International Journal of Control and Automation
Vol.7, No.9 (2014), pp.11-26
http://dx.doi.org/10.14257/ijca.2014.7.9.02

RkNN Query Algorithm Based on K-order Voronoi Diagram

Song Xiaoyu, Xu Jingke*, Yin Zhichao and Sun Huanliang

Department of Information and Control Engineering, Shenyang Jianzhu University, Shenyang 110168
sxy@sjzu.edu.cn

Abstract

Given a site set P and an object set R, Bichroamtic RkNN query of site q∈P finds objects that take q as k nearest neighbors, which can been used to evaluate the influence of q on objects. Existing methods execute RkNN query by pruning strategies based on spatial indexes. For any change of object datasets (like moving objects), these methods need to compute RkNN again. The paper proposes a new algorithm based on K-order Voronoi diagrams. For a fixed site q, its RkNN region does not change whatever object datasets are updated or not. So we only search objects in Voronoi region of q. In this paper, we first give some propositions that provide the searching bounds. Then, BRKVD algorithm is proposed for RkNN query based on R-Tree, which supports the frequent changes of datasets and k. The experimental results show that the proposed algorithm performs the existing algorithms on efficiency.

Keywords: *Spatial Databases; Bichroamtic RkNN; R-tree; K-order Voronoi Diagram*

1. Introduction

BRkNN (Bichroamtic Reverse k Nearest Neighbor) is to find the collection of all the query objects that take the specified site q ∈ P as kNN (k Nearest Neighbor) at a given site set P and query object set R, which can be used to evaluate the influence of the site q [1]. RkNN can be applied to knowledge discovery and decision support, facilities location, geographical information system and a variety of domains such as multimedia database.[2-3] RkNN query methods are mostly used R tree and improve tree indexing structure for processing, or data sets for some and pruning processing. For processing large amount of data objects, and the frequent updated data price is higher. Okabe *et al.*, [4] proposed on the basis of the Voronoi diagram query BRkNN, but it needs a lot of precomputation, the algorithm is only suitable when the order of Voronoi diagram and the k value are the same, and the update operation is low efficiency [5]. Based on Voronoi diagram k order, and the characteristics of space division area put forward three properties to solve the problem of dynamic change of k value. BRKVD algorithm is proposed based on the design to calculate the change under the frequent data set of double color RkNN queries.

RkNN query is complement and development of kNN query, according to the data set divided into monochrome RkNN query and double-color RkNN queries. F.Korn and S.Muthukrishnan[6] proposes the concept of RNN query, and gives a method to solve the RNN query. USES two R tree to query, insert, and delete operations. Yang and Lin [7] to improve the above algorithm, introduced the Rdnn tree, with a single tree RNN query and NN queries become possible. Stanoi [8] proposed an algorithm SAE, this algorithm is not calculated in the case of prediction, and SAE will around the query point range was divided into 6 sectors with the same size. Firstly, SAE finds RNN

ISSN: 2005-4297 IJCA

candidate within their respective sector; second, for each candidate has completed an independent NN query to determine the candidate isn't the final result. Maheshwari [9] the RNN main memory data structure is put forward by the query. For each point of the structure maintained its distance to the nearest neighbors.

Stanoi's method in high dimensional space, along with the increase of the dimension of the RNN candidate values shows exponential growth. Efficiency is greatly reduced. In order to solve this problem, Singh et al. [10] proposed by performing regular kNN query to find RkNN candidate values. But the trouble is that not always find all of the RkNN points. Tao [11] method of and Stanoi are similar, called the TPL. The TPL is divided into two stages, filtering and refining, given a query point q, a recursive algorithm to point q is not divided into space, until there is no candidate for the rest. When filtering step, in addition to some candidates identified as incorrect results.

In terms of RNN query based on Voronoi diagram, Li Song, Hao Zhongxiao[12] proposed in Euclidean plane Voronoi diagram to use Range-k validation method to find the result. As points focus point as the center, radius is the distance to the query point to generate a judge how to round to find RNN. The algorithm is suitable for single color only RkNN queries, when in double color RkNN query, there are a lot of unnecessary I/O operations. Maytham Safar [13] the Voronoi diagram of the proposed network to RNN enquiries in the space. Algorithm is suitable for the traffic network, etc., in the query object changed frequently when efficiency is very low.

Voronoi diagram is with the object of a given discrete set (such as a point set) distance to determine spatial decomposition, especially point set p = {p1, p2... pn} Voronoi diagram is defined as cell sets. Each cell here, V (PI) is a spatial area, PI by distance closer than other point in the distance p all the space of data points. Voronoi diagram can be applied to K-order Voronoi diagram (KVD). In KVD query object according to the definition of BRkNN in each region, the generators are the current area of the RkNN values. Use KVD performs BRkNN query includes the following steps:

(1) By positioning Voronoi region by the query site. Namely find any generator’s Voronoi region which contains the query site.

(2) According to BRkNN definition, the query objects in the Voronoi regions found are the results.

As shown in Figure 3, for searching the R3NN values of the site 3, first locate Voronoi regions of those generators including site 3, such as V (1, 3, 5), V (1, 3, 4), V (1, 3, 6), V (1, 2, 3). Therefore, all query objects in these four Voronoi regions are the result of R3NN (3).

However, the BRkNN query by using KVD has obvious disadvantages.

(1) High cost precomputation. KVD to precompute all KVD cells and all of the information associated with the cell such as generator, Voronoi edge, Voronoi points, etc.

(2)It dose not support the dynamic change of k values. KVD only can adapt with clearly k value RkNN query, KVD to adapt to the k value is not greater than RkNN query graph of order number. Therefore, this technology is not suitable for don't know k value in advance or may dynamically change k values.

(3)It is low efficiency of update operations. For each insert or delete operation, the cell had to recalculate.

As above shortcomings, this paper put forward the improvement, reduces the pre-calculated quantity, and increases the operation such as change the k value, add, delete. Greatly it increases the applicability of the Voronoi diagram.

This paper adopts the frame model of filtering-refining. In the filtering stage, call according to the k value required by the K-order Voronoi diagram, based on the query site, it is certain which Voronoi polygons compose the query range in the K-order Voronoi diagram, then all the query objects within the range is the candidate set. In the refining phase, and on the basis of k value, If the k value and the order of invoked Voronoi diagram are the same, then all the query objects in the candidate set is the result; If different, then the same query objects in all candidate sets are filtered as a result firstly, then the Range-k verification method to be used to verify the rest of the candidates, and the final query result is obtained. Experiments show that BRKVD query efficiency of the algorithm compared with the existing algorithms is improved greatly.

2. Problem Definition

Definition 1 RkNN(q,k,P): given a dataset P and a query point q, if kNN of p∈P contains the object q, p is the query results of RkNN(q). Specific definition form as follows:

$$RkNN\ (q, k, P) = \{p \in P \mid q \in kNN\ (p, k, P)\}$$

Monochromatic RkNN query is to obtain a given data set will be as a result of the k nearest neighbor queries point of the set of points.

Definition 2 BRkNN (q, k, P, R): for a given data set P and R, P is the query point q in the data set; R is the query results in the data set. For each of the r（r∈R）, if the results of kNN (r, k, P) include the q, then this r is an outcome of bichromatic RkNN (q, k, P, r). Specific definition form as follows:

$$BRkNN\ (q, k, P, R) = \{r \in R \mid q \in kNN\ (r, k, P)\}$$

BRkNN query is a query data set will be as a result of the k nearest neighbor queries point following the set.

Definition 3 Voronoi diagram [14]:

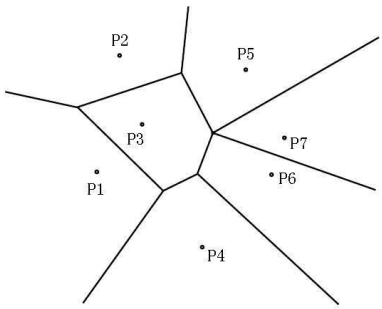


Figure 1. The 1-order Voronoi Diagram

A flat site set S = {p1,p2,…,pn}, n ≥ 2.There is, $V(p_i) = \{x \mid d(p_i, x) \le d(p_i, x)\}$, j=1, 2,…, n, i ≠ j. V(pi)is the Voronoi polygon of point pi, pi is the polygon's generator. Site set S of the Voronoi diagram $V(S) = \cup V(p_i),\ p_i \in S$. Figure 1 shows the seven points of the Voronoi diagram.

Definition 4 k-order Voronoi diagram:

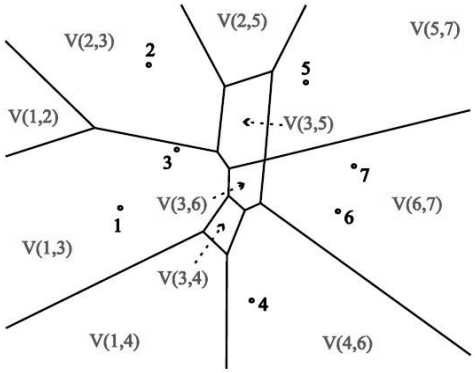


Figure 2. The 2-order Voronoi Diagram

A given site set E in plane S, S={p1, p2, …,pn}, n≥ 2.Pm is a subset of the S, Pm={p1, p2,…,pk}, k∈ {1, 2,…, n-1}.There are Vk(Pm)={x|d(x,pi)≤ d(x,pj)}, pi∈ Pm, pj∈ {S/Pm}, Vk(S) =∪ V (Pm). In which x stands for all the Pm of each site in the distance than other sites in to S (S/{Pm}) distance closer. Vitamin k (Pm) is called the P k order Voronoi polygon, is a region in E or is an empty set. Vitamin k (S) known as the S k order Voronoi diagram, is a subset of the set of all k site formation of order k Voronoi region of the set..

3. RkNN Query Algorithm based on K-order Voronoi Diagram (BRKVD)

This section analyzes the relationship between K-order Voronoi diagram and RkNN, and the related theories are proposed that the K-order Voronoi diagram is used to determine the RkNN query area. Then, the query algorithm is designed based on these theories.

3.1. The Properties of K-order Voronoi

OBJ to query object set; obji⊂ OBJ are query object; Obj(Pm) is any objects in the Voronoi polygon generated by the Pm; obji(Pm)⊂ obj(Pm) is an object in the Voronoi polygon generated by the Pm; Vk(Pm)(Pn) is an adjacent polygon of the Voronoi polygon generated by the Pm. Based on the definition of K-order Voronoi diagram and its generation features, this paper gives the following theorem:

Theorem 1. Let q be a query site, ∀ Vk(Pm), if the q⊂ Pm then obj(Pm) is the RkNN (q), and obj(Pm) is likely to be R(k-1)NN(q).

Proof: p is a query object, p⊂ OBJ. According to the order k, Voronoi polygon definition, query in a polygon object to the polygon recently, the distance from the generator, if p⊂ obj(Pm), then the Pm is kNN (p). Thus, p is RkNN (q) of a solution, including ∀ q⊂ Pm. In addition, because kNN(p)⊃ (k-1)NN(p), as a result, p can be R(k-1)NN(q), a solution of ∀ q ⊂ Pm.

As shown in Figure 3, the query object p is included in the polygon V_3 (2, 3, 5), get a 3NN (p) solution for {2, 3, 5}. Therefore, the point p is a result of R3NN (2), R3NN (3), R3NN (5). And 2NN (p) solution for {2, 3}, so p is a result of R2NN (2), R2NN (3), rather than a result of R2NN (5).

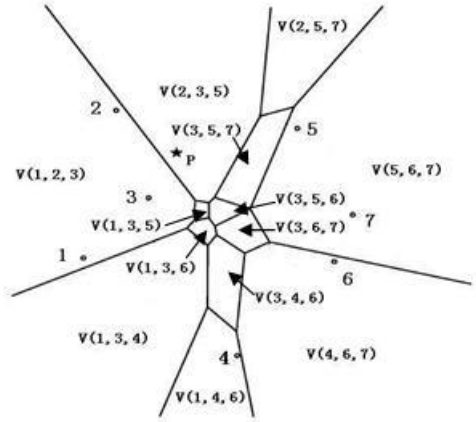


Figure 3. The 3-order Voronoi Diagram

Theorem 2. Let q be a query site, ∀ Vk(Pm), ∀ Vk(Pm)(Pn), if q∉ Pm and q∉ Pn then obj(Pm)∉ R(k+1)NN(q).

Proof: a∀ Vk(Pm), and q∉ Pm, ∀ Vk(Pm)(Pn)and q∉ Pn. Because Vk+1(PM) ⊂ Vk(Pm) ∪ Σ Vk(Pm)(Pn), among them PM is the generator of Vk+1(PM), and |PM|=|Pm|+1. Because of q∉ Pm and q∉ Pn, therefore, PM ∉ q, obj(PM) ∉ R(k+1)NN(q), by the nature of the obj(Pm) ⊂ obj(PM) as a result, obj(Pm) ∉ R(k+1)NN(q).

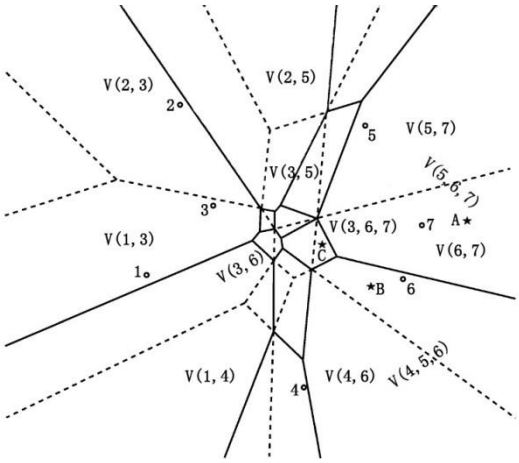


Figure 4. The 2-order and 3-order Mixed Voronoi Diagram

As shown in Figure 4, the points A, B and C are inside the polygon V 2 (6, 7) of 2-order Voronoi diagram, let the site 2 be a query site. The figure shows that the generators of the polygon V2 (6, 7) don't include the site 2, and the generators of the adjacent polygon V2 (5, 7), V2 (3, 5), V2 (3, 6) and V2 (4, 6) also don't include the site 2. So, the points A, B and C are not the results of R2NN (2). In 3-order Voronoi diagram, V2 (6, 7) is segregated by V3 (5, 6, 7), V3 (3, 6, 7) and the V3 (4, 6, 7), and becomes a part of these three polygons. The points A, B and C are contained by the three polygons respectively. Because that these three polygons' generators don't include the site 2, so the points A, B and C are also not the results of R3NN (2).

Theorem 3. R(k+1)NN(q) included $V_k(P_m) \cup \sum V_{k(P_m)}(P_n)$, including Pm⊃ q.

Proof: Let q be a query site, as the theorem 1, RkNN(q)=obj(Pm), including Pm⊃ q. By the RkNN definition, RkNN(q) ⊂ R(k+1)NN(q). So, obj(Pm) ⊂ R(k+1)NN(q). In addition, due to Vk+1(PM) ⊂ Vk(Pm) ∪ Σ Vk(Pm)(Pn), among them PM is the generator of Vk+1(PM), and |PM|=|Pm|+1, so PM⊂ Pm ∪ Σ Pn. And R(k+1)NN(q)=obj(PM), obj(PM) ⊂ Vk(Pm) ∪ Σ Vk(Pm)(Pn). And according to the nature of the two, if Pm⊃ q and q∉ Pn, so obj(Pm)∉ R(k+1)NN(q). The conclusion can be derived in reverse, R(k+1)NN(q)⊂ obj(Pm) ∪ Σ obj(Pn), including Pm⊃ q.

3.2. BRKVD Algorithm

First, the algorithm based on the k value generates the corresponding K-order Voronoi diagram. Because this algorithm each K-order Voronoi diagram can query the result of R(K−1) NN, RkNN and R(K+1)NN, so according to the need to generate the query 1, 3, 6, 9... order Voronoi diagrams. Compare the relationship between k and K, if k = K, so all query objects in the polygons with the generators include site q are the results. If k< K, then check the query objects in the polygons with the generators include site q, if the location of the query object is inside the k-3 order Voronoi polygons with the generators include site q (when k = 3, check the 1-order Voronoi polygons of q), then it is the result; If not, then it is determined by the Range-k validation method. Range-k validation method use the query object as the center of circle, the distance to the site q as radius makes circle, if the site number within a circle or on the circle ≤ k , then that is a result, not vice versa. If k> K, then according to the nature of the generator 3 that contains all the sites within the Voronoi polygon of q query object is the result; and these were examined adjacent polygons within a polygon query object to query object as the center, it makes a circle with the distance to the site of q distance as radius, if the site number in the circle or on the circle≤ k+1, then that is a result, not vice versa.

Algorithm 1：BRKVD()
INPUT：S: sites data set, Sobj: the query objects data set, q: query site, k: k value of RkNN
OUTPUT：RkNN result
1. K=0,R[ ]；
2. **If**(k%3==0)
3. K=k；
4. **Else if**(k%3==1)
5. K=k-1；
6. **Else**
7. K=k+1；
8. create root1,root3,root6,…,rootK index；
9. SearchObj(q,k)；
10. **Return** R；

In the above process, the establishment of the index R tree is depends on the k values to make sure. Specific building several indexes are calculated by k. Then based on the value of input q and k, it calls SearchObj (q, k) results.

Given below SearchObj (q, k) algorithm is described.

```
Algorithm 2:SearchObj(q,k)：
INPUT：q:query site,k:kvalue of RkNN
OUTPUT：RkNN result
1.  vector<CPoint>R1[ ],vector<CPoint>R2[ ],K；
2.  If(k%3==0)
3.      query rootk, result store in R1；
4.      Return R1；
5.  Else
6.      K=k-(k%3)；
7.      query rootK, result store in R1；
8.      K=k+(3-k%3)；
9.      query rootK ,result store in R2；
10. i=0；
11. While(i<R2.size())
12.     If R2[i]∉ R1
13.         ConfirmRkNN(R2[i])；
14.     i++；
15. Return R1；
```

In algorithm 2, the function ConfirmRkNN () is used to confirm whether the result in the candidate set R2 is the RkNN value. If it is then place it in R1. Function ConfirmRkNN () implementation is based on the Range-k validation method as the model. ConfirmRkNN () algorithm is given below.

```
Algorithm 3: ConfirmRkNN(pt):
1. int i,j;
2. vector<MinDist> min;
3. For(j=0；j<R2.size(); j++)
4.   For(i=0；i<S.size(); i++)
5.     calculate R2[j]to S[i]the distance store in min;
6.     min in distAttribute ascending order;
7.   For(i=0；i<min.size(); i++)
8.     If(pt==min[z].Gs)
9.         i++;
10.        If(i<=k)
11.            R2[j] store in R1;
```

3.3. The Index Structure of the Algorithm

BRkNN needs to access two classes of objects, namely site set and query object set. For this problem, some scholars use two R-tree indexes to solve the problem of query, such as finch algorithm. On site collection R-tree index get query range, and then find all the objects in the query range along with the query R-tree index. When the query objects changes, the method needs to be updated query object R - tree index. When the query object changes frequently, the query object R-tree will have to delete, add, divide operation frequently happened. Even to the reconstruction of the frequent query object R - tree index, impact the efficiency of query.

The algorithm on the basis of the traditional R - tree were modified, the query object into site set R - tree index. So you don't need to query object for R-tree indexes. For frequent

query object change, only need to site set R - tree index, modified or deleted. The modified R-tree structure is shown in Figure 5.

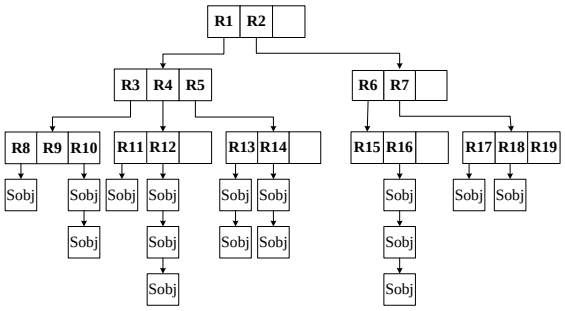


Figure 5. The Index of the BRKVD Algorithm

Seen from the Figure 5, under the R-tree the leaves of the tree nodes and links to a lot of Sobj type of node. This is the query object, the system to all queries within a Voronoi polygon objects in the Sobj type of node, in the form of a singly linked list chain into R - the leaves of the tree node. So, as long as find a leaf node, you can gain all the leaf nodes in the query object.

The node structure of the R-Tree is shown in Figure 6, in which the branch index structure consists of three parts, GP (the generator of Voronoi polygon), VGP(the Voronoi polygons generated by GP), Sobj (the query object included in the VGP). Each node of R-Tree can determine which contains much branch number according to the size of the branch index structure. In the branch of each leaf node, the meanings of GP, VGP and Sobj are unchanged. Each branch of the middle node, VGP and Sobj are empty, the GP is pointed out by the branch node in each branch of the GP is a collection of raw. Namely node branch GP attribute is among them, the generator node refers to all branches of the collection. When doing the query, it is facilitated to do the deep query according to the generating element Voronoi polygon.

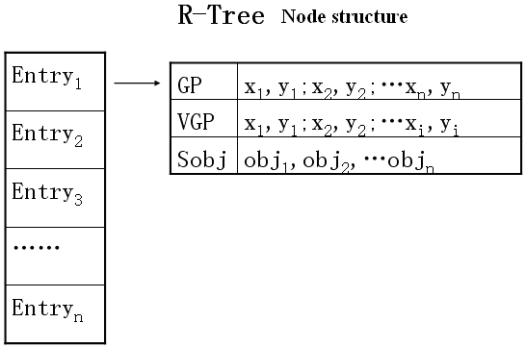


Figure 6. BRKVD Algorithm R-tree Index Table

When the program calls SearchObj (q, k), the function calls the index through the k values to determine the index R-Tree for the query. If k % 3 equals 0, then directly call the R-Tree rootk. In the R-Tree, with the generators contrast in-depth search, query site q belongs to the set S, so q is one of the polygons' generators. Since the root of the R-Tree nodes, called node in each branch of the GP, if GP contains q, is into the branch node. Otherwise, skip, always find a leaf node. If a branch of the leaf node in the GP attribute contains a query site q, then

investigate in this branch of the VGP properties. By VGP attribute to determine the Voronoi polygon shape in the branch, and then determine whether the query q contained in the Voronoi polygons. If so, and then all Sobj in the polygon is q values of RkNN, are deposited in the R. Otherwise, skip. Finally it returned R. If k % 3 does not equal zero, then through the k value to determine which of the two R-Trees is called. The query process of calling each R - Tree is the same as the above. Call the first R-Tree, save the query results in R1, the R1is a subset of the RkNN of q. call the second R-Tree, save the query results in R2, the R2 is candidate set of the RkNN of q. The R2 contains all R1. Therefore, delete the results of R1 inside R2, and then, judge the values in the rest of R2 by the method of circle. If it is the RkNN value of q, put it into the R1, otherwise it is skipped. Finally return to R1.

4. Performance Evaluation

This paper will present the design of algorithms compared with the best RkNN query algorithm Finch. Test two algorithms access the index during query processing node number (I / O times). Parameter settings include: the same site, different values of k queries; different sites of the same values of k queries; different sites of different values of k queries; same site and same k value under different order Voronoi diagram.

The implementation of Algorithm is adopted by VC++, the experiment for P4 2.4 GHz processor, for 1 gb of main memory, for the Windows xp operating system on microcomputer, and experimental testing data set using the real cultural landmarks and settlement data accumulation point part. Mainly there are four data sets: US2 billion (USA), the site of a data set number (519) query object number (1049). The site data set is characterized by uniform set points in some domestic things, and the query object set relatively concentrated in the west. MX (Mexico), the data set on the number of sites (1087) query object number (4293). The site set distribution data set is characterized by relatively concentrated in the west, while the query object set is split across spread across the country. US1 (USA), the site of a data set number (1504) query object number (3801). The site data set is characterized by uniform set points in some domestic things, and the query object set relatively concentrated in the west. CD (Congo), the data set on the number of sites (2099) query object number (4994). The site data set is characterized by uniform set points at home, north of relatively centralizing; The query object set relatively concentrated in the north.

In Figure 7 and Figure 10 testing a different data set a fixed point, under different k values of I/O number. In Figure 7, The I/O times of BRKVD query number is stable under 50 times. While the one of Finch query is in a rapid increase. Based on the characteristics of the US2 billion data set, and nine sites as a test point are chosen. Distribution in the nine sites around the query object, in turn, increases the density. Therefore, seen from the Figure 7, Finch number of I/O algorithm with the increase of site density of the query object around, the I/O number is rising fast. In BRKVD algorithm, the R1NN, R3NN3, R6NN, R9NN is through an index of R-Tree found at a time, so the I/O number obviously less than other RkNN query times. While the rest of the RkNN query Tree index lookup, although through two R - but as a result of the first R - Tree index RkNN results have identified part, so in the second R - Tree index, select the result of the number of candidates is less. So the number of I/O won't rapid increase. Can be seen from Figure 3-5 BRKVD algorithm in dealing with a query object Finch algorithm is obviously better than the relatively centralized data sets.

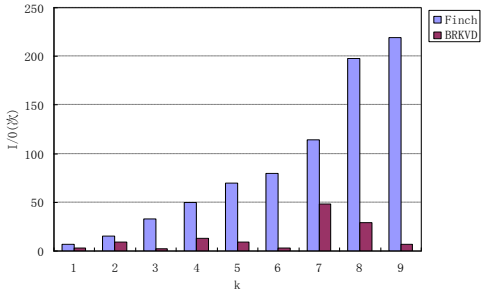


Figure 7. Fixed Query Site Testing Different k Values (US2)

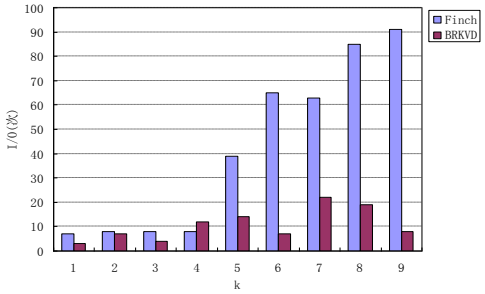


Figure 8. Fixed Query Site Testing Different k Values (MX)

In Figure 8, you can see, the number of I/O point 1 to point 4 two algorithms. This is due to the choice of MX query object is the average distributes of the data sets, and the site is relatively concentrated. Point 1 to point 4 distributions where the site is relatively concentrated, the density of the surrounding query object is relatively smaller to the site. As a result, the number of two kinds of algorithms is not much. And 5 to 9, the site is not much concentrated, and the relative density of the distribution around the query object is larger. According to the principle of Finch algorithm, it dramatic increase I/O number. And BRKVD algorithm of I/O number has increased, but the fluctuation changes without severe Finch algorithm. To sum up, can be seen from Figure 3 to 7 BRKVD algorithm in dealing with a query object average distribution site relatively concentrated distribution of data set, as a whole is better than Finch algorithm. The point 1 to point 6 of the site location is quite centralized place; its distribution around the query object is relatively more. So can see the I/O number of Finch algorithm though as the query object distribution density increased, but not dramatically. While BRKVD algorithm the number of I/O is still relatively stable at lower value. Site location of points 7 to 9 at the edge of the site distribution, the distribution density of the query object around a lot, so I see in the picture Finch algorithm grew rapidly in the I/O number. BRKVD algorithm by a rise in the number of I/O, but still relatively low. BRKVD algorithm, therefore, in dealing with a query object and site location data sets are relatively concentrated, Finch is obviously better than the algorithm.

In Figure 9, the locations of query objects and sites are relatively concentrated in the selected data set US1. The point 1 to point 6 of the site location is quite centralized place; its distribution around the query object is relatively more. So can see the I/O number of Finch algorithm though as the query object distribution density increased, but not dramatically. While BRKVD algorithm the number of I/O is still relatively stable at lower value. Site location of points 7 to 9 at the edge of the site distribution, the distribution density of the query object around a lot, so it can be seen in the picture Finch algorithm grew rapidly in the I/O number. BRKVD algorithm by a rise in the number of I/O, but still relatively low. So you can know, BRKVD algorithm in dealing with a query object and site location data sets are relatively concentrated, Finch is obviously better than the algorithm.

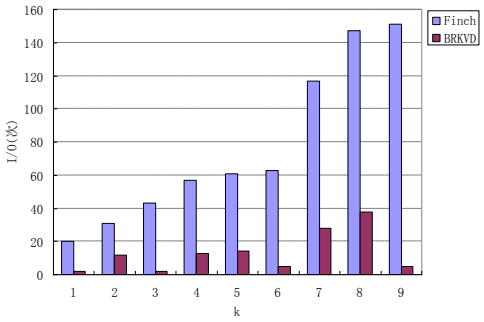


Figure 9. Fixed Query Site Testing Different k Values (US1)

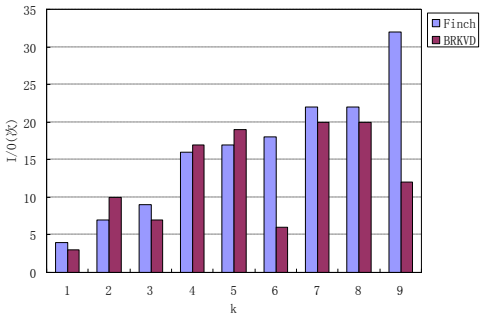


Figure 10. Fixed Query Site Testing Different k Values (CD)

In Figure 10, the choice of the data set CD is the average distribution site and relatively concentrated distribution and query object. Point 1 and point 2 and point 3 is a query object distribution is relatively concentrated distribution and site situation, it can be seen that both algorithms in this case the required I/O number, are maintain in low data. Points 4, 5, 7, 8 is relatively uniformly distributed query object distribution and site situation, in this case I/O number needed for the two algorithms is also similar, are maintained at higher values. Points 6 and 9 is the query object distribution is relatively concentrated sites distributed evenly, then Finch algorithm is the number of I/O required is higher, and BRKVD algorithm is the number of I/O required is relatively low. To sum up, in the treatment of the site average query object relatively concentrated distribution of data sets, BRKVD algorithm on the whole than Finch algorithm.

As shown in Figure 11 to 13, the number of I/O access were tested in the four data sets when k = 2, k = 5 and k = 8, but the query site fixed. Because of the characteristic of the four data sets each are not identical, so the data representing the characteristics of each data set is selected. In US2 , the site is in where the distribution of sites is relatively uniform, and the distribution density of the query objects around the site is relatively uniform. In MX, the site is in where the distribution of sites is relatively concentrated, and the distribution density of the query objects around the site is relatively uniform. In US1, the site is in where the distribution of sites is relatively concentrated, and the distribution density of the query objects around the site is relatively concentrated. In CD, the site is in where the distribution of sites is relatively uniform, and the distribution density of the query objects around the site is relatively concentrated.

From Figure 11, Figure 12 and Figure 13, when taking different k values to query using BRKVD algorithm in the data set US2, the number of I/O is always keep on the lower numerical, with no obvious change. And in the use of Finch query algorithm is several times the required I/O is higher, and change drastically. In data set MX, because the site is relatively concentrated as k = 2, two kinds of algorithm about the required number of I/O is used, and has low value. As k = 5 and k = 8, the numbers of I/O of two kinds of algorithm are different. BRKVD algorithm is still keeps on lower value, while Finch algorithm rapid rise, change is obvious. In data set US1, when k values are different, the two kinds of algorithm is very obvious on the number of I/O. This is because the candidate results of the Finch algorithm are too many, but BRKVD algorithm has identified the part of the query results as the index for the first time, so the number of candidate results is fewer in the second indexes. Number of I/O won't increase rapidly, so it always significantly bellows the Finch algorithm. In data set CD, two algorithms have no obvious difference and number of I/O number is low, this is because the data characteristics. Because the RkNN result is relatively small, therefore

the number of objects that the process of query needed to access is relatively small, so I/O number is low.

By tests under different data sets, fixed query site and different k values, we can see the I/O number of BRKVD algorithm is always in low value under different k value in the different data sets, and changes are not obvious. While the Finch algorithm set different changes in different data with data concentrating, the site and the distribution of the query object is different I/O times, there is an obvious change, and the numbers of I/O values are relatively high. From Figure 11, Figure 12 and Figure 13 shows that the effect of BRKVD algorithm is obviously better than the Finch algorithm for different data sets.

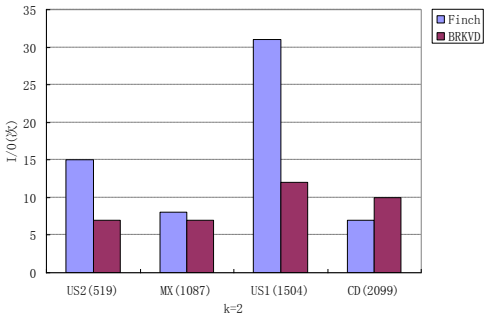


Figure 11. Different Data Set k = 2 Test

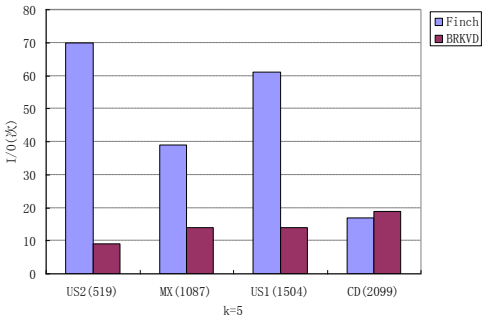


Figure 12. Different Data Set k = 5 Test

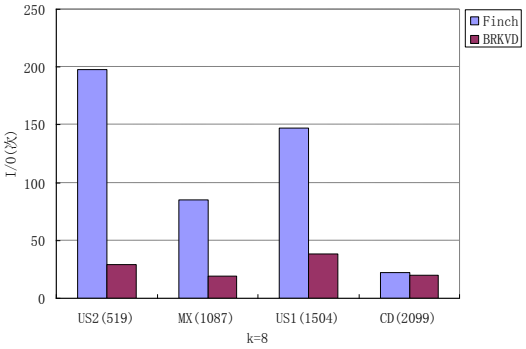


Figure 13. Different Data Set k = 8 Test

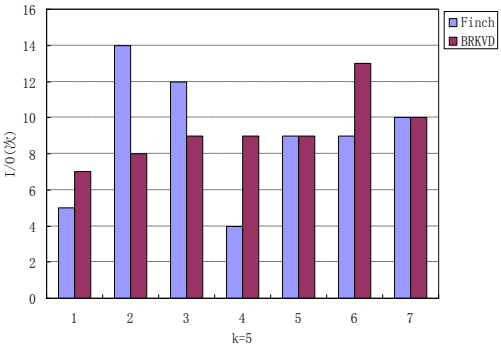


Figure 14. Fixed k Value Different Query Site Testing (US2)

As shown in Figure 14 to 17, in the test of four data sets samples, when k = 5, 7 points respectively, there take different test results. In Figure 14, the US2 data set was used. The points 1, 4, 6 three test points in the data center site distribution intensive place. Point 2 and point 3 is located in the data center sites distributed sparsely populated place. Points 5, and 7 distribution site is located in the data set is relatively uniform. The query object condition of uniform distribution can be seen from the figure, under BRKVD algorithm, the I/O times of the three points, 1, 4, 6 is larger than Finch algorithm, but is almost no difference in the numerical . Point 2 and point 3, BRKVD algorithm is slightly better than the Finch algorithm. In point 5 and 7 cases, the two algorithms are at the same number of I/O. This we can find that, when k = 5, there is not much difference between the effects of the two algorithms for US2 billion data set.

In Figure 15, the data set MX is used. The point 1 and point 7 is located in the site distribution uniform query the location of the object distribution. Point 2, point 4 is located in the site distribution query the location of the object distribution uniform. Points 3, 5, and 6 located on the site distribution query the location of the object distribution. It can be seen from the diagram, in the point 1 and point 7, BRKVD number query algorithm with distributed around the test point query increases with increasing the number of objects, but the increasing effect is not obvious and the number of I/O is a little low. And when increasing the query object number of the Finch algorithm, the number of I/O has been rapidly increasing. In point 2, point 4, because the query object position, BRKVD algorithm in indexing query for the first time identified results is larger, the second index the query to determine the number of candidates are greatly influenced by the other sites, the number is less. So the number of I/O is less over time. And Finch algorithm is affected by the query object location of the I/O number is more obviously. At point 3, point 5, 6 of the affection, because the affection by other sites of the candidate sites is rather small when BRKVD algorithm does the query in the second index, and the number is larger, so the I/O number relative to the Finch algorithm significantly more number of I/O, too. Integrated as a whole, BRKVD algorithm in MX query efficiency centralized data processing different characteristics are better than Finch algorithm.

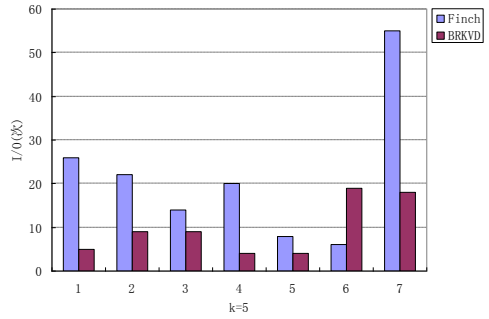


Figure 15. Fixed k Value Different Query Site Testing(MX)

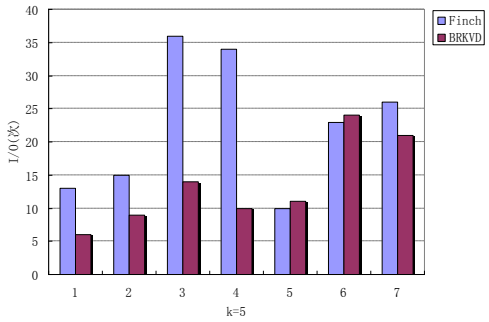


Figure 16. Fixed k Value Different Query Site Testing (US1)

In Figure 16, the data set US2 is used. The point 1, point 2 and point 5 is located in the site and the query object distribution is concentrated. Point 3 and point 4 is located at the site and query objects are uniformly distributed. Points 6 and 7 is located at the site where centralized distribution uniform query object distribution. Can be seen from the graph at point 1 and point 2, point 5, two kinds of algorithm is the number of I/O are maintained on the lower numerical, and BRKVD algorithm is better than the Finch algorithm.

In the point 3 and point 4, due to the query object distribution, Since Finch algorithm has a larger number of candidate, and the number of I/O is increasing rapidly. And to BRKVD algorithm, because the influence of the site is larger, so the candidate numbers are smaller. Therefore it has fewer I/O number and lower value. In points 6, 7, because the site is affected, so BRKVD algorithm is the number of candidates is more I/O number is bigger also. Seen from the figure with Finch algorithm, numerical value is higher. Integrated as a whole, figure 3-15 shows results BRKVD algorithm are better than the effect of Finch algorithm.

In Figure 17, the CD data set is used. Points 1, 3 and 7 points located on the site and the query object are focused. 6 points, 2 points located on the site where centralized distribution uniform query object distribution. A point 4 and 5 locates in site distribution uniform query

object distribution. As can be seen from the diagram points 1, 3 and 7, BRKVD algorithm due to the site and the distribution of the query object I/O number have obvious ups and downs. But numerical maintain in low position and number of I/O with Finch algorithms were similar, and slightly better than the latter. At point 2 and point 6, see the Finch algorithm due to the distribution of the query object is a rapidly increasing number of I/O. While BRKVD algorithm is lower I/O times better than the former. In points 4 and 5, BRKVD algorithm is affected by the distribution of the site and the query object is not big, I/O number is low. Finch algorithm are greatly influenced by the query object distribution, number of I/O is BRKVD algorithm has significantly increased. Overall, Figure 17 shows BRKVD algorithm is better than the Finch algorithm.

In Figure 18, respectively in different data sets, according to the data distribution is different respectively took 50 data with their own characteristics. These data to calculate the average number of I/O for collection accordingly. According to the chart shows, BRKVD algorithm in capacity between different samples in a fixed k value of the average visits are a relatively smooth maintenance at a relatively low number. Average visits volatile Finch algorithm. With site visits, of course, the fluctuation is mainly related to the distribution and query object sets. As can be seen from the diagram BRKVD number average access algorithm is relatively lower than the Finch algorithm.

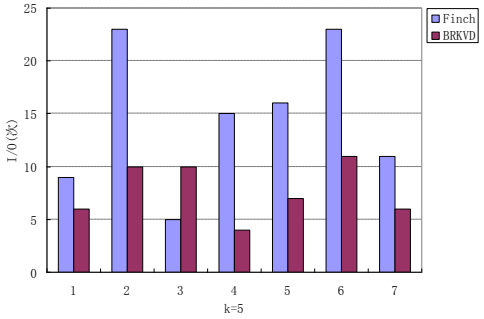


Figure 17. Fixed k Value Different Query Site Testing (CD)

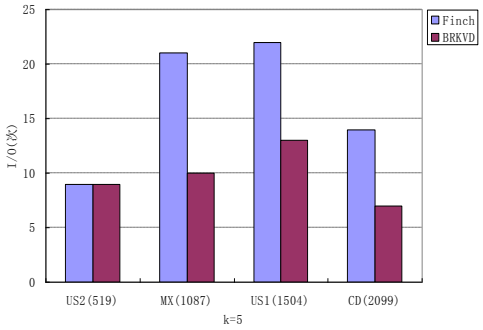


Figure 18. Different Data Sets the Average I/O Test

5. Conclusions

The bichromatic RkNN query algorithm is designed based on K-order Voronoi diagram. In order to improve query efficiency, using the properties of the K-order Voronoi diagram, the query object is filtered, and using the Voronoi diagram to achieve R(K−1) NN, RkNN and R(K+1)NN query. Experiment shows that the proposed bichromatic RkNN query algorithm based on K-order Voronoi diagram is suitable for different sites and query objects, and it can make rapid accurate RkNN queries.

Acknowledgments

This work is supported in part by National Natural Science Foundation of China (61070024).

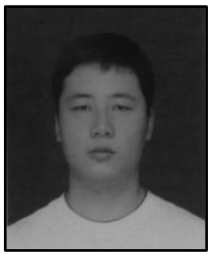
References

[1] W. Wu, F. Yang, C. Y. Chan and K. L. Tan, "FINCH: Evaluating Reverse k-Nearest-Neighbor Queries on Location Data", VLDB, Auckland, New Zealand, **(2008)**.
[2] K. Mitra, D. Bhattacharyya and T. H. Kim, "Urban Computing and Information Management System Using Mobile Phones in Wireless Sensor Network", International Journal of Control and Automation, vol. 3, no. 1, **(2010)**, pp. 17-25.
[3] J. Lee, M. Lee, S. H. Lee, S. G. Oh, B. H. Kim, S. H. Nam and J. S. Jang, "Development of Computerized Facility Maintenance Management System Based on Reliability Centered Maintenance and Automated Data Gathering", International Journal of Control and Automation, vol. 6, no. 1, **(2013)**, pp. 1-12.
[4] A. Okabe, B. Boots and K. Sugihara, "Spatial tessellations: concepts and applications of Voronoi diagrams", Wiley. com, **(2009)**.
[5] S. Nutanong, R. Zhang, E. Tanin and L. Kulik, "The V*Diagram: A QueryDependent Approach to Moving KNN Queries", VLDB'08, AUCKLAND, New Zealand, **(2008)**.
[6] F. Korn and S. Muthukrishnan, "Influence Sets Based on Reverse Nearest Neighbor Queries", SIGMOD, **(2000)**; Dallas, Texas, USA.
[7] C. Yang and K. I. Lin, "An index structure for efficient reverse nearest neighbor queries", ICDE, **(2001)**; Heidelberg, Germany.
[8] I. Stanoi, D. Agrawal and A. El Abbadi, "Reverse Nearest Neighbor Queries for Dynamic Databases", ACM/SIGMOD Int. Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD), Madison, Wisconsin, USA, **(2000)**.
[9] A. Vahrenhold Maheshwari and J. Zeh, "On reverse nearest neighbor queries", Proceedings of the Canadian Conference on Computational Geometry, Tokyo, Japan, **(2002)**.
[10] A. Singh, H. Ferhatosmanoglu and A. Tosun, "High dimensional reverse nearest neighbor queries", Proceedings of ACM CIKM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, **(2003)**.
[11] Y. Tao, D. Papadias and X. Lian, "Reverse kNN search in arbitrary dimensionality", VLDB, Toronto, Canada, **(2004)**.
[12] S. Li and Z. X. Hao, "Reverse nearest neighbor queries using Voronoi diagrams", Journal of Harbin engineering University, vol. 29, no. 3, **(2008)**, pp. 261-265.
[13] M. Safar , D. Ibrahimi and D. Taniar, "Voronoi-based reverse nearest neighbor query processing on spatial networks", Multimedia Systems, no. 15, **(2009)**, pp. 295–308.
[14] R. C. W. Wong, M. T. Ozsu, P. S. Yu, A. W. C. Fu and L. Liu, "Efficient Method for Maximizing Bichromatic Reverse Nearest Neighbor", VLDB, Lyon, France, **(2009)**.

Authors

SONG Xiaoyu, Born in 1963, Ph.D., professor, member of China Computer Federation. His research interests include computational intelligence, pattern recognition and data mining, etc.

XU Jingke, Born in 1976, master, associate professor, member of China Computer Federation. His research interest is Spatial Database.



YIN Zhichao, Born in 1979, master, His research interest is Data Mining.

SUN Huanliang, Born in 1969, Ph.D., professor, member of China Computer Federation. His research interests include Spatial Database and Data Mining.