the nearest group query of constraint group by iteratively updating clustering. In 2013, Yang and Hao [24] proposed an algorithm to solve the nearest neighbor query result set under the obstacle space based on the different location relations between the points in the data set and the minimum outsourcing distance of the query point set, pruning the obstacles without influence and obtaining the obstacle distance between the point sets. In 2017, Guo *et al.* [25] proposed the nearest neighbor query algorithm for line segment groups based on Voronoi graph in spatial database. In the same year, Li *et al.* [26] improved the nearest neighbor query algorithm for the existing uncertain data sets and proposed the GkNN query method based on Voronoi graph. Li *et al.* [27] proposed an algorithm that can efficiently process top k query of reverse spatial preference, determine the attribute level of the object according to the spatial relationship between objects, and use the user grouping strategy based on weight to obtain those users meeting the top k attribute through batch pruning of the data set. In 2018, Wang *et al.* [28] based on the multi-user preference query of user groups, took the preferences of members in the group into consideration of the grouping, generated the pre-selected queue according to the descending order of users' preference similarity to the query object, and obtained the final users in the same group through layers of filtering. In the same year, Guo *et al.* [29] proposed the collaborative space keyword top k query, and obtained the top k objects close to multiple query positions and highly correlated between text and multiple groups of query keywords through the pruning data set based on the keyword relevance calculation formula based on the weight of query keywords. Zhou *et al.* [30] proposed a reverse k rank query algorithm gp-rkr based on group, and obtained query results gradually through dynamic threshold adjustment and pruning strategy based on lg-index Index structure. The above research on group query is limited to the nearest neighbor query and its related variant query, and has not been involved in the reverse distant neighbor query. Guo *et al.* [36] studied a new problem: a GNN search on a road network that incorporates cohesive social relationships (CGNN). Santoso *et al.* [37] provided a solution in the form of processing a top-k dominating query using an indexing grid. Group data indexing task is performed by placing data in groups on a grid based on the aggregate value. Zhao and Xiong [38] studied a new spatial keyword query motivated by the scenario where a group of users staying at different places wishes to find a compact set of POIs (such as a restaurant and two museums) that is close to all users. Zhang *et al.* [39] proposed an algorithm that deals with the group visible nearest surrounder query based on the hybrid index structure for solving the problem of group visible nearest surrounder query in obstacle space.

To sum up, there is a lack of research on group reverse k farthest neighbor query in obstacle space both indoor and abroad. However, this kind of research has important theoretical significance and practical value. Therefore, this paper proposes the definition of group reverse k farthest neighbor query in obstacle space and conducts intensive research.

III. DEFINITIONS AND THEOREMS

A. PROBLEM DEFINITION

The problem studied in this section is based on the discussion in the obstacle space. So it is necessary to emphasize the concept of obstacle. The specific definition is described in definition 1.

Definition 1 (Obstacles): In real life, an obstacle is an object that hinders or hinders the progress of the subject from the beginning to the end. In this paper, the obstacle is abstracted as a straight line segment with varying lengths and directions, represented by the letter o.

Due to the existence of obstacles, the visibility between two points is not visible directly. The definition of visibility between point sets is given in definition 2.

Definition 2 (Visibility): Given two points p and point q and the set of obstacles O, if and only if the two points are connected to any line segment in the set of obstacles do not intersect, point p and point q are said to be visible.

In addition, due to the existence of obstacles, the path between two points in the spatial data is no longer as simple as the connection between two points. Therefore, in order

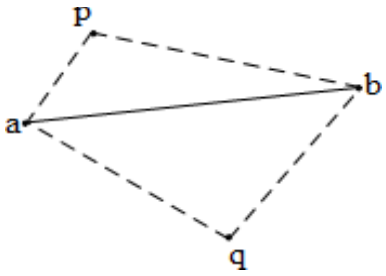


FIGURE 1. Schematic diagram of obstacle distance.

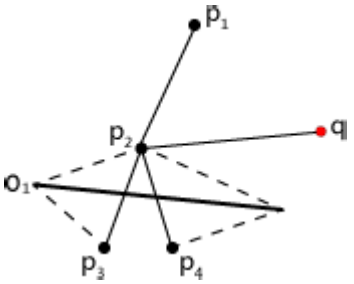


FIGURE 2. Schematic diagram of obstacle reverse *k* farthest neighbor query.

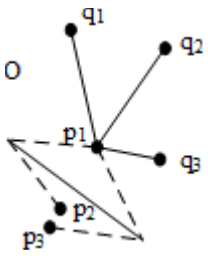


FIGURE 3. Reverse *k* farthest neighbor query of obstacle group.

to solve the problem of the path between two points in the obstacle space, the relevant definition of the shortest obstacle path is introduced, as described in definition 3.

Definition 3 (The Shortest Obstacle Path): Given one sets of obstacles $O = \{o_1, o_2 \ldots, o_n\}$ and two points p and q in one given point sets. The barrier path from point p to point q is called P_{pq}, it can be described as follows. If the path from point p to point q without any obstacles is called the obstacle path between point p and point q. There is more than one obstacle path between any two points in the set. And the shortest path is called the shortest obstacle path, which is defined as SP_{pq}. As shown in figure 1, point p and point q are not visible, and there is an obstacle O_{ab} in the middle. There are two obstacle paths from point p to point q, which are $(pa \rightarrow aq)$ and $(pb \rightarrow bq)$ respectively. The shortest obstacle path between point p and point q is expressed as: $SP_{pq} = min\{dist(pa + aq), dist(pb + bq)\}$.

When the existence of obstacles is taken into account, the distance between two points in space is no longer a Euclidean distance between two points and can't be expressed simply. Therefore, the definition of obstacle distance is derived, as described in definition 4.

Definition 4 (Minimum Obstacle Distance): Given the obstacles set O in two-dimensional space and any two points p and q in the data set, the length of the obstacle path between two points is the obstacle distance between two points, and the length of the shortest obstacle path between two points is the minimum obstacle distance between two points, denoted by $disto(p, q)$. That is the length of the shortest obstacle path between p and q. When point p and point q are visible, the obstacle distance between p and q is the Euclidean distance between p and q, denoted as $diste(p, q)$. The default obstacle distance in this paper is the shortest obstacle distance between two points.

The query object in the obstacle group reverse k distant neighbor query is the query of a group of points. In order to give the definition of the reverse k distant neighbor query in the obstacle space more clearly, the definition of the obstacle reverse k distant neighbor query when the query object is just a point is first given in definition 5.

Definition 5 (Obstacle Reverse k Farthest Neighbors (ORkFN)): In the obstacle space, a set of data points P is given, a set of obstacles O(the set of line segments) and a query point q are alse given. ORkFN query is just to find the query point q in the data set P as the data point set of its k farthest neighbors, which can be defined as follows:

$$ORkFN(q) = \{p \epsilon P | q \epsilon OkFN(p)\} \quad (1)$$

As shown in figure 2, given data object $P = \{p_1, p_2, p_3, p_4\}$, query point q and obstacle o_1, if the line between the two points is a solid line, it means that the two points are visible, which means the Euclidean distance. If the line is a dotted line, it means there is an obstacle between two points, and the dotted line length is the obstacle distance. When obstacles are not considered, $2FN(p_2) = \{p_1, q\}$. When obstacle o_1 appears, p_2's $2FN$ changes, and the result set becomes $2FN(p_2) = \{p_3, p_4\}$. Thus, the existence of visible obstacles affects the results of RkNN query directly.

Based on the above understanding of each element in the obstacle space, the central argument of this paper is given by definition 6: the definition of reverse k distant neighbor query of obstacle group.

Definition 6 (Obstacle Group Reverse k Farthest Neighbors(OGRkFN)): In the obstacle space, given a set of data points P, a set of obstacles O(the set of line segments) and a set of query points Q, OGRkFN query is to find out the data point set that takes any point in the query point set Q as its farthest k neighbor in the data set P.

Figure 3 shows examples of GRkFN query and OGRkFN query. Given data set $P = \{p_1, p_2, p_3\}$, query point set $Q = \{q_1, q_2, q_3\}$, and obstacle O. The solid line represents the Euclidean distance and the dotted line represents the obstacle distance. When obstacles are not considered, $2FN(p_1) = \{q_1, q_2\}$,q_1,$q_2 \epsilon Q$. When obstacles are considered, $2FN(p_1) = \{p_2, p_3\}$, that is, point p_1 is not OGR2FN of query point set Q.

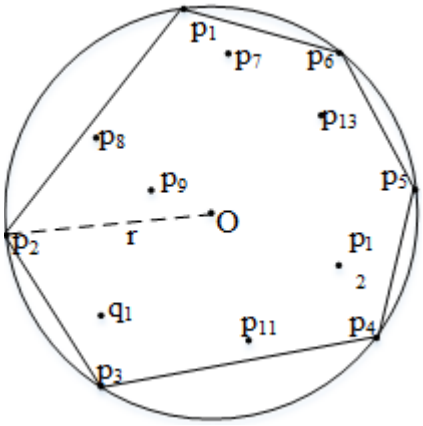


FIGURE 4. Minimum coverage circle.

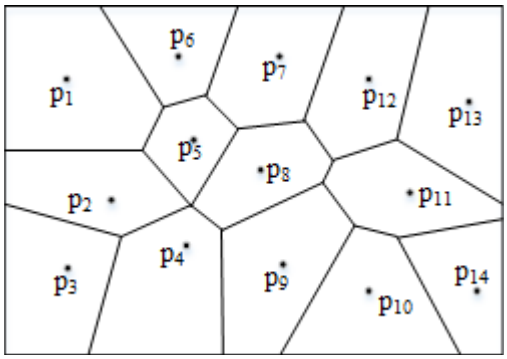


FIGURE 5. Voronoi diagram.

As can be seen from the definition, there are multiple query objects in the group query problem. If a group of query objects are considered one by one, it will incur cost of time. In order to realize the reverse farthest neighbor query of a group of query points quickly, the set of query points needs to be optimized firstly, and all query points should be considered as a whole to reduce the access to data and avoid overlapping of search areas, thus to improve the query efficiency. To this end, the concept of a minimum covering circle is derived as described in definition 7.

Definition 7 (Minimum Coverage Circle [31]): Suppose a set of query objects $Q = \{q_1, q_2 \ldots, q_n\}$, the circle containing all objects in Q with the minimum radius is defined as the minimum covering circle of the set of query points, denoted as $Cir(O, r)$. The point O is the center of the circle and r is the radius, which can be defined formally as a circle domain satisfying as follows:

$$Cir(q, r) = \{q_i | dist(q_i, O) \leq r, q_i \epsilon Q\} \tag{2}$$

As shown in figure 4, data point set $P = \{p_1 - p_{13}\}$, and the circle with point O as the center is the minimum coverage circle of data set P.

Voronoi diagram is an important geometric structure in computational geometry, which can be used to describe the topological relationship between adjacent but it's unrelated to point sets. It provides a powerful tool for solving a series of problems in the field of spatial database. As described in definition 8.

Definition 8 (Voronoi diagram [32], [33]): A set of discrete points with any n distinct positions on a given plane $P = \{p_1, p_2, \ldots, p_n\} \subset R^2$, $2 < n < \infty$, when $i \neq j$, $p_i \neq p_j$. The Voronoi region is denoted as $VR(p_i) = \{p | dist(p, p_i) \epsilon dist(p, p_j)\}$. $dist(p, p_i)$ is the shortest distance between point p and p_i. p_i is Voronoi diagram's generic point. The region $VR(p_i)$ determined by p_i is called Voronoi graph polygon, and the side of Voronoi graph polygon is called $VL(p_i)$. The vertices of a polygon of a Voronoi graph are called the vertices of a Voronoi graph, and the number of edges connected by each Voronoi vertex v is called the degree of v. Voronoi polygons that share the same edge with $VR(p_i)$ are called adjacent polygons of p_i, and their generating points are called first order adjacent generating points of p_i, which are called $AG_1(p_i)$. And $AG_1(p_i) = p_j | \mathrm{VR}(p_i)$ and $VR(p_j)$ have common sides. This leads to the k order adjacency point($k \geq 2$), denoted as: $AG_k(p_i) = \{p_j | VR(p)$ and $VR(p_j)$ have common sides, $p \epsilon AG_{k-1}(p_i)\}$. The Voronoi diagram is shown in figure 5.

From the structure and definition of the Voronoi diagram, the following three basic properties can be obtained.

Property 1: If point q is in the Voronoi polygon $VR(p_i)$, then the distance from q to the point p_i is less than the distance from q to other points.

Property 2: The distance from any point on the edge of the Voronoi diagram to the product of two adjacent Voronoi polygons is equal.

Property 3: For any point p in the Voronoi diagram, p_{k+1}, the nearest neighbor in level $k+1$ of p, has $p_{k+1} \epsilon AG_1(p) \cup AG_2(p) \ldots \cup AG_k(p)(k$ is an integer and $k \geq 1$, $AG_i(\mathrm{p})$ represents the $i-th$ order adjacency point set of the product point p).

Definition 9 (Product points): All the product points appear in this paper is represents the points are generated by the algorithm or some relevant steps. They are only used in the middle of a process.

B. THE RELATED THEOREM

The V-OGRkFN algorithm proposed in this paper obtains the minimum covering circle and its center of the set of query points firstly through the minimum covering circle algorithm of constant complexity, regards a group of query points as a whole, and transforms the query problem of a group of target objects into a single point query skillfully. Secondly, Voronoi graph was constructed for the experimental data set. The pruning strategy based on Voronoi graph was used to filter the data set and obtain the obstacle reverse k nearest neighbor of the target object. Then the candidate set of the reverse k farthest neighborhood of the query point is earned according to the theorem of the conversion between the reverse nearest neighbor and the reverse farthest neighbor. Finally, the candidate set is purified and the result set is obtained by

refining algorithm, which solves the problem of group reverse k farthest neighbor query in obstacle space.

Among them, the mutual transformation theorem between reverse k nearest neighbor and reverse k farthest neighbor is the main idea to the reverse k far neighbor query algorithm of obstacle group. It is assumed that data set $P = \{p_1, p_2, \ldots, p_n\}$ and query point q, if the reverse farthest neighbor of point q is to be watched out. In other words, it is to screen the reverse $(n-1)$ nearest neighbor of dropped point q. By this transformation, many reverse nearest neighbor query algorithms can be applied to reverse farthest neighbor research. The conversion rule is described in theorem 1.

Theorem 1: Given a set of obstacle set O and a set of discrete data points P and a query point q, if P is the farthest neighbor of obstacle reverse k of q, then P must not be the nearest neighbor of obstacle reverse $(n-k-1)$ of q, where n is the sum of all data points including query points. The theorem can be expressed as: $P = \{p_1, p_2, \ldots p_n\}$, $p_i \epsilon P$, $(1 \leq i \leq k)$, $q \epsilon P$. If $p \epsilon$ ORkFN(q), then $p \notin OR(n-k-1)NN(q)$. Vice versa. The proof is as follows:

(1) sufficiency: From the $p \epsilon$ ORkFN(q), the $p \notin$ OR$(n-k-1)$NN(q) can be deduced. Suppose $p \epsilon$ ORkFN(q), for any $p \epsilon$ ORkFN(q), p is the $i-th$ obstacle reverse farthest neighbor of q $(1 \leq i \leq k)$. From definition 7, it can be seen that the most farthest neighbor of p, contains q. Because q is the k farthest neighbor of p, point q is definitely not the $(n-k-1)$ nearest neighbor of p. That is $p \notin$ OR$(n-k-1)$NN(q).

(2)necessity: $p \epsilon$ ORkFN(q) is derived from $p \notin$ OR(n-k-1)NN(q). Assuming $p \notin$ OR$(n-k-1)$NN(q), for any $p \notin$ OR$(n-k-1)$NN(q), it can be known from the definition of the obstacle reverse nearest neighbor that q must not belong to the $n-k-1$ obstacle nearest neighbor of p, namely $q \notin (n-k-1)$NN(P). Since q is not the nearest neighbor of $n-k-1$ obstacle of P, then q is the farthest neighbor of k obstacle of P, namely: qϵ OkFN(P), then it can be known from definition 7 that $p \epsilon$ ORkFN(P).

Note that this theorem only applies to the set of monochromatic points in the obstacle space, and cannot be applied to the set of bichromatic points. In other words, only when the query point and data point set belong to the same class of points can be transformed by this theorem. Cause in monochrome reverse nearest neighbor query, both query point and data point belong to the same data type, and the query point itself is been taken into account. While in bichromatic reverse nearest neighbor query, data point and query point belong to two different types of points, and the query point itself is not required to be considered when transforming in bichromatic query. Therefore, the farthest neighbor of monochrome reverse query is to exclude the nearest neighbor of monochrome reverse $(n-k-1)$ of query point q, while the farthest neighbor of bichromatic reverse query is to exclude the nearest neighbor of bichromatic reverse $(n-k)$ of query point. In this paper, we only discuss the monochromatic point set, and do not discuss the bichromatic point set.

As shown in figure 6, there are 8 discrete data object points $(p_1 - p_8)$, 1 query point q and 4 obstacles (line segment $o_1 - o_4$) in the figure, which means there are 9 interest points (data points and query points). Using the mentionoed theorem, the farthest neighbor of obstacle reverse 2 of query point q is found out. That is the nearest neighbor of obstacle reverse of removing point q. Further analysis is to find out which object points have the nearest neighbor of 6 obstacles that do not contain q, so such object points are the farthest neighbor of the obstacle 2 reverse of q which we are looking for. In figure 4, the nearest neighbor of p_1, O6NN$(p_1) = \{p_2, p_3, p_4, p_5, p_6, p_7\}$, namely q $O6NN(p_1)$. So p_1 is OR2FN of q. Similarly, it can be found out that the result set of the obstacle 2 reverse furthest neighbor query of point q is OR2FN(q) $= \{p_1, p_2, p_3, p_4\}$.

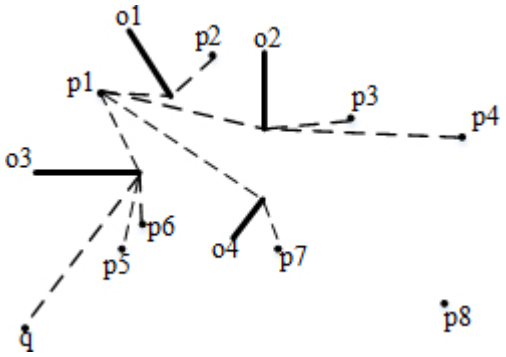


FIGURE 6. Schematic diagram of the obstacle reverse k farthest neighbor query.

The main work of the pruning process is to optimize the experimental data point set, and then to cut off a large number of meaningless points. The remaining data points that have not been pruned will take into other consideration for further query. Three pruning strategies and their related proofs are given below in the form of theorems.

Theorem 2: Only the former k-order adjacency products of the nearest neighbor product point of query point q can be included in the candidate set of the reverse k-neighbor query of query point q, that is, the k-order of the nearest neighbor product point of query point q and the adjacent products and obstacles beyond the k-order are pruned.

Prove: Assuming that the query point q is in the Voronoi graph polygon with point p as the product point, the related properties of the Voronoi graph can be obtained: $q_{k+1} \epsilon AG_1(p) \cup AG_2(p) \cup \ldots \cup AG_k(p)$,($k$ is an integer and $k \geq 1$). That is, the nearest neighbor of $k+1$ of q is in the $pre-k$ order adjacency product point of p, so kNN of q cannot exist in the k order of q and the adjacency product point higher than k order. If $p_j \epsilon AG_n(p)(n > k)$. In other words, the result set of RkNN query of q must not contain point p_j. In the case of obstacles, the obstacle distance from point p_j to point q must be huger or equal to the Euclidean distance between two points, so ORkNN of query point q is less likely to contain point p_j. Which means, the adjacent product points and obstacles beyond k-order and k-order in the region where point q is located are all pruned. Theorem 2 is proved.

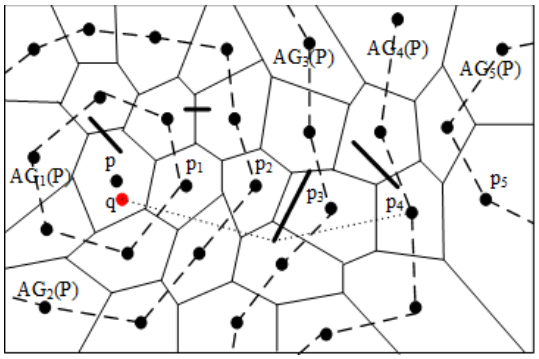


FIGURE 7. Schematic proof of theorem 2.

As shown in figure 7, the nearest neighbor of query point q is point p, and points p_1,p_2,p_3, p_4 and p_5 are any of the 1-order to 5-order adjacency product points of point p respectively. When $k = 3$, according to the property 3 of Voronoi diagram, 3NN of point p_4 is in the first 3 adjacent points of p_4, while point p is the 4-order adjacent point of point p_4, so: $p \notin$ 3NN(p_4). Since query point q is located in $VR(p)$, point q is not 3NN of point p_4, namely $p_4 \notin$ R3NN(q). Similarly, R3NN of q is not $p_5, p_6 \ldots, p_n$. Cause there are obstacles between point p_4 and query point q, the obstacle distance between them must be larger than the Euclidean distance. Therefore, point p_4 in the obstacle space is more unlikely to be the result set of R3NN query of query point q.

Theorem 3: If the data point p on the nearest neighbor query point q n level adjacency involves, p to q after a certain number of data points on the path to the nearest p_0, as p_i, to make a single path through the number of data points to no more than n (including attention, p and p_0 not included in the n), and p visual p_i as the number of combined totalCount (p, p_0), if totalCount $(p, p_0) \geq k$, data point p is pruning.

Prove: When $k = 1$, the path from p to p_0 can only pass through one data point p_i at most. If p_i and p are visible, that is, the distance between them is Euclidean distance, then there must be $diste(p, p_i) < dist(p, p_0)$; When $k > 1$ is true, if the sum of the number of points visible to p on all paths from p to p_0 is larger than or equal to k, then the k nearest neighbor of p, the OkNN of p may contain 1-k of p_i, but it must not contain q.

As shown in figure 8, data point p_{16} is in the 3-order adjacency product point of the nearest neighbor p_0 of query point q. At this point, there are two paths in which the number of data points passing on the path from p_0 to p_{16} is no more than 3, respectively $\{p_0, p_{12}, p_7, p_{16}\}$, $\{p_0, p_6, p_{13}, p_{16}\}$. The sum of data points on two paths between p_{16} and query point q is 4, respectively $\{p_6, p_7, p_{12}, p_{13}\}$. But there are only three data points p_7, p_{12} and p_{13} are visible with p_{16} on these two paths, we define it as totalCount $(p_{16}, p_0) = 3$. When $k = 3$, 3NN of p_{16} may include $\{p_7, p_{12}, p_{13}\}$, but it must not contain point q. When k is less than 3, kNN of p_{16} may contain 1-k of $\{p_7, p_{12}, p_{13}\}$, but it must not contain q. To sum up, when totalCount (p, p_0) is larger than or equal to k, the result set

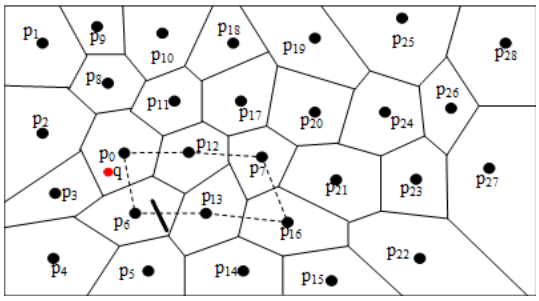


FIGURE 8. Proof schematic diagram of theorem 3.

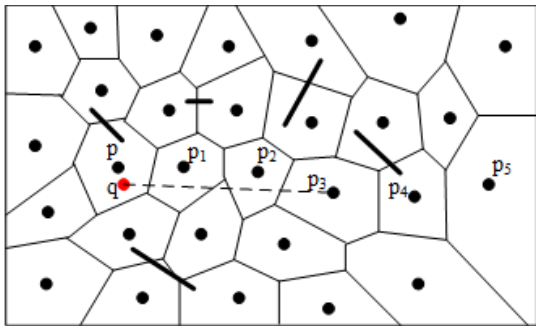


FIGURE 9. Proof schematic diagram of theorem 3.

of p's kNN query must not contain point q, which means point p must can't be included in the ORkNN query result of q. For this reason, data point p is pruned.

Theorem 4: It is assumed that there are several generating points between query point q and data point p, which are collectively referred to as point p_i. p_i and p are connected. If the line does not intersect with any obstacles, the counter value will increase by 1. Data point p will be pruned when the counter value is larger than the k

Prove: Since there are several Voronoi polygons with point p_i as the product points between query point q and point p, suppose that point p is the k-th adjacency product point of the nearest neighbor product point of query point, then point p_i must be the k-th adjacency product point of point p. If such point p_i and point q are visible, then $dist(q, p_i) < dist(q, p)$, When the number of points p_i is visible to point p exceeds k, then point q must not be the result set of p's ORkNN. Theorem 4 is proved.

As shown in figure 9, the path from point q to point p_3 in the Voronoi graph polygon with point p as the generating point passes through the Voronoi diagram region with point p_1 and p_2 as the generating points, and the query point q is visible to both p_1 and p_2. Thus, there are $dist(q, p_1) < dist(q, p_3)$ and $dist(q, p_2) < dist(q, p_3)$. So $p3 \notin$ R2NN(q).

Theorem 5: If the first-fliter adjacent product points of data point p in the candidate set are all pruned, then the point p is also pruned.

Prove: Suppose point p for any point of the candidate set, several points s for one of the primary adjacency involves point p, and s is p to q of the nearest neighbor involves p_0 path through the point, if the level of p adjacency

Algorithm 1 V-OGRkFN Algorithm

Input: Dataset P, Queryset Q, k
Output: Resultsets

```
CreateVoronoi(P);
q ←Minimum Coverage Circle Algorithm(Q);
NNk ←P.Count − 1 − k;
Sc ←firstFilter(P, q, NNK);
if Sc ≠ null then
    Sc ←secondFilter(P, q, NNK);
    if Sc ≠ null then
        Sc ←thirdFilter(P, q, NNK);
        Fc ←getReverse(P, Sc);
        ResultSets← ofOFRange − K(Fc, k);
        return ResultSets;
    end
end
```

involves are pruned, then the data point s are pruned clearly, namely totalCount $(s, p_0 \geq k)$, so there must be totalCount $(p, p_0) \geq k + 1 > k$. Point p will be pruned, theorem 5 is proved.

IV. THE V-OGRkFN ALGORITHM

The reverse k farthest neighbor query algorithm is proposed in this section. We call it as V-OGRkFN(Voronoi-Obstacle Group Reverse k Farthest Neighbors) algorithm. Firstly, the minimum coverage circle of the query points is obtained, and we take a group of query points as a whole into consideration to reduce the access to data. Secondly, the index structure of Voronoi is constructed for the experimental data point set, and the pruning strategy is designed according to Voronoi's properties for cutting off the nonsense points and obstacle sets, so as to obtain the obstacle reverse $(n - 1 - k)$ nearest neighbor candidate set for the query points. Then, the candidate set of obstacle reverse k farthest neighbor is obtained by the conversion algorithm with constant time complexity. Finally, the final result set is obtained by the refining algorithm. The specific implementation of V-OGRkFN query algorithm's pseudo code is shown in algorithm 1. First, the Voronoi index structure was constructed for the set of experimental data points. A set of query point sets was considered as a whole by the minimum covering circle algorithm, and the center of the minimum covering circle was recorded as the query point. The required "k" value in the reverse "k" nearest neighbor query was deduced according to theorem 1. The pruning strategy mentioned in theorem 2-5 was used to obtain the obstacle reverse k nearest neighbor result set. Finally, the candidate set of the obstacle reverse k farthest neighbor query was obtained through the transformation theorem, and the final accurate result set was obtained through the refining algorithm. The minimum coverage circle algorithm, pruning algorithm, conversion algorithm and refining algorithm will be introduced respectively below.

Algorithm 2 Minimum Coverage Circle Algorithm(Q)

Input: Querysets Q
Output: Center of the circle q

```
for each point i in the QuerySets do
    if incircle(querySets[i]) = false then
        P1 ← querySets[i] ;
        for each j in i do
            if incircle(querySets[j]) = false then
                P2 ←getCenterPoint(querySets i,j]);
                Radius q ←getDist(P1,querySets[i]);
            end
            for each k in j do
                if incircle(querySets[k]) = false then
                    P2 ← getCenterPoint(querySets i,j);
                    Radius q ←
                    getDist(P1,querySets[i]);
                    return q;
                end
            end
        end
    end
end
```

A. MINIMUM COVERAGE CIRCLE ALGORITHM

The minimum covering circle algorithm is used to solve the center of the minimum covering circle of the query point set. Although it is an approximate estimate, it can still better meet the application requirements in practice. This algorithm can effectively transform the group reverse farthest neighbor query into a reverse farthest neighbor query of a query point. The specific minimum coverage circle algorithm is shown in algorithm 2.

The method to calculate the minimum covering circle of the given point set has been given in literature [31], but the time complexity of this method is $O[lg(d/R)n]$, where: d represents the distance from the nearest point outside the circle to the circumference, and R represents the radius of the minimum covering circle. The time complexity of the algorithm for calculating the minimum coverage circle mentioned in this section is linear: $O(n)$. It is not regarded the time complexity of this method is $O(n^3)$, it is $O(n)$ in real. The time complexity of the last layer of the for loop is $O(j)$. Next we consider the penultimate layer of for loop, the mathematical expectation of the number j point is having $3/j$'s probability not in the circle forms by the whole number of $j - 1$. In this case, a $O(j)$ third layer cycle is required. The overall time complexity is $O(1)$. If p_j is inside the circle, the time complexity is also $O(1)$, so the total complexity of the second layer loop is $O(i)$. For the same reason, the total complexity of the first layer loop is $O(n)$.

The steps of this algorithm are described as follows:

(1)First, a set of randomly distributed query point set Q is obtained.

(2)The data points are added to the circle one by one, and the position of the point and the circle needs to be determined for each point added.

(3)If the current point p_i is outside the circle, that is, the point p_i must be on the boundary of the minimum covering circle of all the previous points, we perform step (4) to continue to determine the minimum circle, otherwise no update is needed, and return (2).

(4)At this point, point p_i must be on the boundary of the current minimum covering circle. The current point p_i can be set as the center of the new circle with a radius of zero. Then, the previous $i-1$ point is added to the circle through the above steps.

(5)If point p_j is outside the current minimum circle, that is, point p_j must be on the boundary of the current minimum covering circle, we continue to perform step (6). Otherwise no update operation is needed, and return (4).

(6)Point p_i, p_j must be in the current round of minimal covering on the border, at this time it will be the first point with the first j point of attachment point set as the new circle's center. The distance between two points set for the new round radius, and then through the above steps will $j-1$ point before to join in this circle, each join point need to perform the next step (7).

(7)If point p_k is on the outside of the current circle, that is, point p_j must be on the boundary of the minimum covering circle of the first k points because of three points can determine a circle and the center of the common circle of these three points can be directly calculated. Otherwise no update operation is needed.

B. PRUNING ALGORITHM

According to the pruning strategy mentioned in theorem 2, the first order pruning algorithm FirstFliter is given. The idea of the algorithm can be summarized roughly as: get the nearest neighbor p of the obstacle of query point q, put the first k-order adjacency product point of p into the candidate set, and the last of point sets and obstacles are pruned. In this process, the Voronoi diagram needs to be built for the experimental data point set firstly, and then the position of the query point in the Voronoi diagram can be determined. There are three possibilities:

(1) point q is inside VR(p), so Sc is the set of pre-k adjacent product points of p;

(2) Point q is on the common edge of VR(p_1) and VR(p_2), so Sc is the union of the set of $pre-k$ adjacent product points of p_1 and p_2.

(3) When the query point q and a Voronoi diagram polygon vertices v overlap, you first need to consider how much is the vertex degrees, usually 3, when the query point q in VR(p_1) and VR(p_2) and VR(p_3) public vertex, then Sc is p_1, p_2, p_3, a former k level adjacency involves collection and set, when the vertex v degree greater than 3, the same rules apply.

Based on the above discussion, the first-level pruning algorithm in the query process is given, as shown in algorithm 3.

Algorithm 3 The Algorithm of FirstFliter(P, q, k)

Input: datasets P, Center of the circle q, k
Output: First candidate set: first_Sc

```
get the position of q in Voronoi;
if onVertice(q) then
    pointSet←GetPPoint(q);
end
for each p in pointSet do
    first_Sc.add(getK(p));
    if OnEdge(q) then
        pointSet←GetEadgPoints(q) ;
        for each p in pointSet do
            first_Sc.add(getK(p));
        end
    end
    point← GetONN(q);
    first_Sc.add(getK(point));
end
return first_Sc;
```

(1) According to the first-fliter pruning algorithm proposed in theorem 2, the first step is to obtain the position of point q in the Voronoi diagram. If point q is on the vertexes of the Voronoi diagram, then get the set of product points corresponding to the Voronoi diagram polygon containing point vertex q by GetPPoint(q) method. If point q is on the side of the Voronoi polygon, then get the set of product points corresponding to the Voronoi polygon on the side of point q by GetEdgePoint(q) method. Otherwise, point q is inside the Voronoi. The method of getK(p) is used to obtain the first k order adjacency point of the nearest neighbor product point of point q obstacle. The final set of points obtained as the candidate set of first-order pruning.

(2) GetPPoint(q): Gets the product points corresponding to the common vertices. And Voronoi Vertices is the vertices set in the Voronoi graph. Loop through the vertices in the VoronoiVertices collection, and if it overlaps with q, then loop through the delaunay triangle collection allTriangle, and get the center point of the outer circle of each triangle. If point q coincides with the center point of the circumscribed circle, the three vertices of the triangle are stored in the set of product points. According to the relationship between Voronoi diagram and delaunay triangulation network, the vertices of triangles in the triangulation network correspond to the generating points of Voronoi diagram, and the vertices of Voronoi diagram are the center of the external circle of triangles in the triangulation network, so the above algorithm idea is feasible.

(3) GetEdgePoint(p): Gets the product points corresponding to the common edges. In the algorithm, VoronoiEdgeList is the edge set of Voronoi graph. First traversal VoronoiEdgeList edge set of each edge, obtain corresponding vertex assigned to each edge point a and b, through onLine (q, a, b) method for boolean whether point q in the a and b on the side of the vertices. If it's true, the point q is on the edge of ab, and traverse the delaunay triangulation of the triangle

Algorithm 4 The Algorithm of SecondFilter (*first_Sc*,*q*,*k*)

Input: datasets *P*; candidate: *first_Sc*, center of the circle *q*, *k*
Output: second_Candidate: *second_Sc*

```
count←0;
for each n in first_Sc do
    dist ← getDistance(q, n);
    pSets ← GetPointInCircle(P, dist, n);
    for each m in pSets do
        flag ← JudgeVisible(m, n);
        if (!flag) then
            count + + ;
        end
        if count≥ k then
            first_Sc.remove(m) ;
        end
        second_Sc ← first_Sc;
    end
end
return second_Sc;
```

triangleEdgeList edge set, for each edge corresponds to the two vertices and assign a value to the point m and n. Finally using cross product to find the a, b and m, n as vertices of intersection respectively. If they are intersect, then m and n are the product points.

(4) GetONN(q): The obstacle nearest neighbor product point for obtaining point q. When q is in the Voronoi polygon, the obstacle nearest neighbor of the query point is first obtained through GetONN(q). If obstacles are not taken into account, the closest point to point q must be the corresponding point of Voronoi where point q is located. However, this rule cannot be completely guaranteed to hold true in the obstacle space, so it needs to be judged. This algorithm by using getGenerator(q) method for polygoning if point q is involves, and then get the involves the level of the adjacent involves. Next traverse the requested level of adjacency involves in the collection points, the distance to the query point q if the calculated distance is less than the distance of the generator to q. Finally the distance between the corresponding involves is requested, otherwise the generator for prayer points.

According to the pruning strategy of filtering data sets according to the visibility of points between two points proposed in theorem 4. The second pruning algorithm SecondFilter in the query process of V-OGRkFN algorithm is given. The specific algorithm description is described in algorithm 4.

First of all, initialize a counter count, loop through the element n in the first-level candidate set, and get the distance *dist* between point n and point q. Take point q as the center of the circle and *dist* as the radius of the circle to obtain the points in the circle in dataset P. Loop through point set m in the circle and judge whether point m and point n are visible. If it's true, the counter will increase 1. When count

Algorithm 5 The Algorithm of ThirdFilter (*second_Sc*,*q*,*k*)

Input: Second_Candidate *second_Sc*, querypoint *q*,*k*
Output: Third_Candidate: *t_Sc*

```
for each p in second_Sc do
    adjSets ← getAdjPoints(p);;
    if !insectContain(adjSets, second_Sc) then
        t_SC.remove(p);
        t_Sc ← third_Sc;
    end
end
return t_Sc;
```

to k, the k nearest neighbor of point n must not contain point q. Which means, the reverse k nearest neighbor of point q does not contain point n, which is the pruning point n. The unconstructed set of points constitutes a second-filter candidate set.

According to the pruning strategy proposed by theorem 5 to filter the data set according to the pruning situation of the first-fliter adjacent product points of the candidate concentration point set. The third-fliter pruning algorithm ThirdFilter in the query process of V-OGRkFN algorithm is given next step, it is shown in algorithm 5.

Data in a loop through the secondary candidate set *second_Sc* point p. Retrieve the first-level adjacency points of p and place them into the collection adjSets, judge adjSets and the second of candidates *_Sc* elements in the existence of intersection. If there is no intersection, that level of adjacency involves around the element p was pruning. On the basis of theorem 5, point p will be pruning.

C. TRANSFORMATION ALGORITHM

According to theorem 1, the transformation relation between reverse k nearest neighbor and reverse k farthest neighbor can solve the query problem of reverse k farthest neighbor in obstacle space effectively. The V-OGRkFN algorithm mentioned in this chapter reduced the data set with the minimum covering circle algorithm mentioned above effectively and the Voronoi based pruning algorithm. Both all obtained the result set of the obstacle reverse $(n-1-k)$ neighbor query. Then, the difference set between the result set of the obstacle reverse $(n-1-k)$ neighbor query and the data set set P was calculated through the conversion algorithm GetReverse, so as to obtain the candidate set of OGRkFN query. The conversion algorithm is shown in Algorithm 6.

Loop through element s in candidate set Sc and make judgement whether point s exists in datasets in turn. If it does, point s is removed from datasets. The reserved point set will become candidate set.

D. REFINING ALGORITHM

The V-OGRKFN query algorithm is a query optimization algorithm based on filtering and refining framework. The refining process is the refining algorithm proposed in this

Algorithm 6 The Algorithm of GetReverse(*Sc*,*q*)

Input: third_candidate: *t_Sc*, Datasets: *P*
Output: candidate: reverseSets

```
for each s in t_Sc do
    if P.contain(s) then
        P.remove(s);
        reverseSets ← p;
    end
end
return reverseSets;
```

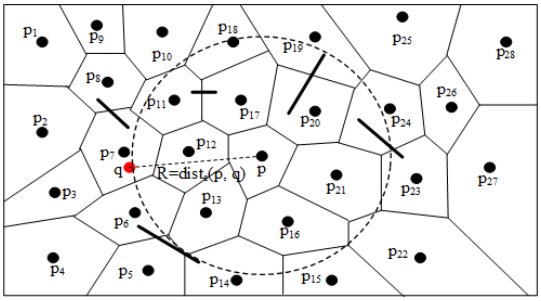


FIGURE 10. Schematic diagram of refining algorithm 1.

section – Obstacle farther-range query algorithm (OFRange-*k*). This algorithm is improved on the basis of range-*k* algorithm, which can refine candidate sets and remove incorrect result sets effectively, which ensures the accuracy of query results.

For a given data set *P*, obstacle set *O* and query point *q*, the candidate set of obstacle reverse *k* farthest neighbor query of point *q* can be obtained through the above pruning algorithm and transformation algorithm. In the refining stage, data points in the candidate set need to be verified one by one, and the verification is mainly divided into the following two aspects:

If the data point *p* and query point *q* are visible, the data point *p* is the center of the circle, the Euclidean distance between *p* and *q*, $diste(p, q)$ is the radius of the circle, and the counter count is used to count the number of data points outside the circle. Because the data points outside the circle regardless of whether or not the visual and centre point *p*, the distance between two points will be larger than $diste(p, q)$, so when the count≥k was founded, the query point *q* will not appear in the data point *p*'s query result set of kFN, namely q∉ kFN(*p*). In other words, data point *p* must not be reverse *k* point *q* distant neighbors query result set, which named p∉RkFN(*q*), the data points are eliminated *p*.

As shown in figure 10, point *q* and *p* are invisible, if want to refining of candidate focused point *p*, to *p* as the center of the circle, the distance between two points at point *q* and *p* for the radius of a circle. At this point, there are 20 points outside the circle, when requested obstacle space under reverse *k* farthest neighbors query in *k* value less than or equal to 20, then the data point *p* will be removed.

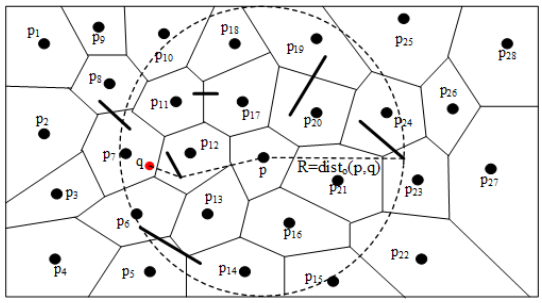


FIGURE 11. Schematic diagram of refining algorithm 2.

If the candidate concentration point *p* is not visible to the query point *q*, then point *p* is still taken as the center of the circle. At this time, the obstacle distance $disto(p, q)$ between two points is taken as the radius for the circle. Then count the number of data points outside the circle. If count≥k, it represents the point *p* need to be pruned.

As shown in figure 11, refining data points *p*, because of point *p* with the query point *q* is invisible, obstacle distance between two points in $disto(p, q)$ as the radius of a circle centered on *p*, there are 14 points outside the circle when the distant neighbors query for obstacle space under the *k* value of *k* is less than or equal to 14. So the data point *p* will be removed.

Based on the above discussion, the refined algorithm is given nextly, as is shown in algorithm 7.

Iterate over the data point *p* in the candidate set *SC*, make judgement between the points *p* and *q* if they are visible. If the flag is false, namely the two points *p* and *q* are visible. Then the Euclidean distance are calculated between two points called radius. Next step, determine whether data collection of dataSets in elements in query point *q* is as the center of the circle. If the result trues wrong, then the counter count will plus 1. When $count \geq k$ was established, the data point *p* will be removed. If flag is true, which means, point *p* and point *q* are not visible, then the obstacle distance between points *p* and *q* is calculated. Similarly, continue to judge the number of points outside the circle with the query point *q* as the center and the obstacle distance as the radius in dataset *p*. If count *k* is established, point *p* will be corrected. The set of reserved points is gaind.

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. EXPERIMENTAL ENVIRONMENT SETTING AND DATA

The experimental operating environment is configured as follows:64-bit Microsoft Windows 7 operating system, hardware environment Intel(R) Core(TM) i5-2330m CPU @ 2.20ghz, 4-core, 8GRAM. All experiments were conducted using C# development language and Visual Studio 2010 development tool. Among them, the data objects studied are all distributed in regions with a two-dimensional data region of 0<x<700, 0<y<700. Test data set, obstacle set and query point are generated randomly by calling random function in C# language in VS2010 environment.

Algorithm 7 The Algorithm of OFRange-$k(Sc,q,k)$

Input: Candidate: SC, query point q,k
Output: result: resultSets

```
count ←0;
for each p in SC do
    flag ← JudgeVisible(p, q);
    if !flag then
        radius ← getDistance(p, q);
        reverseSets← p;
        for each s in dataSets do
            flag ← SiteInCircle(s, radius, q);
            if !flag then
                count + +;
            end
            if count ≥k then
                SC.remove(p);
            end
        end
    end
    else
        radius ← getObacleDistance(p, q);
        for each s in dataSets do
            flag ← SiteInCircle(s, radius, q);
            if flag.count ≥ k then
                SC.remove(p);
            end
        end
    end
    resultSets←SC;
end
return reverseSets;
```

B. ANALYSIS OF EXPERIMENTAL RESULTS

In this paper, the traditional algorithm Basic-Obstacle Reverse k Farthest Neighbor(B-OGRkFN)query is adopted as the comparison algorithm and the V-OGRKFN algorithm proposed in this paper is compared and tested. The traditional algorithm firstly through the acquisition of point set in the center of mass for a query point, and by calculating the distance barriers to obtain each data point k distant neighbors, if included in the query results query any point in point set Q, then the data points can be incorporated into the query point set Q obstacles set reverse k distant neighbors query result set. The specific analysis of the experiment is described below.

Firstly, the impact of different data set sizes are tested efficiency. In order to follow the principle of single variable, the experiment set other attribute values of the two groups of algorithms to be consistent, which are: obstacle set $O = 1 \times 10^2$, $k = 10$, query point set $Q = 1 \times 10^2$. As shown in figure 12, with the increase of data set size, the response time of both algorithms increases, as the increase of data set size will increase the execution time of pruning algorithm in the query process, so the query time will gradually become

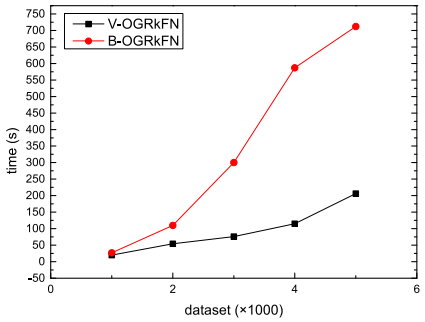


FIGURE 12. The effect of data set size on query efficiency.

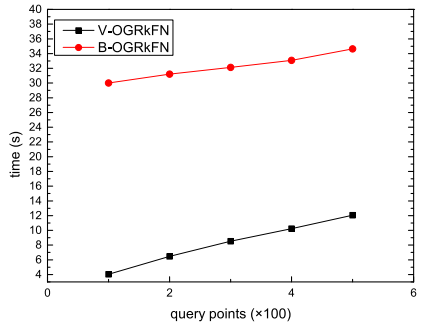


FIGURE 13. The impact of query point set size on query efficiency.

longer. It can be find that in the comparison when the data set size is small, the query time gap between the two is relatively small, but with the increase of the data set size, the gap will be quickly larged. In addition, from the perspective of growth trend, the growth rate of V-OGRkFN algorithm's running time is close to 0, while B-OGRkFN algorithm's running time is gradually increasing, and the growth trend is significant, which shows that V-OGRkFN algorithm has better query performance.

Secondly, the impact of different query point set sizes on query efficiency was tested. In order to maintain a single variable, other attribute values of the two groups of algorithms were set as follows: data point set $P = 1 \times 10^3$, obstacle set $O = 1 \times 10^2$, $k = 10$. As shown in figure 13, both from the perspective of growth trend, the rate of growth are relatively small and the curve rises gently, indicating that the size of the query point set does not have a great impact on the query time. On the whole, the time consumption of B-OGRkFN algorithm is much greater than the running time of V-OGRkFN algorithm, which indicates that V-OGRkFN algorithm has better query performance.

Thirdly, the impact of different obstacle sets on query efficiency was tested. In order to maintain a single variable, the experiment set other attribute values of the two groups of algorithms to be consistent: data point set $P = 1 \times 10^3$, query point set $Q = 1 \times 10^2$, $k = 10$. As shown in figure 14, the response time of both algorithms increases with the increase of obstacle set capacity, which is mainly because of the increase of obstacle set capacity will increase the number of obstacle distance calculation, so the query

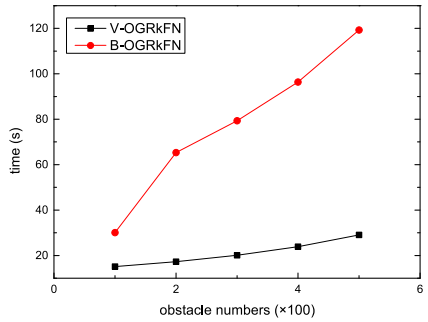


FIGURE 14. The influence of obstacle set size on query efficiency.

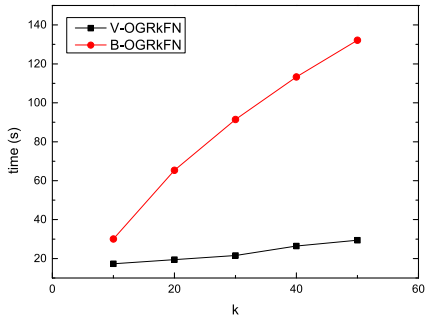


FIGURE 15. The impact of k value size on query efficiency.

time will gradually increase. The comparison shows that the running time of B-OGRkFN algorithm increases gradually and has a significant growth trend, while the V-OGRkFN algorithm tends to be 0, which is mainly due to the fact that Voronoi-based pruning algorithm can filter data sets and obstacles effectively. This shows that the V-OGRkFN algorithm has a better query performance.

Next step, the impact of different k values on query efficiency was tested. In order to maintain a single variable, other attribute values of the two groups of algorithms were set to be consistent in the experiment, which were: data point set $P = 1 \times 10^3$, query point set $Q = 1 \times 10^2$, and obstacle set $O = 1 \times 10^2$. As shown in figure 15, with the increase of k value, the response time of both algorithms increases, mainly because of the number of required results increases, so the query time will gradually increase. However, the comparison shows that the query time growth rate of the V-OGRkFN algorithm is close to 0, while the running time of B-OGRkFN algorithm is gradually increasing with a significant growth trend, which indicates that the V-OGRkFN algorithm has a better query performance.

Finally, the impact of test obstacle rate on experimental efficiency was tested. Data point set $P = 1 \times 10^3$, query point set $Q = 1 \times 10^2$, $k = 10$. As shown in figure 16, with the increase of obstacle rate, the response time of both algorithms increases, mainly because of the obstacle density increases and the time needed to calculate the obstacle distance increases, as the query time will gradually become longer. However, the comparison shows that the running time of B-OGRkFN algorithm increases gradually and has a

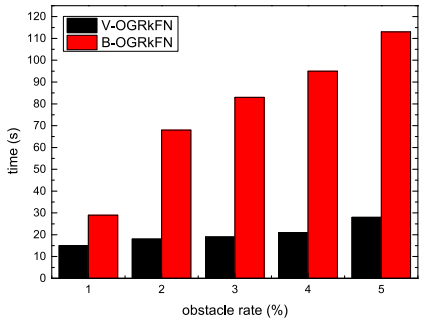


FIGURE 16. The impact of obstacle rate on query efficiency.

significant growth trend, while the query time growth rate of V-OGRkFN algorithm is close to 0, which is mainly due to the fact that Voronoi-based pruning algorithm can filter data sets and obstacles effectively. This shows that the V-OGRkFN algorithm has a better query performance.

VI. CONCLUSION

In this paper, the concept of group reverse k farthest neighbor query in obstacle space is defined for the first time, and a query optimization algorithm based on Voronoi graph which named V-OGRkFN algorithm is given. The algorithm used the minimum cover circle algorithm to change the query point set as a whole firltly, and then build Voronoi index by using the experimental set. Next step to filter data points by a series of pruning strategy based on Voronoi's properties, and with the transformation of the k nearest neighbor and the k farthest neighbors to get the candidate set. Finally, through the refined algorithm to focus the point set of candidates were purified and the final result set are geted, it solved the group reverse k farthest neighbors query problems in the obstacles. An experimental system is constructed based on the proposed algorithm. The V-OGRkFN algorithm and B-OGRkFN algorithm are compared and analyzed respectively through data sets of different sizes, query point sets, obstacle sets, as well as different k values and different obstacle rates, to verify that the algorithm proposed in this paper has good query performance and robustness.

Although this way is fast and convenient solution for group k farthest neighbor query problems, but there are still some shortcomings and limitations. For example, in this paper, the proposed algorithm is only for static data objects, and doesn't take the presence of moving objects in reality into consideration, it can be discussed in our future futher studies.

REFERENCES

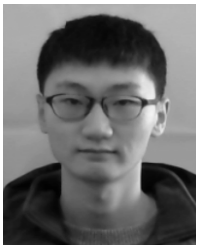
[1] F. Korn and S. Muthukrishnan, "Influence sets based on reverse nearest neighbor queries," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Dallas, TX, USA, 2000, pp. 201–212.

[2] L. Bohan and Z. Hao, "Effective filtering and Query algorithm for reverse farthest neighbor," (in Chinese), *J. Chin. Comput. Syst.*, vol. 30, no. 10, pp. 1948–1951, Feb. 2009.

[3] B. Yao, F. Li, and P. Kumar, "Reverse furthest neighbors in spatial databases," in *Proc. IEEE 25th Int. Conf. Data Eng.*, Shanghai, China, Mar. 2009, pp. 664–675.

[4] Q. T. Tran, D. Taniar, and M. Safar, *Reverse k Nearest Neighbor and Reverse Farthest Neighbor Search on Spatial Networks*. Berlin, Germany: Springer-Verlag, 2009, pp. 353–372.
[5] J. Liu, H. Chen, and K. Furuse, "An efficient algorithm for reverse furthest neighbors Query with metric index," in *Proc. Database Expert Syst. Appl., Int. Conf.*, Bilbao, Spain, 2010, pp. 437–451.
[6] Y. Gao, L. Shou, and K. Chen, "Aggregate farthest-neighbor queries over spatial data," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, Hong Kong, 2011, pp. 22–25.
[7] J. Liu, H. Chen, and K. Furuse, "An efficient algorithm for arbitrary reverse furthest neighbor queries," in *Proc. Asia–Pacific Int. Conf. Web Technol. Appl.*, Berlin, Germany, 2012, pp. 68–72.
[8] A. Said, B. Fields, and B. J. Jain, "User-centric evaluation of a K-furthest neighbor collaborative filtering recommender algorithm," in *Proc. ACM Conf. Comput. Supported Cooperat. Work*, Dallas, TX, USA, 2000, pp. 1399–1408.
[9] H. Wang, K. Zheng, and H. Su, "Efficient aggregate farthest neighbour Query processing on road networks," in *Proc. Australas. Database Conf.*, Cham, Switzerland, Aug. 2014, pp. 13–25.
[10] B. Li, "Dynamic reverse furthest neighbor Querying algorithm of moving objects," *J. Chin. Comput. Syst.*, vol. 30, no. 10, pp. 226–279, Feb. 2016.
[11] S. Wang, M. A. Cheema, X. Lin, Y. Zhang, and D. Liu, "Efficiently computing reverse k furthest neighbors," in *Proc. IEEE 32nd Int. Conf. Data Eng. (ICDE)*, Helsinki, Finland, May 2016, pp. 1110–1121.
[12] X.-J. Xu, J.-S. Bao, B. Yao, J.-Y. Zhou, F.-L. Tang, M.-Y. Guo, and J.-Q. Xu, "Reverse furthest neighbors Query in road networks," *J. Comput. Sci. Technol.*, vol. 32, no. 1, pp. 155–167, Jan. 2017.
[13] D. Li, B. Li. Chao Zhang, "Probabilistic reverse farthest neighbor Query algorithm for uncertain moving objects," (in Chinese), *J. Chin. Comput. Syst.*, vol. 38, no. 2, pp. 282–286, Nov. 2017.
[14] R. Liu and Z. Hao, "A multi-order based index structure for spatial data—MOIS-tree," (in Chinese), *Chin. J. Comput. Res. Develop.*, vol. 47, no. 5, pp. 849–857, Feb. 2010.
[15] H. Wang, Y. Gu, and G. Yu, "Reverse k furthest neighbor Query verification technology in outsourced spatial database," (in Chinese), *Chin. J. Comput.*, vol. 41, no. 08, pp. 1896–1911, Feb. 2018.
[16] D. Papadias, Q. Shen, and Y. Tao, "Group nearest neighbor queries," in *Proc. 20th Int. Conf. Data Eng.*, Boston, MA, USA, 2004, pp. 301–312.
[17] H. Li, H. Lu, B. Huang, and Z. Huang, "Two ellipse-based pruning methods for group nearest neighbor queries," in *Proc. Int. Workshop Geographic Inf. Syst. (GIS)*, Bremen, Germany, 2005, pp. 566–570.
[18] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui, "Aggregate nearest neighbor queries in spatial databases," *ACM Trans. Database Syst. (TODS)*, vol. 30, no. 2, pp. 529–576, Jun. 2005.
[19] M. L. Yiu, N. Mamoulis, and D. Papadias, "Aggregate nearest neighbor queries in road networks," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 820–833, Jun. 2005.
[20] X. Lian and L. Chen, "Probabilistic group nearest neighbor queries in uncertain databases," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 6, pp. 809–824, Jun. 2008.
[21] X. Song and C. Yu, "GRkNN: Group reverse k nearest neighbor Query in spatial database," (in Chinese), *Chin. J. Comput.*, vol. 33, no. 12, pp. 2229–2238, Feb. 2010.
[22] D. Sun and Z. Hao, "Group nearest neighbor Query based on Voronoi graph," (in Chinese), *Chin. J. Comput. Res. Develop.*, vol. 47, no. 7, pp. 1244–1251, Feb. 2010.
[23] M. Chen, Z. Jia, and G. Yu, "Constrained group nearest group Query method in spatial database," (in Chinese), *Chin. J. Northeastern Univ. Natural Sci.*, vol. 32, no. 05, pp. 630–633, Dec. 2011.
[24] Z. Yang and Z. Hao, "Research on group obstacle nearest neighbor Query in spatial database," (in Chinese), *Chin. J. Comput. Res. Develop.*, vol. 50, no. 11, pp. 2455–2462, Feb. 2013.
[25] Y. Guo, L. Zhang, and S. Li, "Nearest neighbor Query of line group based on Voronoi diagram in spatial database," (in Chinese), *J. Chin. Comput. Syst.*, vol. 38, no. 10, pp. 2341–2345, Feb. 2017.
[26] S. Li, J. Jia, and X. Hao, "Probability threshold group k nearest neighbor Query method for uncertain Voronoi diagram," (in Chinese), *J. Chin. Comput. Syst.*, vol. 38, no. 1, pp. 44–48, Feb. 2017.
[27] M. Li, Y. Gu, and M. Chen, "Efficient processing method for top-k Query of reverse spatial preference," (in Chinese), *J. Softw.*, vol. 28, no. 2, pp. 310–325, Nov. 2017.
[28] Q. Wang, G. Jiang, and X. Qin, "Multi-user preference Query based on user grouping," (in Chinese), *J. Chin. Comput. Syst.*, vol. 39, no. 8, pp. 1787–1793, Feb. 2018.
[29] S. Guo, L. Liu, and X. Qin, "Top-k Query for collaborative space keywords," (in Chinese), *J. Chin. Comput. Syst.*, vol. 39, no. 7, pp. 1532–1536, Feb. 2018.
[30] Y. Zhou, X. Qin, and X. Xie, "A group-based reverse k-rank Query algorithm," (in Chinese), *J. Chin. Comput. Syst.*, vol. 39, no. 10, pp. 2271–2278, Feb. 2018.
[31] W. Wang, W. Wang, and J. Wang, "Algorithm for finding the smallest circle containing all points of a point set," (in Chinese), *J. Softw.*, vol. 28, no. 09, pp. 1237–1240, Nov. 2000.
[32] L. Liu, L. Zhang, and J. Yu, "Line segment reverse k nearest neighbor Query based on Voronoi diagram in spatial database," (in Chinese), *J. Chin. Comput. Syst.*, vol. 38, no. 4, pp. 716–720, Feb. 2017.
[33] L. Zhu, W. Sun, and Y. Jing, "Voronoi graph-based k-nearest nearest neighbor Query method for road network," (in Chinese), *Chin. J. Comput. Res. Develop.*, vol. 48, no. 3, pp. 155–162, Mar. 2011.
[34] P. Pei, D. Zhang, and F. Guo, "A density-based clustering algorithm using adaptive parameter K-reverse nearest neighbor," in *Proc. IEEE Int. Conf. Power, Intell. Comput. Syst. (ICPICS)*, Shenyang, China, Jul. 2019, pp. 455–458.
[35] X. Chen, X. Cao, Z. Xu, Y. Zhang, S. Shang, and W. Zhang, "Accelerate MaxBRkNN search by kNN estimation," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Macao City, Macao, Apr. 2019, pp. 1730–1733.
[36] F. Guo, Y. Yuan, G. Wang, L. Chen, X. Lian, and Z. Wang, "Cohesive group nearest neighbor queries over road-social networks," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Macao City, Macao, Apr. 2019, pp. 434–445.
[37] B. J. Santoso, R. Mumpuni, H.-J. Hong, and D. S. Muhammad, "A grid-based approach in answering Top-k dominating queries on groups," in *Proc. 12th Int. Conf. Inf. Commun. Technol. Syst. (ICTS)*, Surabaya, Indonesia, Jul. 2019, pp. 343–348.
[38] S. Zhao and L. Xiong, "Group nearest compact POI set queries in road networks," in *Proc. 20th IEEE Int. Conf. Mobile Data Manage. (MDM)*, Hong Kong, Jun. 2019, pp. 106–111.
[39] H. Zhang, D. Sun, F. Ji, M. Xu, S. Gao, and Y. Xu, "Group visible nearest surrounder Query in obstacle space," in *Proc. IEEE Int. Conf. Comput. Sci. Educ. Inf. (CSEI)*, Kunming, China, Aug. 2019, pp. 345–350.

YONGSHAN LIU was born in Zhangjiakou, Hebei, China, in 1963. He received the M.S. degree in computer science and technology from Yanshan University, in 1989, and the Ph.D. degree in computer and applications from the Harbin University of Science and Technology, in 2006. Since 1994, he has been a Professor with the Department of Information Science and Engineering, Yanshan University. He also serves as the Tutor for Ph.D. student.



XIANG GONG was born in Taiyuan, Shanxi, China, in 1991. He received the M.S. degree in software engineering from Yanshan University, in 2017, where he is currently pursuing the Ph.D. degree in computer science and technology. His research interests include spatial database, information visualization, and visual analysis of obstacle environment.

DEHAN KONG was born in Dalian, Liaoning, China, in 1986. He received the Ph.D. degree in computer science and technology from Yanshan University, in 2017. He is currently teaching with the Department of Information Engineering, Hebei University of Environmental Engineering. His research fields are mainly spatial database and point cloud reconstruction and registration.

TIANBAO HAO was born in Handan, Hebei, China, in 1985. He received the M.S. degree in computer software and theory from Yanshan University, in 2011, where he is currently pursuing the Ph.D. degree in computer science and technology. His research field is mainly spatial database.

XIAOQI YAN was born in Xingtai, Hebei, China, in 1993. She received the M.S. degree in computer science and technology from Yanshan University, in 2016. Her research field is mainly spatial database.