Spatial Database Indexing Optimization Method Based on NVD-GkNN Algorithm

Huili Xia

(Corresponding author: Huili Xia)

College of Big Data and Artificial Intelligence, Zhengzhou University of Economics and Business
Zhengzhou 451191, China
Email: xiahuili80@163.com

(Received July 11, 2023; Revised and Accepted May 20, 2024; First Online Aug. 17, 2024)

Abstract

Spatial data query methods are essential in spatial databases and intelligent transportation. The nearest neighbor query is one of the most widely used methods for spatial data query. However, the nearest neighbor query and its variants suffer from low query efficiency, inefficient processing, and high computational cost. Therefore, to address the shortcomings of the nearest neighbor query and its variants in the obstacle space and road network environments, the study proposes to add Voronoi diagrams and grid Voronoi diagrams to these methods, using the filtering function of Voronoi diagrams to decrease the quantity of queried points and enhance the query efficiency. The results show that the Network Voronoi Diagram-group k Nearest Neighbor (NVD-GkNN) algorithm in the road network environment combined with the grid Voronoi diagram has a CPU running time as long as the Timing Algorithm (TA) and Incremental Euclidean Restriction (IER) CPU running time. The Voronoi graph-based nearest neighbor and its variants are designed to improve the query efficiency, which also provides a way to improve the efficiency of other spatial data query methods. The research aims to improve the query efficiency of spatial databases, maintain data security of databases, and assist in network security.

Keywords: Database; Indexing; kNN; Road Network Environment; Voronoi Diagram

1 Introduction

With the technology boost in China, the fields of geographic information systems and intelligent transportation systems are also developing rapidly. Spatial data query methods play an important role in these fields [3]. When dealing with complex spatial data, traditional data query methods are not effective, so new data query methods need to be studied [13]. Nearest neighbor query is one of the most widely used techniques for spatial data query, and many variants of it have been developed, like k-nearest neighbor query and reverse nearest neighbor query [19]. However, most of these query methods suffer from low query efficiency, inefficient processing and high computational cost. In addition, nearest neighbor query methods can be divided into two types according to the data environment, one is based on the Euclidean space and the other is based on the road network environment, and different query methods are applicable to different data environments [14]. Voronoi diagrams are mainly used to partition the space, and their filtering function can decrease queried points' quantity and enhance the data query efficiency, which is effective in spatial query and multimedia database [1]. The Voronoi diagram can be used to decrease query points' quantity and enhance the data querying. In addition, Voronoi diagrams can also reduce development and maintenance costs. Based on the above background, the research innovatively combines Voronoi diagrams and grid Voronoi diagrams with nearest neighbor query and its variant query, and then applies these algorithms to data query in obstacle space and road network environments respectively. The Network Voronoi Diagram-group k Nearest Neighbor (NVD-GkNN) algorithm improves the query efficiency of group k-nearest neighbor in road network environment, and the reverse nearest neighbor in road network environment combined with grid Voronoi diagrams is also enhanced. The Network Voronoi Diagram-Reverse Nearest Neighbor (NVD-RNN) algorithm performs better in certain situations. The k Nearest Neighbors-Obstacle (kNN-Obs) algorithm combined with Voronoi diagrams in obstacle space can improve its own query efficiency under certain circumstances. The study aims to enhance the query efficiency and diminish the query cost of nearest neighbor queries and their variants. The research aims to improve the query efficiency of spatial databases, maintain data security of databases, and assist in network security.

2 Related Works

The querying of data in spatial databases is a popular research area that has been explored by many researchers. Eiter and other researchers used local dynamic maps to set up spatial flow query responses in collaborative intelligent transportation systems, which could complete data flow queries oriented towards semantic concepts and spatial relationships. In addition, the semantic enhancement method of this study was conducted in ontology based query responses. The experimental results showed that the method designed by the research institute is feasible and can improve the efficiency of queries [6]. Sarode and Reshmi proposed a group search optimization algorithm based on neural networks to study data aggregation models, and developed a query based data aggregation model based on this algorithm. The query ranking in this study was determined based on latency and throughput. The research results indicated that the data aggregation model could improve throughput while reducing latency, and its performance was significantly better than traditional data aggregation models [16]. Jakob and Guthe proposed to use GPUs for exact domain problems in point clouds in order to optimise the query performance of kNNs on point clouds. The results showed that this approach gave real-time capabilities for large queries on very large point clouds, and the speed of the queries was greatly improved [9]. Chen et al. proposed a HorseIR-based approach to better querying of databases, combining database queries with compiled code based on dynamic arrays. Experimental results proved that this method was usable for database queries [4]. Scholars such as Shi and Yang proposed a spectral clustering based method to classify the climate in China. This method first analyzed the correlation between meteorological variables, and then constructed a similarity matrix graph based on k-nearest neighbor and sparse subspace representations. Finally, the study used a method of determining the number of clusters to conduct sensitivity analysis on different parameters. The research results showed that this classification method has high accuracy [18]. Experts such as Jang et al. proposed an input variable initialization method based on the k-nearest neighbor method to achieve the optimal input of neural networks. This method required finding the input that made the output very close to the target output and processing it as the initial input variable. The experimental results indicated that the method designed by the research institute is significantly superior to random initialization [10].

Shi and other researchers designed a flutter identification method based on enhanced k-nearest neighbors to identify flutter. This method compared the information of different sensors based on a large number of experiments, and extracted the features. Finally, the enhanced k-nearest neighbor method was used to identify chatter under different cutting conditions. The experimental results showed that the method designed by the research institute could effectively identify flutter with high accuracy [17]. Jg et al. proposed a generalised fragment allocation strategy to avoid the current database fragment allocation strategy that was prone to low-quality allocation plans, which implemented multiple candidate allocation schemes based on cost through PostgreSQL improved genetic algorithms evaluation. The results of the study showed that the performance of this strategy was improved by a factor of 2-4 with good robustness and scalability [11]. Ding and other experts proposed an R-KWS method in view of the evaluation of combinatorial candidate networks in order to improve the efficiency of existing database query methods, which could share the overlapping parts between candidate networks. The experiment indicated that this method can enhance the query efficiency without losing the quality of the query results [5]. Gulzar et al. proposed an optimised and incomplete framework for Skyline that simplified the Skyline process for databases with missing data, in order to avoid the situation of Skyline query methods' error when there was missing data in the database. The results showed the superiority of the framework and the number of control tests required to retrieve the skyline [8]. In order to improve the performance of distance metric learning, scholars such as Ruan et al. designed a nearest neighbor search model for distance metric learning. This model could construct metric optimization constraints by searching for the optimal nearest neighbor number. In addition, the study also designed a k-free nearest neighbor model based on a support vector machine solver. The experimental results showed that the performance of the nearest neighbor search model designed by the research institute for distance metric learning was significantly superior to existing baseline methods [15].

In summary, there have been many studies on spatial database data query methods, but most of them suffered from low query efficiency, less efficient processing and high computational cost. Based on these problems, the study innovatively combines Voronoi diagrams and nearest neighbor queries, aiming to make up for the shortcomings of existing methods in different data environments.

3 Spatial Database Optimization Based on NVD-GkNN Algorithm

3.1 Optimal Design of Spatial Databases Under Different kNN Algorithms

The Nearest Neighbor (NN) method plays an important role in spatial data querying [20]. The traditional k-Nearest Neighbor (kNN) algorithm uses the second-order Ming's distance as a measure of similarity between samples. By noting the observations of samples i and samples j as $x_i = (x_{i1}, x_{i2}, \cdots, x_{ip})$ and $x_j = (x_{j1}, x_{j2}, \cdots, x_{jp})$, where each sample has a different variable, the Ming's

distance is described in Equation (1).

$$d_{ij}(q) = (\sum_{k=1}^{p} |x_{ik} - x_{jk}|^q)^{1/q} \tag{1}$$

In Equation (1), d_{ij} serves as the range between samples i and samples j, q represents the order, k denotes the first k variable for each sample, and k takes values in the range $[1, p]$. Depending on the value of q, the Ming's distance can be classified as Manhattan distance, Euclidean distance and Chebyshev distance [12]. When the value of q is 1, $1/q$ is 1. The first-order Ming's distance can be called the Manhattan distance, as shown in Equation (2).

$$d_{ij}(1) = \sum_{k=1}^{p} |x_{ik} - x_{jk}| \tag{2}$$

When the value of q is 2, $1/q$ is $1/2$ and the second order Minkowski distance can be called the Euclidean distance, as in Equation (3).

$$d_{ij}(2) = \sqrt{\sum_{k=1}^{p} (x_{ik} - x_{jk})^2} \tag{3}$$

The change in the Ming's distance is greater when the q value $\to \infty$. At this point the Ming's distance has been transformed into the Chebyshev distance, as in Equation (4).

$$\begin{aligned} d_{ij}(\infty) &= \lim_{p\to\infty} (\sum_{k=1}^{p} |x_{ik} - x_{jk}|^q)^{1/q} \\ &= \max_{1\le k\le p} |x_{ik} - x_{jk}| \end{aligned} \tag{4}$$

The traditional kNN has relatively good query performance, but when the number of samples is relatively large, it suffers from computational complexity and memory consumption problems. Based on these problems, the study combines a kNN with optimised weights on the basis of the third-order Ming's distance and applies it to the filling of missing values. Equation (5) shows the details.

$$\hat{x}_{ik}^m = \sum_{j=1}^{K} \frac{D_{ij}^{-1}}{\sum_{V=1}^{K} D_{iv}^{-1}} x_{jk} \tag{5}$$

In Equation (5), $x_i = (x_i^c, x_i^m)$ is the vector to be filled, x_i^c represents the complete part of the vector, x_i^m represents the missing part of the vector, D_{ij} serves as the range in the vector i and the vector j, then $\hat{x}_{ik}^m$ is the filled value of x_i^m. When the samples are close to each other, the fill value obtained by the kNN algorithm satisfying and the algorithm needs to be improved, as in Equation (6).

$$x_{ik}^m = \frac{K}{(K-1)^2} \sum_{j=1}^{K} (1 - \frac{D_{ij}}{\sum_{V=1}^{K} D_{ij}} x_{jk} \tag{6}$$

In Equation (6), x_{ik}^m denotes the new filled value, D_{ij} is the third-order Ming's distance formula.

$$D_{ij} = d_{ij}(3) = (\sum_{k=1}^{p} |x_{ik} - x_{jk}|^3)^{1/3}$$

K denotes the number of similar samples. The Euclidean distance used in the traditional kNN algorithm does not work well in mixed attribute data, compared to grey correlation analysis which is more appropriate. It replaces the Euclidean distance with a grey distance. In addition, to deal with missing values in mixed attributes, a grey weighted kNN filling method combining iterative kNN and grey distances can be used. Before carrying out grey correlation analysis on the data, the data needs to be pre-processed with datacentering, as in Equation (7).

$$\begin{cases} \hat{x_{ij}} = x_{ij} - \bar{x_i} \\ \hat{x_{ij}} = x_{ij} - \bar{x_j} \end{cases} \tag{7}$$

In Equation (7), $\hat{x_{ij}} = x_{ij} - \bar{x_i}$ is sample-centered, $\hat{x_{ij}} = x_{ij} - \bar{x_j}$ is attribute-centered, i takes the values of $[1, p]$ and j takes the values of $[1, p]$. x_{ij} and $\hat{x_{ij}}$ each represent the value of the ith sample in the jth attribute before and after standardisation, $\bar{x_i}$ represents the mean of all attributes in the ith sample, and $\bar{x_j}$ represents the mean of all samples within the jth attribute.

$$\begin{cases} \hat{x_{ij}} = \frac{x_{ij} - \bar{x_i}}{\sqrt{\sum_{j=1}^{q} (x_{ij} - \bar{x_i})^2}} \\ \hat{x_{ij}} = \frac{x_{ij} - \bar{x_j}}{\sqrt{\sum_{i=1}^{q} (x_{ij} - \bar{x_j})^2}} \end{cases} \tag{8}$$

Equation (8) shows the outlier normalisation, which is done by dividing the data normalised by the outlier after datacentering. Equation (9) is data regularisation, which is normalised by the standard deviation (SD).

$$\begin{cases} \hat{x_{ij}} = \frac{x_{ij} - \bar{x_i}}{\sqrt{\sum_{j=1}^{q} (x_{ij} - \bar{x_i})^2 / (p-1)}} \\ \hat{x_{ij}} = \frac{x_{ij} - \bar{x_j}}{\sqrt{\sum_{i=1}^{q} (x_{ij} - \bar{x_j})^2 / (q-1)}} \end{cases} \tag{9}$$

In addition, the data are pre-processed by means of Z-score normalisation, which starts with the mean and SD of the attributes, as shown in Equation (10).

$$\delta_m = \sqrt{(\sum_{i=1}^{n} (x_{im} - \bar{m})^2) / (n-1)} \tag{10}$$

In Equation (10), δ_m serves as the SD of the attribute m, n serves as the data points' quantity in the data set, x_{im} represents the value of the attribute m for the ith sample, and $\bar{m}$ represents the mean of the attribute m. The Min-Max normalisation method uses the linear variation of the original data as the entry point for data processing, one of which is the upper bound validity measure, as in Equation (11).

$$\acute{x_p}(j) = \frac{x_p(j) - \min_\forall x_i(j)}{\max_\forall x_i(j) - \min_\forall x_i(j)} \tag{11}$$

In Equation (11), $\max_\forall x_i(j)$ serves as the maximum value in the data, $\min_\forall x_i(j)$ represents the minimum value, and $\acute{x_p}(j)$ serves as the value of the sample p after pre-processing on the attribute j. When the data attribute is extremely small, a lower bound validity measure

is used, as in Equation (12).

$$x_p^{'}(j) = \frac{\max_{\forall} x_i(j) - x_p(j)}{\max_{\forall} x_i(j) - \min_{\forall} x_i(j)} \tag{12}$$

Equation (12) is called the lower bound validity measure, where very small data are also called cost-based indicators. In addition, the Min-Max standardisation method involves a moderate validity measure, as in Equation (13).

$$x_p^{'}(j) = \frac{|x_p(j) - x_{specified}|}{\max_{\forall} x_i(j) - \min_{\forall} x_i(j)} \tag{13}$$

In Equation (13), $x_{specified}$ is a pre-set value, as the upper and lower bound validity measures have relatively similar results, the moderate validity measure can also be used when selecting the data processing method. Voronoi diagrams are mainly used for spatial partitioning and the Voronoi diagram is constructed as shown in Figure 1 [7].

There are multiple methods for constructing Voronoi diagrams and one of them was used for the study. As shown in Figure 1(a), it is a part of a Voronoi diagram that must be partitioned from the region where p_k is located. The region is divided by the vertical bisector of p_{k+1} and p_j, and forms two parts. The boundary between the vertical bisector and the area where p_k is located intersects at two points and one of these points needs to replace the other. This behaviour is repeated until the dashed polygon shown in Figure 1(a) is formed. Figure 1(b) forms the Voronoi polygon of p_{k+1} by deleting the polygon vertices and the middle edge. A grid Voronoi diagram is also a type of Voronoi, an example of which is shown in Figure 2 [2].

Figure 2(a) shows the initial road network diagram with the generation points from p_1 to p_3 and the intersections of the road network from p_4 to p_{16}. It is connected via L. Figure 2(b) is a grid Voronoi diagram with each line having a Voronoi link set corresponding to the generation point. The links may be in the generating point V_{link} only, or in different V_{link}. When a link is in a different V_{link}, any one of the lines that does not pass through L may be connected to it.

3.2 Design of the GkNN Spatial Database Optimization Method Based on Voronoi Diagrams

In different data environments, nearest neighbor query methods can be separated into two kinds, the first is based on the Euclidean space and the other is based on the road network environment. For obstacle queries in Euclidean space, the study used the kNN query method k Nearest Neighbors-Obstacle (kNN-Obs) based on Voronoi diagram with the algorithm k NN-Obs $q, k, P, O)$, where q denotes the query point, P is the dataset, O represents the set of obstacles and k is the value of the kNN query. This method involves two aspects, the filtering process and the refinement process. The filtering process mainly generates a kNN candidate set, while the refining process refines the objects in it through Voronoi diagrams, and prunes the objects that do not meet the criteria. Before using the kNN algorithm, the originality of the data needs to be maintained, so the randomly distributed data is processed, as shown in Figure 3.

Figure 3(a) is a spatial data Euclidean diagram showing the distribution of the data in the spatial dimension. Figure 3(b) is a data redistribution diagram, showing the differences and characteristics of the data before and after the distribution. After redistribution of the data, the original local Voronoi diagram is then generated from this data, as shown in Figure 4(a). However, the original local Voronoi diagram is not particularly accurate, as it does not take into account the relationship between the server boundary points, so this Voronoi diagram needs to be optimised and the result is shown in Figure 4(b).

Both the multi-core technique and the optimised scanline algorithm are applied to the generation of the original local Voronoi diagram. In Figure 4(a), L1 to L11 are the original generation points and S1 to S3 are the servers. The Voronoi cells are the polygons where the original generation points are located and the boundary points of the S1 and S2 servers are L2 and L3. Optimisation of the original local Voronoi diagram requires finding the clustering centers above each server. Then the closest location data to the clustering centers needs to be find, which is used as the server's clustering result, and then it is finally optimised. For the nearest collar variant query in the road network environment, the study uses two types of methods. The first one is the NVD-RNN based on the grid Voronoi diagram, the algorithm is NVD-RNN $q, S, k)$, where q is the query point, S is the dataset and k is the value of the RNN query. The steps in this method involve filtering and refining, which results in a candidate set in which all possible points are stored as results. Refinement involves computing the points in the candidate set and finding the final result, then filtering and refinement are used recursively for each query. The Voronoi diagram of the grid involved in this method is shown in Figure 5.

In Figure 5, the grid graph NVP (p_2) has the query point q, so its 1-RNN is p_2. The resulting candidate set after filtering is $\{p_7, p_8, p_9\}$, and then the refinement process calculates the nearest distance between the query points q and $p + 7$, p_8, and p_9. The filtering process involves two while loops, the first of which terminates after all the data in the dataset are looped through once, while the second while loop requires all the data in the candidate set to be looped through once before aborting. The refinement process involves only one while loop and the data in it is finite. The loop terminates when this data has been looped through once. RNN is a variant of nearest neighbor, and its query is shown in Figure **??**(a).

In Figure **??**(a), NN(c) denotes the result obtained by NN query for point c and RNN(c) denotes the result obtained by RNN query for point c. In addition, the data used in both query methods are from a database or collection of data identified in advance. In Figure **??**(b), the

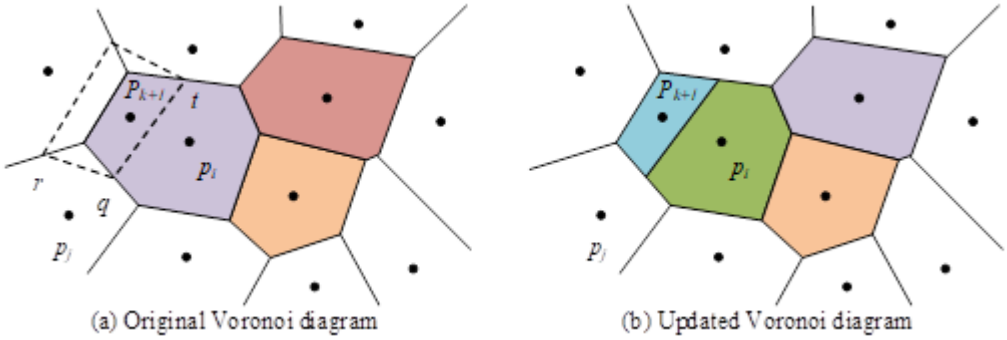


Figure 1: The process of updating the Voronoi diagram

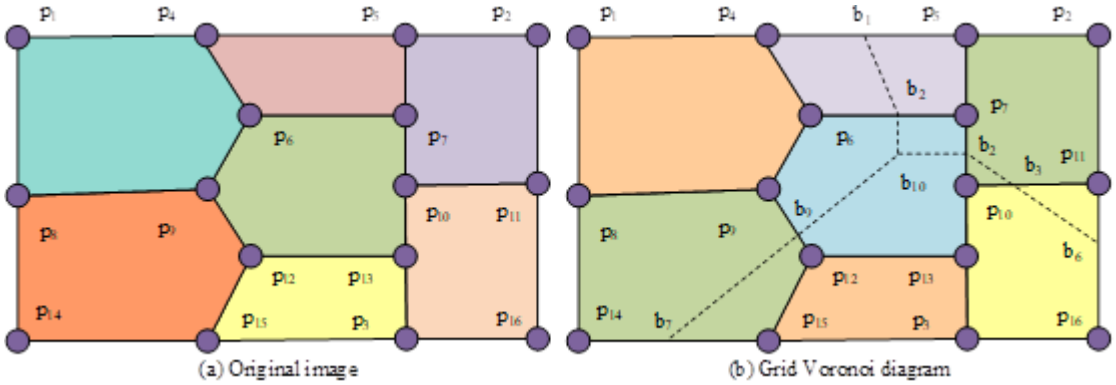


Figure 2: Instance of original and network Voronoi diagrams

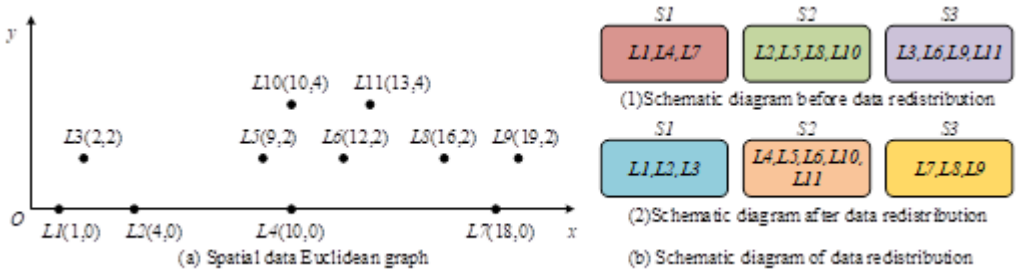


Figure 3: Spatial data Euclidean diagram and schematic diagram before and after data redistribution

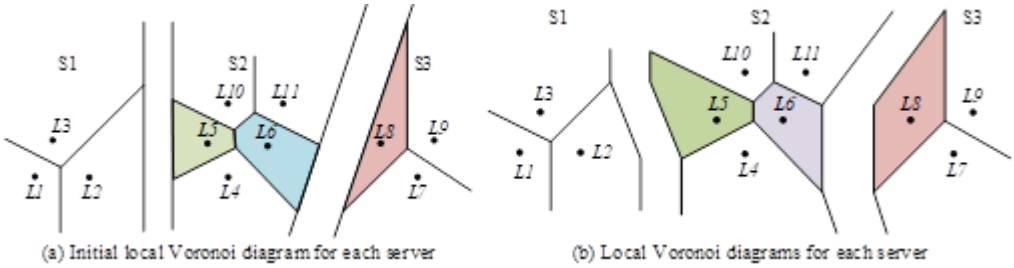


Figure 4: Original and improved local Voronoi diagrams for each server

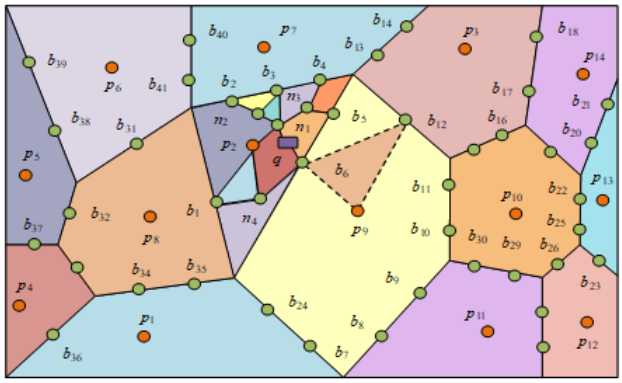


Figure 5: Instance of network Voronoi diagram

RNN query mainly involves a location query with overlapping parts among each circle. In the road network environment, the second nearest neighbor variant query is NVD-GkNN method. This method involves three aspects: dataset processing, filtering and refinement. The processing of the dataset is shown in Equation (14).

$$\begin{cases} \frac{\partial dist(q,Q)}{\partial x} = \sum_{i=1}^{n} \frac{x-x_i}{\sqrt{(x-x_1)^2+(y-y_i)^2}} = 0 \\ \frac{\partial dist(q,Q)}{\partial y} = \sum_{i=1}^{n} \frac{y-y_i}{\sqrt{(x-x_1)^2+(y-y_i)^2}} = 0 \end{cases} \tag{14}$$

(14) In Equation (14), q is the centre of mass of the dataset Q, (x, y) serves as the coordinates of q and (x_i, y_i) serves as the coordinates of any point in the dataset Q. However, when the value of n is taken to be 2, Equation (14) does not give a closed solution and only an estimated value can be obtained, and the centre of mass q can only be obtained as an approximation. Therefore, for obtaining a good estimate, the coordinates of the centre of mass need to be modified, as shown in Equation (15).

$$\begin{cases} x = x - \mu \frac{\partial dist(q,Q)}{\partial x} \\ y = y - \mu \frac{\partial dist(q,Q)}{\partial y} \end{cases} \tag{15}$$

In Equation (15), x represents the modified prime horizontal coordinates, y represents the modified prime vertical coordinates, and μ represents the step size. The algorithm for querying the GkNN of point sets in a road network environment is NVD-GkNN (Q, P, k), where Q represents the query point set, P represents the generated point set, and k is the value of the GkNN query. In addition, the study also analysed the impact of adding and removing points on GkNN. The algorithm for the effect of added points on GkNN is ADDNVD-GkNN (Q, P, k, w), where w is the added point in the generated point set, while the algorithm for the effect of deleted points on GkNN is DENVD-GkNN (Q, P, k, d), where d is the deleted point in the range $d(d \in P)$.

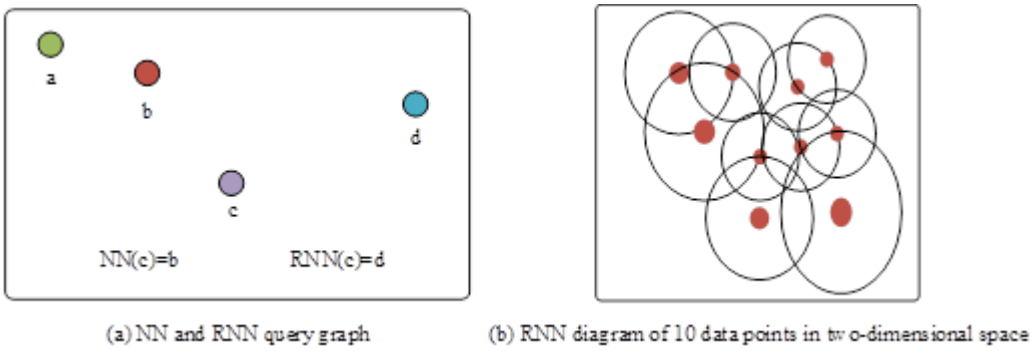
4 Analysis of Spatial Database Optimization Results Based on Voronoi Diagrams and kNN

4.1 Analysis of the Results of kNN-based Spatial Database Optimization

The study under the obstacle space mainly uses the kNN-Obs algorithm. The following contentl is a comparative analysis of the kNN-Obs algorithm and the Pruning algorithm (PostPruning) algorithm, in terms of both the k value and the obstacle dimension. Firstly, the impact of different k values on the Central Processing Unit (CPU) running time (RT) and page views was analysed, as shown in Figure 7.

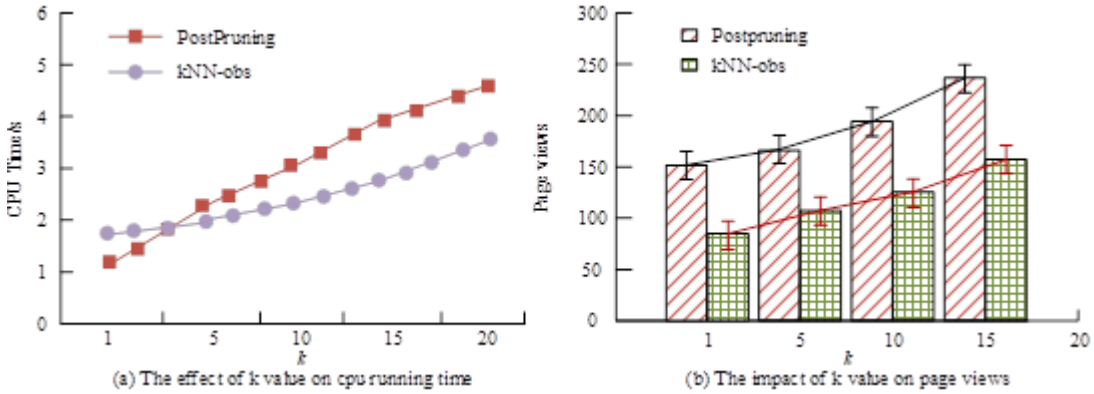
Figure 7(a) showed that the CPU runtime of the kNN-Obs and PostPruning algorithms increased with the value of k. The maximum value of the CPU runtime of the kNN-Obs algorithm was between 3s and 4s, and the minimum value was close to 2s. The maximum value of the CPU runtime of the PostPruning algorithm was between 4s and 5s, and the minimum value was around 1.1s. When the value of k was less than 4s, the PostPruning algorithm CPU runtime was shorter. When the value of k was greater than 4s, the kNN-Obs algorithm CPU runtime was shorter. Figure 7(b) illustrates that as the k values of the kNN-Obs and PostPruning algorithms gradually increased, their respective page views also increased. The maximum value of page views for the kNN-Obs algorithm was close to 150 and the minimum value was close to 100. The maximum value of page views for the PostPruning algorithm was close to 250 and the minimum value was around 150. The overall number of page views for the kNN-Obs algorithm was smaller than that for the PostPruning algorithm. Therefore, on the whole, the kNN-Obs algorithm outperformed the PostPruning algorithm. The number of different obstacles also had an impact on CPU runtime and page accesses, as shown in Figure 8.

As can be seen in Figure 8(a), the CPU runtime of the kNN-Obs and PostPruning algorithms gradually got larger as the quantity of obstacles increases. The kNN-Obs had a maximum CPU runtime of between 6s and 8s, with a minimum value close to 2s, while PostPruning had a maximum CPU runtime of between 10s and 12s, with a minimum value close to 4s. As can be seen in Figure 8(b), the number of page views for both the kNN-Obs and PostPruning algorithms increased as the quantity of obstacles increased. The kNN-Obs had a maximum value of around 175 page views and a minimum value of close to 100, while PostPruning had a extreme value around 150. In summary, the kNN-Obs algorithm outperformed the PostPruning algorithm. In order to better validate the performance of the kNN Obs algorithm, comparative analysis was conducted from the accuracy and F1 value of the algorithm. In the experimental environment, the CPU frequency was 2.0 GHz, the memory was 4GB, and



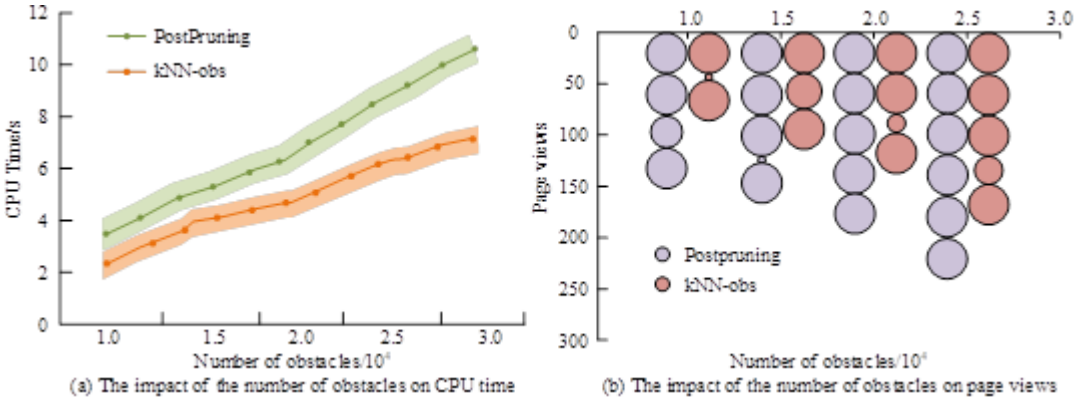
(a) NN and RNN query graph (b) RNN diagram of 10 data points in two-dimensional space

Figure 6: NN and RNN query graph and RNN graph for 10 data points in two dimensional space



(a) The effect of k value on cpu running time (b) The impact of k value on page views

Figure 7: The impact of k value on CPU runtime and page views



(a) The impact of the number of obstacles on CPU time (b) The impact of the number of obstacles on page views

Figure 8: The impact of the number of obstacles on CPU runtime and page views

the operating system was Windows 7. A total of 4 performance tests of the algorithm were conducted. The comparison of accuracy and F1 values between the kNN Obs algorithm and the PostPruning algorithm was shown in Table 1.

From Table 1, the maximum accuracy of the kNN Obs algorithm was 97.9%, while the minimum accuracy was 95.7%. The maximum accuracy of the PostPruning algorithm was 94.6%, and the minimum accuracy was 92.9%. The accuracy of the kNN Obs algorithm was generally about 3% higher than that of the PostPruning algorithm. The maximum F1 value of the kNN Obs algorithm was 0.981, and the minimum value was 0.964. The maximum F1 value of the PostPruning algorithm was 0.921, and the minimum value was 0.892. It can be seen that the F1 value of the kNN Obs algorithm was much higher than that of the PostPruning algorithm, which also indicated that the performance of the kNN Obs algorithm was superior to that of the PostPruning algorithm.

4.2 Analysis of Spatial Database Optimization Results Based on Voronoi Diagrams and kNN

The study employed a grid Voronoi graph-based reverse nearest neighbor algorithm, NVD-RNN, in a road network environment. The CPU RT of the NVD-RNN algorithm, the Expectation Propagation (EP) algorithm and the Auto-Regression k Nearest Neighbors (ARkNN) algorithm were compared and analysed. The outcomes of the comparison of the three algorithms were shown in Figure 9.

In Figure 9, k was the number of target nodes to be obtained and D represented the number of generated points. From Figure 9(a), it indicated that the CPU RT of the three algorithms increased gradually under the growth of k value. The maximum CPU RT of the NVD-RNN algorithm was close to 900s and the minimum value was around 400s. The maximum CPU RT of the EP algorithm was around 1100s and the minimum value was close to 300s. The maximum CPU RT of the ARkNN algorithm was over 1000s, with a minimum value of around 200s. The NVD-RNN algorithm CPU runtime started to be smaller than the EP algorithm when the value of k was greater than 7s, and the NVD-RNN algorithm CPU runtime started to be smaller than the ARkNN algorithm when the value of k was greater than 17s. Figure 9(b) showed that the CPU runtime of the three algorithms increased gradually as the value of D increased. The maximum CPU runtime of the NVD-RNN algorithm was about 3s and the minimum value was about 1.3s. The maximum CPU runtime of the EP algorithm was about 5.3s and the minimum value was close to 3s, while the maximum CPU runtime of the ARkNN algorithm was close to 4s and the minimum value was about 2.1s. The minimum value was about 2.1s. In summary, the NVD-RNN algorithm was greatly superior to the rest algorithms. In order to further validate the performance of the NVD-RNN algorithm, the study selected Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) as comparative indicators. The experimental environment was also under the Windows 7 system, and the number of experiments was also 4. The comparison of mean square error and root mean square error of NVD-RNN algorithm, ARkNN algorithm, and EP algorithm was shown in Table 2.

From Table 2, it can be seen that the maximum MSE value of the NVD-RNN algorithm was 1.25 and the minimum value was 1.13. The maximum MSE value of the ARkNN algorithm was 2.01, and the minimum value was 1.87. The maximum MSE value of the EP algorithm was 2.51, and the minimum value was 2.37. The maximum RMSE value of the NVD-RNN algorithm was 0.971, and the minimum value was 0.863. The maximum RMSE value of the ARkNN algorithm was 1.373, and the minimum value was 1.329. The RMSE of the EP algorithm had a maximum value of 1.572 and a minimum value of 1.542. From this, it can be seen that the NVD-RNN algorithm had significantly lower values in MES and RMSE than the ARkNN algorithm and EP algorithm, which also indicated that the NVD-RNN algorithm had better performance. The k-nearest neighbor NVD-GkNN algorithm for the Voronoi group under the road network was the second algorithm. The following was a comparative analysis of the NVD-GkNN algorithm, the Incremental Euclidean Restriction (IER) algorithm and the Timing Algorithm (TA) algorithm, as shown in Figure 10.

Figure 10(a) illustrated that the CPU runtime of all three algorithms increased as the value of k increased. The maximum CPU runtime of the NVD-GkNN algorithm was about 2s and the minimum value was about 0.6s. The maximum CPU runtime of the IER algorithm was about 3.2s and the minimum value was about 0.8s. The maximum value of the TA algorithm CPU runtime was about 4.6s and the minimum value was about 2.1s. On the whole, the NVD-GkNN algorithm CPU runtime was always below the other two algorithms' value. Figure 10(b) shows that as the value of k gradually grew, the page accesses of the three algorithms also gradually increased. The maximum value of page accesses for the NVD-GkNN algorithm was around 280 and the minimum value was around 130. The maximum value of page accesses for the IER algorithm was around 340 and the minimum value was around 150. The maximum value of page accesses for the TA algorithm was around 450 and the minimum value was around 280. On the whole, the page views of the NVD-GkNN algorithm were consistently lower than those of the other two algorithms. In summary, it can be seen that the NVD-GkNN algorithm had a clear advantage. In addition to comparing the impact of different k values for the three algorithms on CPU runtime and page accesses, the study also compared the influence of the query point set Q on CPU runtime and page accesses for the three algorithms, as shown in Figure 11.

Figure 11(a) demonstrated that the CPU runtime of

Table 1: Comparison of accuracy and F1 values between the kNN Obs algorithm and the PostPruning algorithm

Algorithm	Number of experiments							
	1		2		3		4	
	Precision	F1	Precision	F1	Precision	F1	Precision	F1
PostPruning	93.7%	0.903	94.6%	0.892	92.9%	0.913	93.2%	0.921
kNN-Obs	95.8%	0.964	96.4%	0.976	95.7%	0.981	97.9%	0.975

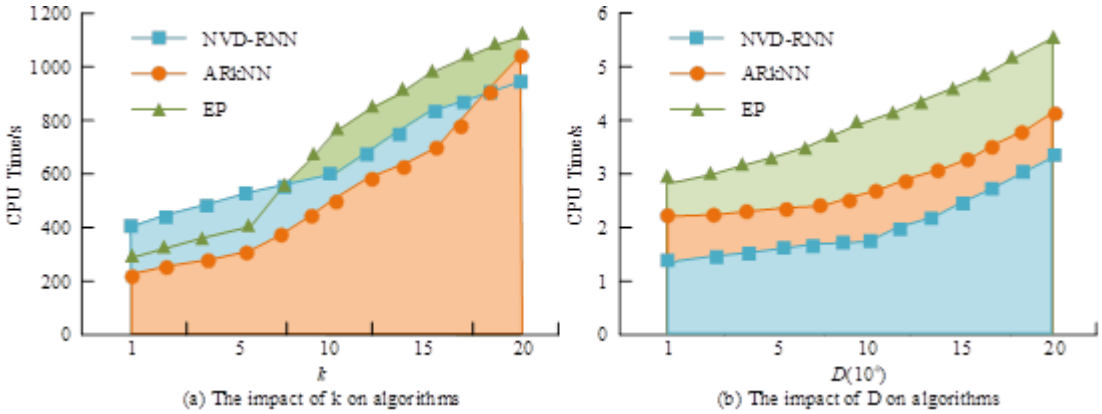


Figure 9: The impact of changes in k and D values on CPU running time

Table 2: Comparison of mean square error and root mean square error of NVD-RNN algorithm, ARkNN algorithm and EP algorithm

Algorithm	Number of experiments							
	1		2		3		4	
	MSE	RMSE	MSE	RMSE	MSE	RMSE	MSE	RMSE
NVD-RNN	1.21	0.971	1.17	0.863	1.25	0.954	1.13	0.874
ARkNN	1.87	1.373	1.97	1.329	2.01	1.356	1.89	1.337
EP	2.37	1.542	2.45	1.568	2.51	1.572	2.47	1.556

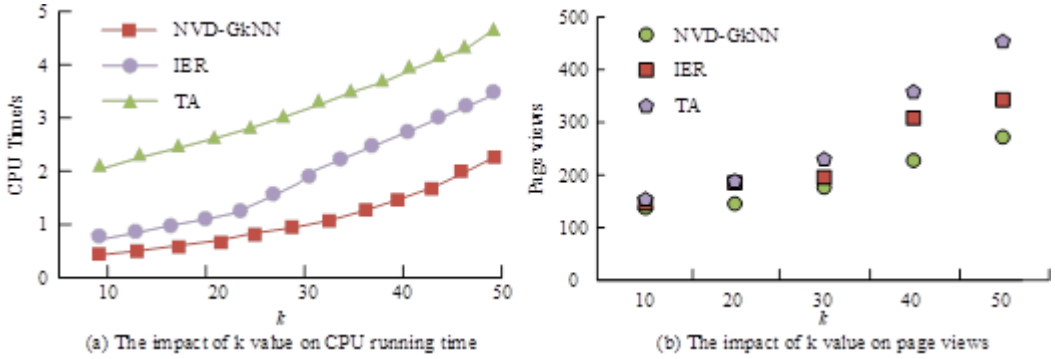


Figure 10: The impact of k value on CPU runtime and page views

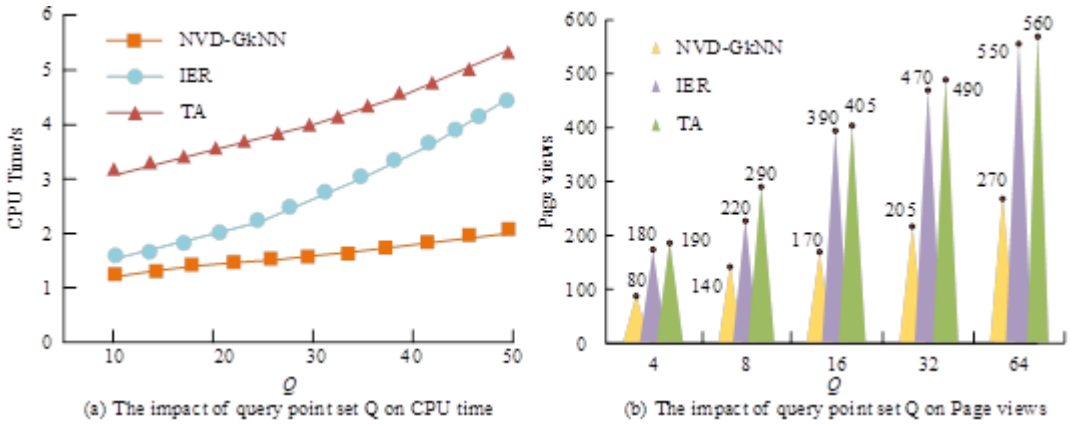


Figure 11: The impact of query point set Q on CPU running time and page views

all three algorithms grew as the value of Q increased. The maximum CPU runtime of the NVD-GkNN algorithm was about 2.5s and the minimum value was about 1.2s. The maximum CPU runtime of the IER algorithm was about 4.3s and the minimum value was about 1.7s. The maximum value of the TA algorithm CPU runtime was about 5.2s and the minimum value was about 3.1s. As can be seen from Figure 11(b), the page views of all three algorithms increased with the value of Q. The maximum value of page views for the NVD-GkNN algorithm was about 270 and the minimum value was about 80. The maximum value of page views for the IER algorithm was about 550 and the minimum value was about 180. The maximum value of page views for the TA algorithm was about 560 and the minimum value was about 190. In summary, it can be seen that the NVD-GkNN algorithm had more obvious advantages. In the following, the dynamic update algorithm, TA algorithm and IER algorithm were compared, as shown in Table 3 and Table 4.

Table 3: Performance comparison of dynamic update algorithm (ADDNVD-GkNN) with TA and IER algorithms

Algorithm	P	
	1000	1001
ADDNVD-GkNN	1.262	1.972
TA	3.102	6.165
IER	1.857	3.283

As can be seen from Table 3 and Table 4, the comparison in the ADDNVD-GkNN algorithm, the DENVD-GkNN algorithm and the other two algorithms was performed mainly in two dimensions: execution time and page accesses. When the number of page views was 1000, the execution time of the ADDNVD-GkNN algorithm was 1.84s and 0.595s faster than the TA algorithm and the IER algorithm's value, respectively. When the number of page views was 1001, the execution time of the ADDNVD-GkNN algorithm was 4.193s and 1.311s faster than the TA

Table 4: Performance comparison of dynamic update algorithm (DENVD-GkNN) with TA and IER algorithms

Algorithm	P	
	2000	1999
DENVD-GkNN	2.009	2.168
TA	4.326	8.186
IER	2.634	5.154

algorithm and the IER algorithm's value, respectively. In summary, it can be seen that ADDNVD-GkNN algorithm outperformed the TA algorithm and the IER algorithm. When the quantity of page views was 2000, the execution time of the DENVD-GkNN algorithm was 2.317s and 0.625s faster than the TA and IER algorithms' value, respectively. When the number of page views was 1999, the execution time of the DENVD-GkNN algorithm was 6.018s and 2.986s faster than the TA and IER algorithms' value, respectively. It can be seen that the DENVD-GkNN algorithm was faster than the TA and IER algorithms. GkNN algorithm had an advantage over the TA algorithm and the IER algorithm.

5 Conclusion

To address the shortcomings of the nearest neighbor query and its variants in the obstacle and road network environments, the study combines Voronoi diagrams and grid Voronoi diagrams with the nearest neighbor query and its variants. These methods are then used to query spatial data in the obstacle space and road network environments and compared with existing spatial database query methods. The results show that after a value of k greater than 4s, the kNN-Obs algorithm's minimum CPU runtime was 0.2s less than the PostPruning algorithm, and the maximum runtime was 1.2s less. The kNN-Obs has a minimum of 50 and a maximum of around 100 fewer page

views than the PostPruning algorithm. After a value of k greater than 7s, the CPU runtime of the NVD-RNN algorithm is at least about 100s less than that of the EP algorithm, and at most about 180s less. After a value of k greater than 17s, the CPU runtime of the NVD-RNN algorithm is at least about 50s less than that of the ARkNN algorithm. Under the influence of the k value, the CPU runtime of the NVD-GkNN algorithm is at least 0.1s and 1.2s less than that of the IER and TA algorithms, and at most 1.2s and 1.4s less. The page views under the NVD-GkNN algorithm are at least 10 and 20 less than that of the IER and TA algorithms, and at most 90 and 110 less. Under the influence of the Q set, the CPU RT of the NVD-GkNN algorithm is at least 0.3s and 1.9s less than that of the IER and TA algorithms, and at most 2.8s and 3.5s less than that of the IER and TA algorithms. The page views under the NVD-GkNN algorithm are at least 50 and 65 less than that of the IER and TA algorithms, and at most 290 and 300 less. The research results have improved the query efficiency of spatial databases, maintained data security of the database, and also contributed to network security. Future research can continue to explore the spatial data query method based on Voronoi diagram, improve the existing query method and enhance the query efficiency.

Acknowledgments

The research is supported by: the Henan Provincial Key R & D and Promotion Special (Science and Technology Tackling) Project, China. Project name: Research on the consistency checking algorithm of spatial direction relation in spatial database (No.232102210085).

References

[1] F. Baccelli, S. S. Kalamkar, "On point processes defined by angular conditions on Delaunay neighbors in the Poisson-Voronoi tessellation," *Journal of Applied Probability*, vol. 58, no. 4, pp. 952-965, 2021.

[2] D. Bonnet, S. Cabello, B. Mohar, H. Pérez-Rosés, "The inverse voronoi problem in graphs I: Hardness," *Algorithmica*, vol. 82, no. 10, pp. 3018-3040, 2020.

[3] H. Cai, Y. Zhu, J. Li, J. Yu, "A profit-maximizing mechanism for query-based data trading with personalized differential privacy," *The Computer Journal*, vol. 64, no. 2, pp. 264-280, 2020.

[4] H. Chen, J. V. D'Silva, H. Chen, B. Kemme, L. Hendren, "HorseIR: bringing array programming languages together with database query processing," *ACM SIGPLAN Notices*, vol. 53, no. 8, pp. 37-49, 2020.

[5] G. Ding, H. Sun, J. Li, C. Li, Y. Fei, "An efficient relational database keyword search scheme based on combined candidate network evaluation," *IEEE Access*, no. 99, pp. 30863-30872, 2020.

[6] T. Eiter, R. Ichise, J. X. Parreira, P. Schneider, L. Zhao, "Deploying spatial-stream query answering in C-ITS scenarios," *Semantic Web*, vol. 12, no. 1, pp. 41-77, 2021.

[7] F. Feng, S. Xiong, Z. Liu, Z. Xian, Y. Zhou, H. Kobayashi, "Cellular topology optimization on differentiable Voronoi diagrams," *International Journal for Numerical Methods in Engineering*, vol. 124, no. 1, pp. 282-304, 2023.

[8] Y. Gulzar, A. A. Alwan, S. Turaev, "Optimizing skyline query processing in incomplete data," *IEEE Access*, vol. 7, no. 1, pp. 178121-178138, 2019.

[9] J. Jakob, M. Guthe, "Optimizing LBVH offshore construction and Hierarchy placeholder raversal to accelerate kNN Queries on Point Clouds using the GPU," *Computer Graphics Forum*, vol. 40, no. 8, pp. 124-137, 2020.

[10] S. Jang, Y. E. Jang, Y. J. Kim, H. Yu, "Input initialization for inversion of neural networks using k-nearest neighbor approach," *Information Sciences*, vol. 519, no. 9, pp. 229-242, 2020.

[11] A. Jg, A. Wl, A. Zl, Z. B. Jian, S. B. Li, "A general fragments allocation method for join query in distributed database-ScienceDirect," *Information Sciences*, vol. 512, no. 13, pp. 1249-1263, 2020.

[12] K. Koufos, H. S. Dhillon, M. Dianati, C. P. Dettmann, "On the kk nearest-neighbor path distance from the typical intersection in the Manhattan Poisson line cox process," *IEEE transactions on mobile computing*, vol. 22, no. 3, pp. 1659-1671, 2023.

[13] T. Liu, S. Wang, Z. Lei, J. Zhang, X. Zhang, "Trajectory risk cognition of ship collision accident based on fusion of multi-model spatial data," *Journal of Navigation*, vol. 75, no. 2, pp. 299-318, 2022.

[14] J, Park, H, L. Dong, "Parallelly running k-nearest neighbor classification over semantically secure encrypted data in outsourced environments," *IEEE Access*, vol. 8, pp. 64617-64633, 2020.

[15] Y. Ruan, Y. Xiao, Z. Hao, B. Liu, "A nearest-neighbor search model for distance metric learning," *Information Sciences*, vol. 552, no. 3, pp. 261-277, 2020.

[16] P. Sarode, T. R. Reshmi, "Optimized query ordering data aggregation model using neural networks and group search optimization in wireless sensor network," *Ad Hoc & Sensor Wireless Networks*, vol. 46, no.3, pp. 189-214, 2020.

[17] F. Shi, H. Cao, X. Zhang, X. Chen, "A reinforced k-nearest neighbors method with application to chatter identification in high-speed milling," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 12, pp. 10844-10855, 2020.

[18] J. Shi, L. Yang, "A climate classification of China through k-Nearest-neighbor and sparse subspace representation," *Journal of Climate*, vol. 33, no. 1, pp. 243-262, 2020.

[19] J. Wang, J. Shen, "Fast spectral analysis for approximate nearest neighbor search," *Machine Learning*, vol. 111, no. 6, pp. 2297-2322, 2022.

[20] A. Yl, C. Rxab, W. A. Xin, A. Hl, Z. Shi, D. Xin, "GLDH: Toward more efficient global low-density locality-sensitive hashing for high dimensions," *Information Sciences*, vol. 533, pp. 43-59, 2020.

Biography

Huili Xia, born in November 1980, female, from Wugang City, Henan Province, Han ethnicity. She obtained her Bachelor's degree in Computer Science and Technology from Zhengzhou University of Light Industry in 2005 and her Master's degree in Computer Application Technology from Hubei University of Technology in 2008. Since 2008, she has been a full-time teacher at Zhengzhou University of Economics and Business. She published fifteen academic papers, including seven papers in Chinese core journals and one paper indexed by EI. She edited two textbooks, hosted one Henan Provincial Key R & D and Promotion Special (Science and Technology Tackling) Project, and participated in four provincial-level scientific research projects, has been granted one utility model patent by the China National Intellectual Property Administration.