IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 18, 2021, accepted May 27, 2021, date of publication June 3, 2021, date of current version June 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3085605

POD: A Parallel Outlier Detection Algorithm Using Weighted kNN

YANG MA AND XUJUN ZHAO
School of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan 030024, China
Corresponding author: Xujun Zhao (zxj0226@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572343, in part by the Natural Science Foundation of Shanxi Province of China under Grant 201901D111257 and Grant 201901D211303, and in part by the Taiyuan University of Science and Technology Scientific Research Initial Funding under Grant 20192013.

ABSTRACT Outlier detection algorithms based on k nearest neighbors (kNN) can effectively find outliers from massive data, but most algorithms are difficult to adapt to high-dimensional data sets. In order to highlight the importance of attributes in k nearest neighbors, we propose a weighted kNN query method, which uses the Z-order curve to find the kNN. The method first applies information entropy to calculate each attribute weight, and then uses the Z-order curve to encode high-dimensional data into Z-value. The weighted kNN of each object are searched according to its Z-value. Meanwhile, a novel outlier detection algorithm is presented based on the minimum distance and average distance between each object and its weighted kNN. On this basis, we propose a parallel outlier detection algorithm called POD to improve the efficiency of the outlier detection. Finally, we implement and evaluate POD algorithm on a 10-nodes Hadoop cluster, on which synthetic and UCI standard data are tested. Experimental results show that POD achieves high performance in terms of effectiveness, scalability and extensibility.

INDEX TERMS Outlier detection, k nearest neighbors, weights, Z-order curve.

I. INTRODUCTION

The k nearest neighbor(kNN) query is one of the simplest machine learning algorithms. It is a method for querying k closest objects to a given object in multidimensional space [1], which is widely used in databases [2], data mining [3], geographic information systems [4] and other fields [5], [6]. Nevertheless, the kNN query which performs well in low-dimensional space has different degrees of deterioration in high-dimensional space, for example: R-tree [7] and KD-tree [8]; most methods [9], [10] assume that all the attributes of the data are equally important, but in practical applications, the importance of each attribute is different, and each attribute has a different contribution to the nearest neighbor query. Thus ignoring the importance of the attributes will seriously affect the kNN query.

Z-order is a space filling curve, which can effectively map data objects in d-dimensional space into one-dimensional subspace, and transform weighted kNN query in the high-dimensional space into a linear space query. It can improve kNN query efficiency. Outliers are some abnormal data objects that are inconsistent with the general behavior or model. kNN query is a staple method in outlier detection, which affects the efficiency and accuracy of outlier results. However, most algorithms based on kNN query are difficult to adapt to high-dimensional data sets.

In this paper, using the Z-order curve, a weighted kNN query method and outlier data mining algorithm are proposed. Firstly, the importance of all attributes is measured by the information entropy, and the high-dimensional data objects are encoded by the Z-order curve and mapped into a Z-value. Secondly, the weighted kNN of each object is queried according to the Z-value, so that it improves the efficiency of kNN query in high-dimensional space. Then, based on the minimum distance and average distance between each object and its weighted kNN, the weighted kNN method is used for outlier detection. Finally, the experimental results verify that the effectiveness of the weighted kNN query on both synthetic and UCI data sets.

A. MOTIVATIONS

The outlier detection algorithm based on kNN in this study is motivated by the following three observations:

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

- The traditional k nearest neighbor query method is difficult to adapt to high-dimensional data sets.
- Most k nearest neighbor query methods ignore some differences among attributes in a dataset.
- With the increase of data volume and data dimension, many existing outlier detection algorithms based on kNN have poor performance.
- With the rapid development of data acquisition methods, the outlier detection algorithm on a single node can not meet the requirements of practical application.

Motivation 1. k nearest neighbor is a common query method in outlier data mining, but the traditional k nearest neighbor query method is difficult to adapt to massive high-dimensional data sets. The k nearest neighbor query method which performs well in low dimensional space has different degrees of deterioration in high-dimensional space. In particular, when the amount of data increases rapidly, its k nearest neighbor query method has low processing efficiency, and it is difficult to meet the actual needs of massive data analysis.

Motivation 2. Most k nearest neighbors query methods regard all attributes as equally important. However, in practical application, the importance of attributes is different. If we ignore the differences among attributes, we will not be able to query meaningful neighbor data objects.

Motivation 3. In the high-dimensional outlier data mining, the traditional outlier detection algorithm based on kNN mostly uses data object and its kNN object count as the outlier score to measure the abnormal trend of the object. These algorithms will be affected by the ''dimension disaster'' in high-dimensional space, which leads to the poor effect of outlier mining. In addition, these algorithms need to iterate the artificially set parameters to get a satisfactory outlier result set, and the selection of parameters has no prior knowledge to use for reference. So the accuracy of outlier mining is low.

Motivation 4. MapReduce is a kind of parallel programming model with scalability and high fault tolerance. MapReduce divides and distributes data to multiple work nodes, and processes these data by using the parallelism between cluster data nodes. With the rapid development of data acquisition methods, the outlier detection algorithm on a single node can not meet the requirements of practical application. The design of parallel outlier algorithm based on MapReduce is becoming an important way for processing extreme-scale data.

B. OUR APPROACH AND CONTRIBUTIONS

Our key contributions are summarized as follows.

1) We propose an attribute weighting method based on information entropy, which can effectively improve the accuracy and pertinence of k nearest neighbor query. In order to meet the requirements of massive data sets, we propose a sampling method for outlier detection. In the sample data set, information entropy is used to calculate the weights of each attribute, which provides a weighted data set for the subsequent parallel outlier detection algorithm.
2) We propose a weighted k nearest neighbor query method based on Z-order, in which the weighted data objects are mapped to the Z-order space filling curve and the corresponding Z-value is generated. So we can get the weighted k nearest neighbor set of data objects, which can be effectively used for high-dimensional data. At the same time, we design its parallel method, which uses Hadoop parallel computing platform.
3) In parallel computing weighted k nearest neighbor sets, we use LSH strategy to divide the k nearest neighbor candidate sets, and put the similar Z-value objects on the same computing node. This strategy can alleviate the problem of data skewness between nodes and improve the accuracy of weighted k nearest neighbor query.
4) We propose a novel outlier data mining algorithm based on weighted k nearest neighbors, which combines the individual differences between each object and its weighted k nearest neighbors. We also design a parallel outlier detection algorithm, which can effectively run on Hadoop platform.
5) We implement and evaluate our proposed parallel algorithm on a 10-nodes Hadoop cluster, on which synthetic and UCI standard data are tested. Experimental results show that our algorithm achieves high performance in terms of effectiveness, scalability and extensibility.

C. ROADMAP

The rest of the paper is structured as follows: Section II introduces the related work on k nearest neighbors and outlier detection algorithm. Section III proposes a method of weighted k nearest neighbor. In section IV, the outlier detection algorithm based on weighted kNN is introduced and analyzed. Section V gives a parallel algorithm's design method and Section VI describes experimental settings and offers result analysis. Finally, we conclude our study in Section VII.

II. RELATED WORK

A. k NEAREST NEIGHBOR

Given a new input object in the training data set, the kNN query returns k objects in the data set with the smallest distances to it. The search process generally compares all objects in the training set with the new input objects and identifies the most similar k objects from them [7]. Typical researches have been proposed to solve kNN query. Roussopoulos *et al.* [7] proposed to use the depth-first traversal of R-tree to query kNN. In recursion and backtracking, the subtrees which does not contain the nearest neighbor are pruned, avoiding traversal of the entire R-tree. However, during the tree construction process, there is an overlap between the minimum boundary rectangles. The nearest neighbor query needs to access a large number of nodes, and almost linearly scans the entire data set, therefore it is not suitable for high-dimensional data; In [8], it is mentioned that buffer k-d tree [11] is the fastest kNN algorithm, which uses buffer and modern many-core devices such as GPU to accelerate

in parallel. On this basis, Yang *et al.* [12] propose a new kNN algorithm, which greatly reduces unnecessary visiting nodes and distance computations; Indyk and Motwani [13] proposed LSH, an approximation metric algorithm between similar objects. It uses a specific hash function to build a hash table, so that similar objects are mapped to the same bucket with higher probability, thus their nearest neighbors are found in each bucket. However, with the advent of the era of big data, the LSH algorithm can no longer meet the actual needs; Lin *et al.* [14] proposed a nearest neighbor query based on Voronoi diagram. It divides the rule-based Voronoi diagram into different cells. Each cell region is represented by a minimum outer rectangle and its index is constructed. Nearest neighbors query is achieved by querying the corresponding index; Zhang Liping [4] uses the filtering function of the Voronoi diagram to effectively reduce the number of points to be queried, and performs secondary screening on the objects in the candidate set according to the obstacle distance and the adjacent generation points, thereby kNN are queried from the selected candidate set; Shichao Zhang [15] proposed a k parameter computation, called S-kNN approach, by replacing the fixed k value for all test samples with learning different k values for different test samples according to the distribution of the data. GOU et al [16], [17]. proposed a generalized mean distance-based k nearest neighbor classifier, which effectively reduces sensitiveness to parameter k. In addition, they fused k-means local vector into k nearest neighbor algorithm [16], which effectively improved the robustness and effectiveness of kNN algorithm. In [18], a new algorithm is developed which uses the ensemble technique and obtains better results.

B. OUTLIER DETECTION METHOD

Outliers [19], [20] are data objects that are inconsistent with the general behavior or model of the data. So, as an important branch of data mining, outlier detection can often find some real but unexpected knowledge. kNN query is a staple method in outlier detection, which affects the efficiency and effect of outlier detection. Ramaswamy and Kyuseok *et al.* [21] first proposed an outlier detection method based on kNN. This method mainly uses the distance between data points and its kNN as the basis for judging outliers, but cannot determine which point is more likely to be an outlier at the same distance; Angiulli and Pizzuti [22] extended the above method on the basis of the outliers factor, and proposed that the outliers factor of each point in the data set are the sum of distances between their own and their kNN. The larger the outliers factor, the greater the probability of the corresponding points become outliers. But it only considers the overall level of the data point and its kNN, without taking into account the individual differences between them. In [23], a fuzzy clustering anomaly detection algorithm based on session feature similarity is proposed, which improves the efficiency of anomaly detection; In [24], Liu *et al.* regard outlier detection as a binary classification problem, and extract potential outliers from uniform reference distribution. On this basis, they propose a single target generation antagonism active learning outlier detection method. Debasrita *et al.* [25] classified the outlier data and proposed an outlier detection system, which used stacked automatic encoder to extract features, and then used a whole probabilistic neural network to detect outliers. Feng *et al.* [26] proposed an outlier detection method based on Entropy Measurement on spark computing platform. The method is divided into three stages. In the first stage, the attribute entropy is calculated. In the second stage, k nearest neighbors are found. In the third stage, each point is sorted according to the degree of outliers, and outliers are detected. Li *et al.* [27] proposed a parallel outlier mining method based on feature grouping for high-dimensional classification data sets, which uses vertical transformation to improve the efficiency and accuracy of outlier detection.

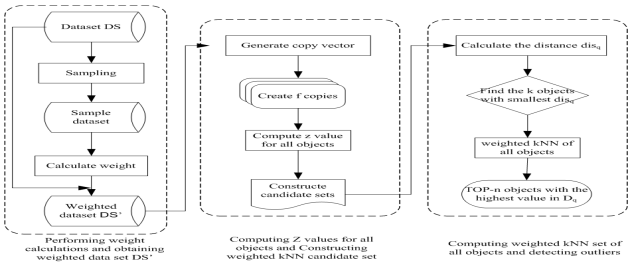


FIGURE 1. The description of outlier detection process.

In summary, most kNN query are based on the same importance of all attributes, that is, all attributes of data are assumed to be equally important, but in many practical applications, each attribute is of different importance to the neighbor query. In the meantime, some outlier detection methods do not consider the impact of the average distance on outliers. Other methods only consider the overall level of data points and their kNN without considering the individual difference between data points. Therefore, when detecting outliers, it is necessary to take into account overall level and individual differences at the same time. In this paper, the main process of the outlier detection is divided into three parts, as shown in Fig.1. In the first part, we perform weight calculations in the sample dataset by using two formulas, and obtain the weighted data set DS'. In the second part, we compute Z-values for all objects and construct weighted a kNN candidate set. In the third part, we compute weighted kNN set of all objects and detect outliers.

III. WEIGHTED k NEAREST NEIGHBOR

In the process of computing the kNN of each object using weighted kNN, the importance of the attributes (i.e., the weights of the attributes) must be fully considered. In the absence of prior knowledge of experts or users, the importance of attributes cannot be directly given, that is, the attribute weight is uncertain. Information entropy uses the average information to reflect the uncertainty of each discrete message of the source, so it can better solve the problem that uncertainty of attribute weight.

A. INFORMATION ENTROPY AND WEIGHTED k NEAREST NEIGHBOR

Information entropy [28] is defined as the average probability of occurrence of discrete random events. The average information is used to reflect the uncertainty of each discrete message of the source, effectively characterizing the quantitative measurement problem of information, and is applied to the measurement of the attribute weight [29]. Referring to [29], the related concepts and definitions are described as follows:

Given a data set DS, $A = \{A_1, A_2, \ldots, A_d\}$ is the set of attributes of the DS. Since each attribute needs to be given a weight, each attribute can be treated as a single source, and n values of the attribute (for example, $A_i = A_{i1}, A_{i2}, \ldots, A_{in}$) are considered as the n kinds source symbols issued by the source. It is assumed that $P_l = P_l(A_{il})$ represents the probability of occurrence of the source symbol A_{il} (or called an event). A_{il}'s information $I(A_{il}) = -log_2 P_l$ which indicates the information contained in the event A_{il} when an event occurs. For n mutually independent source symbols, n probabilities will be generated, namely $P_1 \ldots P_i \ldots P_n$. The information entropy of the source is the average uncertainty describing the source, i.e., the average information of the source can be measured by the average of all individual symbol information.

$$H(A_l) = E(I(A_{il})) = -\sum_{l=1}^{d} p_l \log_2 p_l. \tag{1}$$

In this paper, the larger $H(A_l)$, the higher importance of the attribute A_l, and the more information of the attribute. Information entropy represents the general characteristics of the attribute A_l in an average sense, consequently the information entropy of all attributes are normalized to determine the weight of each attribute.

Given the attribute sets $A = A_1, A_2, \ldots, A_d$ of data set DS, the attribute weight is $W(A) = w_1, w_2, \ldots, w_d$, where:

$$w_l = \frac{H(A_l)}{\sum_{l=1}^{d} H(A_l)} (l = 1, 2, \ldots, d). \tag{2}$$

W_l is the weight of the lth attribute in the attribute set.

Ramaswamy and Kyuseok [21] proposed a method to solve the outlier detection using the kNN, but it did not consider the influence of different attributes on the result. Therefore, in order to solve this problem, a weighted kNN is proposed. It allots different weights to different attributes, fully considering the impact of the importance of the attribute on the result.

Let x_{il} and x_{jl} be the lth weighted attribute value of the ith and jth objects, and the weighted distance d_{ij} between the two objects is calculated by the Euclidean distance as shown below:

$$d_{ij} = \sqrt{\sum_{l=1}^{d} (x_{il} - x_{jl})^2}. \tag{3}$$

This distance is based on the weighted data set and is not simply a Euclidean distance calculation of the original data set.

Definition 1 (Weighted kNN): The distances between O and other objects are calculated according to formula 3 for any object O on the weighted data set DS', and the weighted kNN of O are obtained, namely k objects with the smallest distance.

B. WEIGHTED k NEAREST NEIGHBOR BASED Z-ORDER

The search methods for weighted kNN are LSH, Voronoi diagram, and Z-order curve, etc. The Z-order curve is a space filling curve that passes through and passes through only once each discrete grid in a high-dimensional space. The original high-dimensional space query was transformed into a linear space range query. Therefore, the Z-order curve can well protect the proximity between high-dimensional data and can be applied to data set with different densities.

The main purpose of using the Z-order curve to realize kNN query is to map data objects from high-dimensional to low-dimensional. Each data object corresponds to a point on the Z-order curve, which is called Z-value. Thus, the weighted kNN query in the data set can be converted into a range query of Z-values in one dimensional space. After weighting all the attributes of the data objects in the high-dimensional space and mapping them onto the Z-order curve, the weighted kNN query of the object is transformed into the query of k objects that are closest to the Z-value of the query object. To improve the accuracy of weighted kNN, the concept of weighted kNN candidate sets and random translation operations is introduced.

Definition 2 (Object Precursors): Given a data set DS, $O = O_1, O_2, \ldots, O_n$ is the object set of DS, Z' is the Z-value of the object O_i. After sorting all Z-values, an ordered sequence is generated. Suppose $Z_1, Z_2, \ldots, Z_k$ is k neighboring values smaller than Z' value, the k objects corresponding to $Z_1, Z_2, \ldots, Z_k$ are called precursors of O_i. Similarly, suppose $Z_{k+1}, Z_{k+2}, \ldots, Z_{2k}$ are k neighboring values greater than the Z' value, the corresponding k objects are called successors of O_i. The set consisting of the precursors and successors of O_i is called the weighted kNN candidate set of O_i. It is denoted as $C(O_i)$.

Definition 3 (Weighted kNN Candidate Set): Let the random translation vector be $\vec{v}_i = (2^m i/(d+1), \ldots, 2^m i/(d+1))(i \in [0, d])$, $\vec{v}_i \in R^d$, d is the dimension, and m is the order (refer to Section 3). According to the random translation vector, the data set DS' is derived from DS (i.e., $DS' = DS + \vec{v}_i$), and the object O in the data set is transformed into O_i. The same operation is performed repeatedly $d+1$ times with different random translation vector $\vec{v}_i$ each time. As a result, $d+1$ random translation copies are generated, denoted as $f, f \in [0, d]$, and then a weighted kNN candidate set is constructed on each copy.

The random translation is an operation of deforming the data set DS. It can make objects whose distances are adjacent

TABLE 1. The inspection results of patient.

name	nasal congestion	headache	hoarse	fever	cough
Jim	2	3	1	0	3
Merry	3	4	0	1	1
Jack	5	0	0	2	3
Joce	3	2	1	2	3
Sun	0	0	1	0	2
Ance	1	0	1	1	0
Lili	4	1	0	3	1
Lucy	1	2	1	1	2

in the original data set but Z-values are not adjacent become the objects with adjacent Z-values after being deformed.

Weighted kNN candidate set and random translation operations can effectively increase the accuracy of weighted kNN, but bring higher time complexity meanwhile. To balance this contradiction, it is considered that performance is the best when the translation time is set to 10 through a large number of experiments.

The detailed steps of searching for weighted kNN by using Z-order curve are as follows:

Step 1. The weight of each attribute is calculated in the original data set *DS* according to the information entropy, and the weighted data set *DS*' is obtained instantly;

Step 2. The weighted attribute values are binary coded, and the binary attribute values of all objects in *DS*' are bit interleaved to obtain the respective Z-values;

Step 3. The objects in data set are reordered according to the Z-value. In the sorted result, the k precursors and k successors of each object are found, thereby determining the weighted kNN candidate set of all objects;

Step 4. In order to ensure the accuracy of the query results, random translation operations are performed on *DS*' according to definition 3, and f random translation copies are generated. A weighted kNN candidate set with $2kf$ elements are generated for all objects in *DS*' after repeat step 2 and step 3 for each copy;

Step 5. The Euclidean distance of each object and all objects in its weighted kNN candidate set is calculated. The first k of the smallest distance is the weighted kNN of the object.

C. EXAMPLES

Table 1 is the inspection results of 8 patients, which records 5 attributes of each patient. The value of nasal congestion and headache in a range between 0 and 5. 1 represents hoarse and 0 represents normal towards the hoarse. The range of fever and cough is 0-3. The larger the value, the more serious the situation for the above attribute.

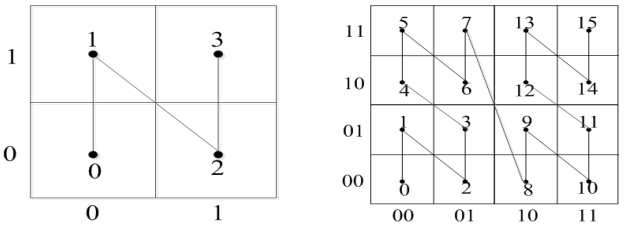
Taking the nasal congestion as an example, the average value of the eight objects in this attribute is 2.375, where it represents unhappen when the value is lower than the average value, and otherwise, it represents occurs. The probability value of nasal congestion *p(nasal congestion)* is 0.5 according to the ratio of the number of samples occurs to the total number of samples. Through formula 1, the information entropy of the nasal congestion *H(nasal congestion)* is 0.5. The information entropy of all attributes are calculated in the same method. Finally, according to formula 2,the weight of the nasal congestion is 0.2102. All results are shown in Table 2.

TABLE 2. Attribute weight.

X	P(X)	H(X)	onormalize H(X)	w
nasal congestion	0.5	0.5	0.2102	0.2102
headache	0.5	0.5	0.2102	0.2102
hoarse	0.625	0.4238	0.1782	0.1782
fever	0.375	0.5306	0.2231	0.2231
cough	0.625	0.4238	0.1782	0.1782

The original inspection record values of the object *Jim* are 2, 3, 1, 0, and 3. The weights of the attributes are 0.2102, 0.2102, 0.1782, 0.2231, and 0.1782 in Table 2. After the relevant weights are given to the respective attributes of Jim, the values become 0.4204, 0.6306, 0.1782, 0, and 0.5346. Other objects are weighted by the same method, and the weighted data set *DS*' is obtained after calculation.

In the process of calculating Z-value, the order of the Z-order curve is first determined. The order represents the degree to which the space is divided. The larger the order, the thinner the space is divided. For a d-dimensional space of order m, each dimension is divided into 2^m parts, then the space is divided into 2^{dm} grids, so the order is determined according to the largest value p_{ij} in the data set. That is, m is the smallest integer that is satisfied $m \geq \frac{\log_2(p_{ij}+1)}{d}$, so that all objects can be mapped to the Z-order curve. In this example, the maximum attribute value is 1.051, so m is calculated as 3. For easy understanding, the two-dimensional Z-order curve of *1*st order and 2nd order are given. The corresponding Z-value $(1011)_2 = 11$ after bit interleaving for the point (3,1) of the 2nd order image in Fig.2. The detailed calculation is shown in Fig.3, where p_1, p_2 are the binary encoding at every attribute of the point p.



(a) 1st order Z-curve (b) 2nd order Z-curve

FIGURE 2. Z-curve.

In the same way, the elements in Table 2 are rounded up to get an integer. Taking the object *Jim* as an example, the values of each attribute after rounded up are 000, 001, 000, 000, and 001, and the Z-value $(000000000001001)_2 = 9$. In turn, the Z-values of the eight objects are 9, 24, 17, 17, 0, 0, 18,

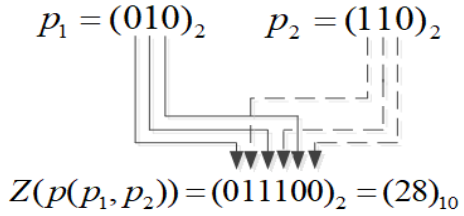


FIGURE 3. Bit interleaving operation of (3,1).

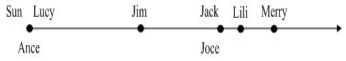


FIGURE 4. Straightened Z-order curve.

and 0, respectively, and then the objects in the data set *DS*' are rearranged according to the Z-value to obtain *Sun*, *Ance*, *Lucy*, *Jim*, *Jack*, *Joce*, *Lili*, and *Merry*. The correspondence relation among them and the straightened curve is shown in Fig. 4.

Let the number of neighbors $k = 2$, we take *Jim* as an example to perform a weighted kNN query. According to Fig. 4, *Jim*'s precursors are *Ance* and *Lucy*, and *Jim* is followed by *Jack* and *Joce*, then *Jim*'s weighted kNN candidate sets are *Ance*, *Lucy*, *Jack* and *Joce*. According to the random translation vector, 5 copies of the data set are generated based on the data set *DS*'. Hence, the weighted kNN candidate set can be found in each data set(a total of 24 neighbor objects are found, removing the same objects). The weighted kNN of *Jim* is *Lili* and *Lucy* according to formula 3.

IV. OUTLIER DETECTION ALGORITHM BASED ON WEIGHTED *k*NN

In the outlier detection, when the kNN method is adopted, the relationship between kNN and the query data object needs to be considered. However, the existing outlier detection method only considers the overall level of the data object and its kNN, or only considers the relationship of the kth nearest neighbor and the data object. These do not reflect the relationship between the data object and its kNN well. Accordingly, individual differences between data objects also need to be considered.

Let the weighted kNN of the query object q be $q_1, \ldots q_k$, the distance between q and each neighbor can be calculated in the light of formula 3, marked as $d_{q,q1}, \ldots, d_{q,qk}$, where $d_{q,qi} (i = 1, 2, \ldots, k)$ is the distance between q and its ith weighted nearest neighbor q_i. The minimum distance between q and the weighted kNN is $d_{min} = min\{d_{q,q1}, \ldots, d_{q,qk}\}$, and the average distance is $d_{ave} = \frac{1}{k}\sum_{i=1}^{k} d_{q,qi}$. A new distance formula is gotten after calculating the arithmetic mean of d_{min} and d_{ave}, as follows:

$$d_q = \frac{d_{min} + d_{ave}}{2}. \tag{4}$$

Algorithm 1 POD

input: data set *DS*, number of data objects n, dimension d, number of copies f, number of nearest neighbors k;
output: Outliers;
Perform weight calculations according to formula 1 and 2 to obtain the weighted data set *DS*';
for (i=1;i<f;i++) **do**
  $\vec{v}_i = (2^m i/(d+1), \ldots, 2^m i/(d+1))$;//Construct a random translation vector, m is the order
  $DS^i = DS + \vec{v}_i$;//Build a copy of the data set
  $Z_i = ComputeZ(DS_i, n)$;//Calculation of Z-values for all objects in data set DS_i
  $C_i = Candidate(DS_i, n, k)$;//Construction of weighted kNN candidate set
end for
$C = C_1 \cup \ldots \cup C_f$; //Integrate weighted candidate set in all replicas
$D_q = WKNN(DS_i, C, n, d, f, k)$; //weighted kNN of all objects
Return the TOP-n objects with the highest value in D_q;

Algorithm 2 ComputeZ(DS_i,n) Function

input: data set DS_i, number of data objects n;
output: Z_i that a set about Z-value;
for (j=0;j<n;j++) **do**
  The attribute values of each object are binary coded;
  Perform bit interleaving operations on binary encode;
  The result of bit interleaving is converted to decimal, that is the Z-value of the object;
  Add the Z-value to the set Z_i;
end for
return Z_i;

Algorithm 3 Candidate(DS_i,n,k) Function

input: data set DS_i, number of data object n, number of nearest neighbors k
output: Weighted candidate set of all objects C_i;
for (j=1;j<n;j++) **do**
  k objects smaller than the Z-value of q' are placed Z^-, and k objects larger than the Z-value of q' are placed in Z^+;// q' is an element in DS_i;
  Insert Z^-, Z^+ into the weighted kNN candidate set $C_i(q)$;
end for
add C_i to $C_i(q)$;
return $C_i(q)$

d_q is a judgement criterion based on the overall level and individual difference of q and its weighted kNN. The TOP-n objects with the highest d_q values are considered as the outliers.

In the above algorithm, the weighted data set *DS*' is obtained by formulas 1 and 2 at first, and the random

Algorithm 4 WKNN(DS_i, C, n, d, f, k) Function

1: **input:** data set DS_i, Integration of weighted kNN candidate set for all objects C, number of data object n, dimension d, number of copies f, number of nearest neighbors k;
2: **output:** The D_q value of all objects;
3: **for** (j=0;j<n;j++) **do**
4: **for** (t=0;j<2kf;t++) **do**
5: **for** (l=0;l<d;l++) **do**
6: $dis_q = \sqrt{dis_q + (q_l - ptl)^2}$;
7: //Calculate the distance between the object q and the element in $C(q)$, where p_t is the tth element in $C(q)$
8: **end for**
9: **end for**
10: Find the k objects with smallest dis_q as the weighted kNN of object q;
11: $d_q = \frac{d_{min}+d_{ave}}{2}$; //Calculate the d_q value of q and its weighted kNN, d_{min} is the minimum Euclidean distance between q and the weighted kNN, and d_{ave} is the average value
12: add d_q to D_q;
13: **end for**
14: return D_q

translation vector is used to construct the random translation copy. Then the *Z-order* curve is used to calculate the Z-value of all the objects in each copy. The Z-value is stored in the set Z_i. For example, in the fifth line of the algorithm, the function $ComputeZ(DS_i, n)$ is called to implement the construction of the set Z_i, and then the function $Candidate(DS_i, n, k)$ is called to construct the weighted kNN candidate set of the object in the DS_i. After that, $WKNN(DS_i, C, n, d, f, k)$ is called to get the weighted kNN, and calculate the distance value d_q of each object, and select the TOP-n objects with the largest distance. The TOP-n objects with the highest d_q values are considered as the outliers, where the TOP-n method was proposed by Jiawei Han in [30]. These largest distance values are regarded as outlier degrees, which can effectively measure the deviation degree of an outlier, and play an important role in the interpretation of outlier results.

The outer loop of the algorithm POD is executed f times, and the inner loop is executed n times. The insert operations of predecessor and subsequent in each copy are executed $2k$ times in total, so the algorithm can end automatically and will not fall into an infinite loop. The algorithm firstly assigns corresponding weights to different attributes of the object, then accesses k precursors and k successors of the query object q in f copies of the data set, and calculates the distance between the query object q and them. Finally,the d_q values of all objects are compared to get the TOP-n objects with the largest distance which are selected as outliers. Therefore, the time complexity of the algorithm POD is O(fk). Since the POD algorithm needs to generate f copies, in addition there are n objects in each copy, the space complexity of the algorithm POD is O(fn).

V. PARALLEL ALGORITHM DESIGN

MapReduce is a programming framework for processing large data sets on Hadoop platform, which can easily run on thousands of nodes of ordinary machines. A MapReduce task will cut the input data set into independent small data blocks, which are placed on the DataNode in parallel by the map task. The reduce task will collect the output of the map task and do the following calculation, and then output the result after the calculation.

k nearest neighbor query is to query the k data objects closest to a given object in the dataset according to the similarity measure, that is, this query step is only related to the k nearest neighbors of a given object, and has nothing to do with other objects. In weighted kNN, different attributes are set different weights to distinguish the importance of each attribute. Therefore, the outlier detection algorithm based on weighted kNN can be implemented in parallel under MapReduce framework. Each DataNode is assigned an independent data block, and the weighted kNN of the objects in these data blocks are calculated by the map task. The reduce task collects the weighted kNN on each DataNode and calculates the outlier of each object. Some objects with the largest outlier factor are regarded as outlier data. The implementation process of the parallel weighted kNN and outlier detection algorithm proposed in this paper is shown in Fig.5.

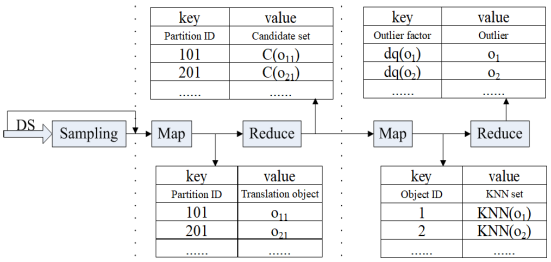


FIGURE 5. Implementation framework of MapReduce.

A. DATA PREPROCESSING

With the development of information science, the amount of data is increasing. When we measure the importance of data attributes, we need to know the distribution features of the whole data. The weight of each attribute is calculated by scanning all data objects, which is a very time-consuming and expensive operation. In order to solve this problem, we use sampling technique. The data distribution of the whole data set is replaced by the sample data set, so the attribute weights in the whole data set are replaced by the attribute weights in the sample data set. In order to apply to various data sets, we propose two sampling techniques to obtain the distribution features of the original data. If an original data set follows an uniform distribution, we randomly and evenly extract part of the data as the sample data. If the distribution features of an

original data set are unknown, we use the interval sampling method to sample. That is, we extract sample data at equal intervals from an original data set to generate its sample data set. Please note that calculating weights in sample data sets can improve the efficiency of the algorithm, but it will slightly reduce the accuracy of weights.

After the sample dataset is obtained, the information entropy of all attributes is calculated according to formula 1, and then the weight of each attribute is computed according to formula 2. The weights are saved to a file and uploaded to HDFS, so that it can be called when the data attributes are weighted in the first MapReduce job.

B. PARALLEL CONSTRUCTION OF WEIGHTED kNN CANDIDATE SETS

The first MapReduce job reads data set DS from HDFS, and its running process is divided into three steps. Firstly, the attributes of each object are weighted. Secondly, f copies of the object are obtained according to the random translation vector. Finally, the weighted kNN candidate set of each object is constructed.

In Map phase, the key-value pair <key offset, value DS> is the input information of this phase. The Map task reads the attribute weights from the output file of data preprocessing, and then assigns weight values to each data object. Because the amount of data is too large, data skew phenomenon may occur, which will cause unbalanced workload on each node and make some nodes become performance bottlenecks. Therefore, LSH data partition strategy is adopted to disperse data and balance the workload of each node. For each data object, its hash value is calculated according to the hash function shown in formula 5, and then all the hash values form a tuple, which is a bucket number corresponding to the object.

$$h_{\alpha,\beta(o)} = \lfloor \frac{\alpha \cdot \beta + \beta}{\omega} \rfloor \tag{5}$$

where o is a data object in the dataset, ω is the width value, and α is a random vector with the same dimension as o, which is generated by normal distribution. Each hash function maps a d-dimensional data object to an integer. If there are *num* hash functions, a hash table with *num* values will be formed. Each data object is mapped into *num* dimension vector by hash table, which is converted into an integer value to represent a bucket. According to the hash value, different objects are divided into different buckets. Next, f copies of data objects are constructed according to the random translation vector, in which the bucket number of each copy is the same. Finally, <key partition-number, value object-value> is output. These objects with the same key value are merged and sent to a reduced node. In the reduce phase, the Z-value of each object is calculated according to the bit crossover, in which the precursor and successor of each object are obtained by comparing the Z-value. The precursor and successor of an object constitute its weighted k nearest neighbor candidate set. Finally, <key partition-number, value kNN-candidate-set> is taken as the output of the first MapReduce job, and the output result is saved in the HDFS file which is used as the input of the second MapReduce job.

C. PARALLEL COMPUTING WEIGHTED kNN

The second MapReduce job reads the output file of the first MapReduce and calculates the final weighted k nearest neighbor according to the distances between each object and all objects in its weighted kNN candidate set. Next, an outlier factor is calculated according to the distance between each object and its weighted k-NN, and the n objects with the largest outlier factor are determined as outliers. In the Map phase, the key-value pair <key offset, value weighted kNN> is used as the input data. After these data are segmented, the distance between each object and all elements in its weighted kNN candidate set is calculated. Finally, k objects with minimum distance are determined as weighted-kNN.

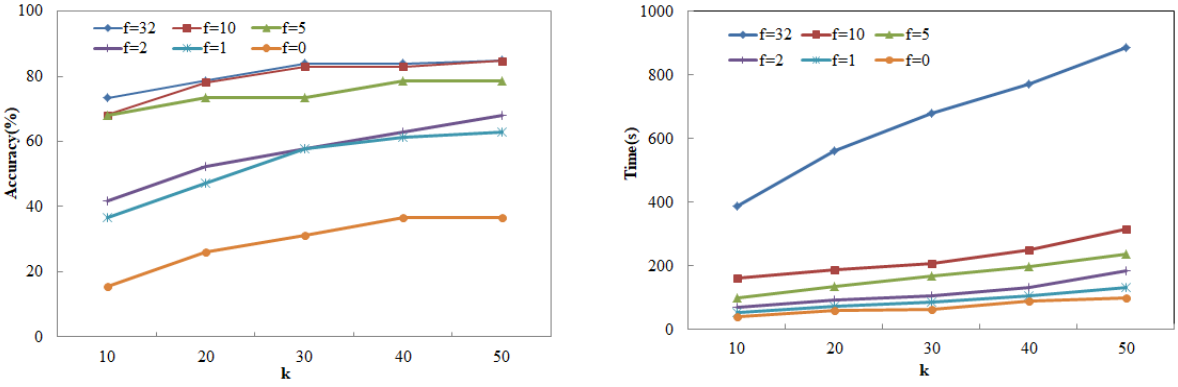
In the Reducer phase, the minimum distance d_{min} and the average distance d_{ave} of each object are calculated, and then the outlier factor d_q of each object is calculated according to formula 4. We compare the outlier factor d_q of all objects and output the N objects with the largest outlier factor. The output form is <key outlier-factor, value outlier>, in which these output objects are outliers.

VI. EXPERIMENTAL ANALYSIS

Experimental environment: Intel(R) Core(TM) i5-4570 CPU, 2G memory, Ubuntu 14.04 operating system, parallel computing platform is hadoop2.6.0, the integrated development environment is eclipse, and Java language is used to implement partition-based [20], OMAAWD [31] and POD algorithms.

We process synthetic, UCI, and celestial spectral data sets to evaluate the performance of the proposed POD algorithm.

- Synthetic data sets are generated by Microsoft Excel's random data generator. The data sets follow a standard normal distribution, and are inserted into a small amount of special data objects as outliers. These special objects follow a uniform distribution in a range between 0 and 1 accounting for 1% of the whole data set. In this paper, two groups of synthetic data sets are created. The first group is D1, D2, D3, D4, and D5, where they contain 500000 objects,and 20, 40, 60, 80 and 100 attributes, respectively. The second group is S1, S2, S3 and S4, where they contain 50 attributes, and the number of data objects are 200000, 400000, 600000 and 1000000, respectively.
- We use six UCI data sets in our experiments, including Parkinsons Telemonitoring, Drug consumption (quantified), Diabetic Retinopathy Debrecen Data Set, QSAR biodegradation, Anuran Calls (MFCCs), and EMG Physical Action Data Set. For the convenience of labelling, they are referred to as Park, Drug, Diab, QSAR, Anuran, and EMG, respectively. In Table 3, the characteristics of these data sets are summarized. The number of outliers is 1% of the amount of data, which are 59, 19, 12, 11, 72, and 98 respectively.



(a) The impact on accuracy.

(b) The impact on efficiency.

FIGURE 6. Impacts of The number of copies *f*.

TABLE 3. UCI data sets.

Dataset	Park	Drug	Diab	QSAR	Anuran	EMG
Number of objects	5875	1885	1151	1055	7195	9725
Number of attributes	26	32	20	41	22	8
Number of outliers	59	19	12	11	72	98

TABLE 4. Celestial spectral data sets.

Dataset	Star	Galaxy	Qso	LTStar
Number of objects	5365	2760	7983	6553
Number of attributes	90	90	90	90
Number of outliers	54	28	80	66

- Celestial spectral datasets are provided by the China National Astronomical Observatory [32], each of which has a total of 90 attributes. The datasets are star spectral, galaxy spectral, quasar spectral, and late-type star, respectively. In Table 4, the characteristics of celestial spectral datasets are summarized. Some high red-shift quasar spectral data are added to each spectral dataset as outliers. The number of outliers is 1% of the amount of data, which are 54, 28, 80, and 66, respectively.

A. PERFORMANCE ANALYSIS IN UCI AND CELESTIAL SPECTRAL DATASET

1) PERFORMANCE AFFECTING PARAMETERS

In this group of experiments, we use the UCI data set Drug to compare the effects of number of different copies (ie, parameter f) on the performance of the algorithm POD. The number of copies f is set to 0, 1, 2, 5, 10, and 32, respectively, where $f = 0$ represents the original data set DS.

Now we evaluate the impacts of the number of copies f on *POD*'s outlier detecting accuracy. To discriminate true outliers from false outliers (a.k.a., false negatives) in experimental results, we employ the $F1$-*measure* method [33] to quantify outlier detecting accuracy. Formally, $F1$-*measure* is denoted as follows

$$F1\text{-}measure = \frac{2 \times R \times P}{R + P} \tag{6}$$

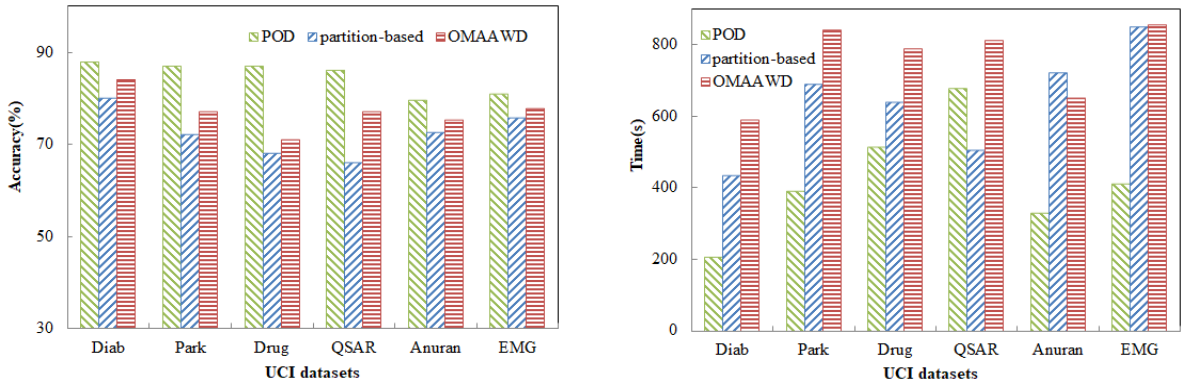
where $R = \frac{TP}{TP+FN}$ (i.e., Recall) is a function of correctly classified examples (i.e., true positives) and misclassified examples (i.e., false negatives); $P = \frac{TP}{TP+FP}$(i.e., Precision) is a function of true positives and examples misclassified as positives (i.e., false positives). In this paper, TP is the number of real outliers detected by *POD* (i.e., true positives); FP is the number of normal data that are misclassified as outliers by *POD* (i.e., false positives); and FN is the number of real outliers that are not detected by *POD* (i.e., false negatives). We derive (7) from (6) as

$$F1\text{-}measure = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{7}$$

(7) is used to evaluate *POD*'s accuracy.

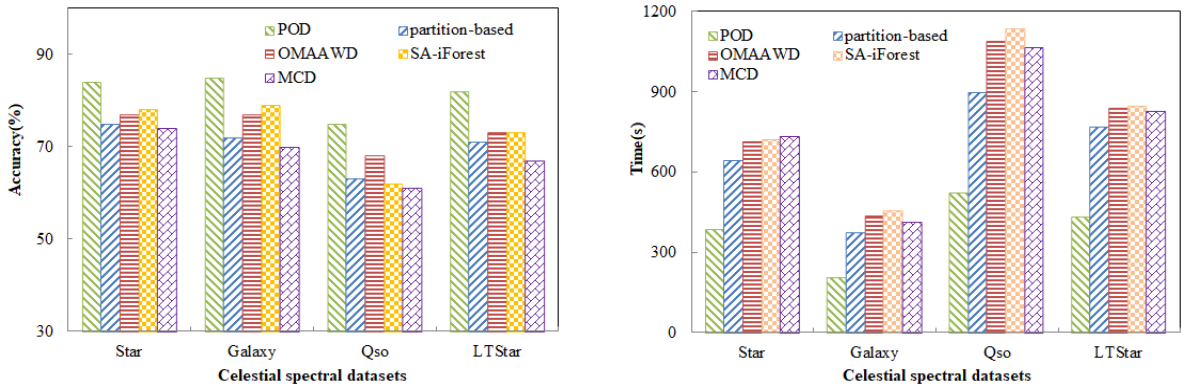
The results plotted in Fig.6(a) indicate the effect of the different f of the same data set on accuracy. In general, the accuracy improves with the increase of k for same f; the accuracy improves with the increase of f for same k, and the larger the f, the slower the acceleration rate of accuracy. For example, the difference between the number of copies of $f = 1$ and $f = 0$ is 1, but the accuracy is doubled; the difference between the number of copies of $f = 10$ and $f = 32$ is 21, and the curve of accuracy is almost completely coincident. The main reason is that as the number of replicas goes up, the number of weighted kNN candidate sets of the object grows, however the number of duplicate candidates objects is increases, thereby the acceleration rate of accuracy will slow down.

Fig. 6(b) shows the effect of the different f of the same data set on efficiency. When f is the same, the larger the k, the more the query time; when k is the same, the more the f, the more the query time. It mainly because the f is larger, the more weighted kNN candidate sets for each object, and the more objects to be processed when performing weighted kNN query, accordingly the more corresponding time-consuming; similarly, the larger the k, the more



(a) The impact on accuracy. (b) The impact on efficiency.

FIGURE 7. The comparison of accuracy and efficiency of three algorithms.



(a) The impact on accuracy. (b) The impact on efficiency.

FIGURE 8. The comparison of accuracy and efficiency of three algorithms.

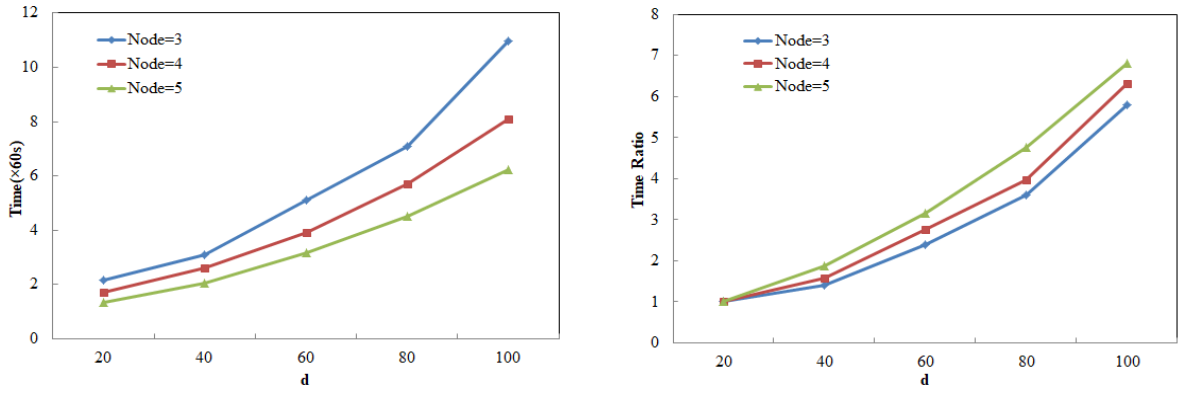
weighted kNN needs to be queried for each object, and the more time-consuming. Referring to Fig. 6(a), in some instances with high requirement for time efficiency, $f = 10$ is a good choice, and $f = 32$ is suitable for the instances of very high requirement for accuracy of outlier detection.

2) PERFORMANCE COMPARISON

In this group of experiments, we use the UCI data sets Diab, Park, Drug, and QSAR to compare the performance differences among algorithms POD, OMAAWD [31] and partition-based [20]. The algorithm OMAAWD uses information entropy to distinguish the importance of different attributes, and combines pruning techniques to query outliers. Partition-based first employ the clustering algorithm to partition the input object, and calculates the upper and lower limits of the objects in each partition, and then adopt these information to identify the partition most likely to contain the TOP-n outliers, from which calculate the outliers. Fig 7(a) demonstrates the accuracy trend of the three algorithms in different data sets. In each data set, POD is more accurate than the other two algorithms, and as the data dimension continues to increase, the result of comparison is constant. The main reason is that when performs data mapping in POD, many copies are generated through random translation. The nearest neighbors of the objects in each copy are different. After all the neighbors are integrated, they are filtered to ensure more accurate kNN are found. It is possible to cut out the real outliers and reduce the accuracy owing to OMAAWD has to prune the data set. The TOP-n points with the largest distance from their nearest neighbors are treated as outliers in Partition-based algorithms, but there may be some errors when judging objects of the same distance.

Fig. 7(b) shows the efficiency comparison of the three algorithms in different data sets. POD algorithm takes less time on the relatively low dimension datasets of Diab, Park and Drug. But in the dataset QSAR, POD takes more time than partition-based. The main reason is that the partition-based complexity is O(n), which is greatly affected by n, while POD is mainly affected by dimension d. The data set QSAR has less data than Drug, and the dimension is higher than Drug, so time-consuming of partition-based is on a downward and time-consuming of POD is on the rise.

In addition, we compare algorithms' performance among POD, OMAAWD [31], partition-based [20], SA-iForest [34],



(a) POD algorithm's running time.

(b) POD algorithm's time ratio.

FIGURE 9. Impacts of the number of attributes on the performance of POD algorithm.

and MCD [35] to future analyze efficiency and accuracy of POD algorithm in celestial spectral datasets. Fig.8 shows a similar trend as that plotted in Fig.7. In particular, the accuracy of outlier detection in quasar spectral dataset is lower than that in other celestial spectral data sets. This is because there is a strong similarity between quasar spectral and high red-shift quasar spectral in data characteristics, which leads to some high red-shift quasars being wrongly judged as quasars.

B. PERFORMANCE ANALYSIS IN SYNTHETIC DATASET

1) PERFORMANCE AFFECTING DIMENSIONS

In this group of experiments, we use the data sets D1, D2, D3, D4 and D5 to evaluate the performance of the POD algorithm, where the number of data nodes is 3, 4 and 5, respectively. In addition, the parameter k is set to 30, and the experimental results are shown in Fig.9.

Fig. 9(a) reveals that the running time of POD algorithm is increasing significantly when the dimension increases from 20 to 100. The key reason is that when the weighted kNN query is performed by *Z-order* curve in the high-dimensional data, the query time of each object will increase due to the expansion of the dimension, and the calculation amount of the weighted distance has a linear relationship with the dimension. Therefore the processing time increases linearly and tilt degree is slightly higher than linear. Another phenomenon in Fig.9(a) is that in the same dimension number, the less the number of computing nodes, the longer the running time of POD algorithm. If the number of objects remains the same, the number of data blocks in the HDFS file system remains the same. As the number of nodes increases, the number of blocks allocated on each node decreases, and the whole running time decreases proportionally. At the same time, the increase of the number of computing nodes will lead to the increase of network transmission time, which slightly weakens the scalability of the cluster.

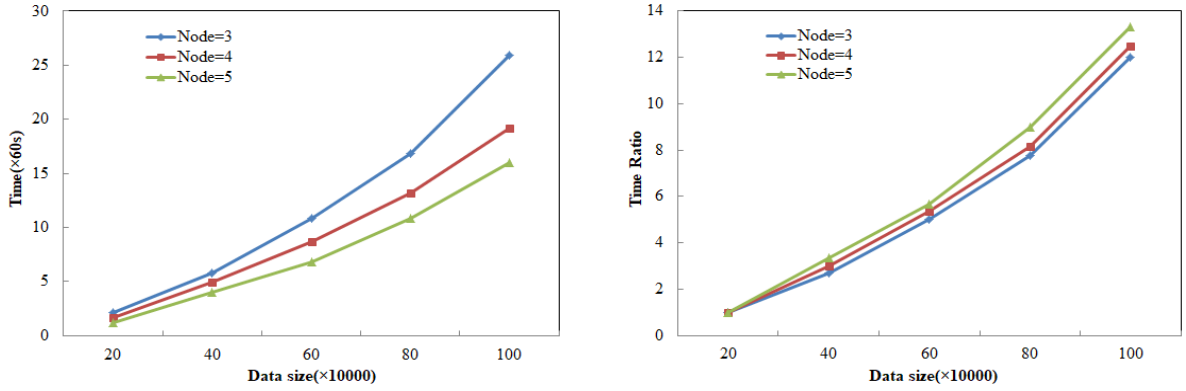
Fig.9(b) shows the time ratio change when the number of nodes are set 3, 4 and 5. With the increase of dimension, the time ratio increases continuously, and the more the number of nodes, the greater the time ratio. As a whole, the slope angle of the curve is gradually higher than that of the linear curve. With the increase of dimensions, the capacity of data set is increasing, and the number of data blocks in HDFS file system is increasing. Therefore, the shuffle cost between map and reduce is increasing, resulting in the larger data dimension, the larger time ratio.

2) EXTENSIBILITY

In this group of experiments, artificial data sets S1, S2, S3, S4 and S5 are used to evaluate the scalability of POD, in which the number of nodes is 3, 4 and 5 respectively. The experimental results are shown in Fig. 10.

Fig.10(a) shows the effect of the amount of data on the efficiency of POD algorithm when the number of computing nodes is different. With the increase of the amount of data, the algorithm costs more time. At the same time, with the increase of the number of computing nodes, the running time of the algorithm becomes less and the efficiency is higher. The number of tasks in Hadoop platform is determined by the number of data blocks allocated to computing nodes by HDFS. A large data set results in more data objects to be allocated to each node. The result is that each computing node has to deal with more tasks. In addition, as the number of objects increases, more data copies need to be built in the first MapReduce stage. Therefore, more weighted k nearest neighbor candidate sets are generated, which leads to the linear growth of outlier search with the increase of data volume. When the same amount of data remains unchanged, the number of computing nodes decreases, makes the number of data objects running on each computing node increase. The larger the load, the longer the running time, that is, the running time of the algorithm is inversely proportional to the number of nodes.

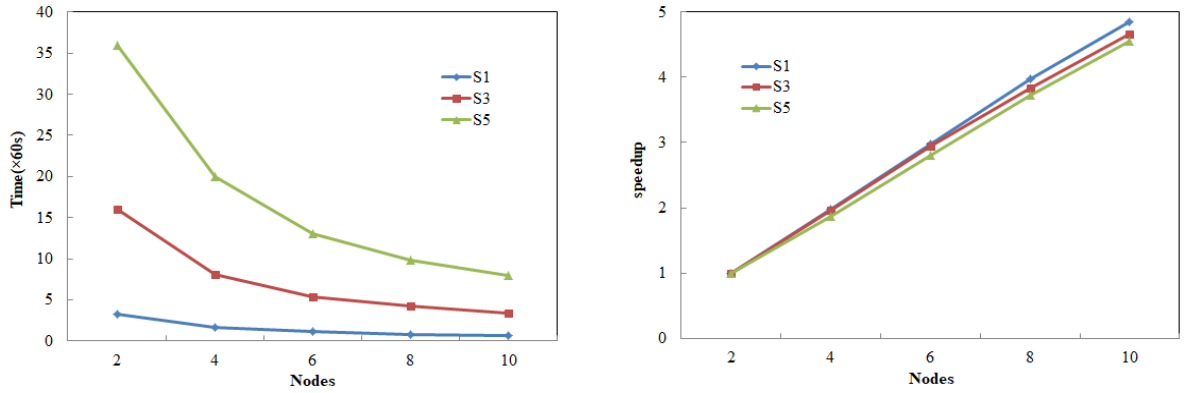
Fig.10(b) shows the trend of time ratio when the number of nodes is 3, 4 and 5. The larger the data is, the greater the time ratio is. Similarly, the more the number of nodes, the greater the time ratio. Therefore, the inclination angle of the curve is gradually higher than that of the linear curve. The reason is that the increase of data leads to more data blocks in HDFS file system and more data objects allocated to each computing



(a) POD algorithm's running time.

(b) POD algorithm's time ratio.

FIGURE 10. The extensibility of the POD algorithm in various data size.



(a) POD algorithm's running time.

(b) POD algorithm's speedup.

FIGURE 11. The scalability of the POD algorithm in various data nodes.

node. After the first map stage, the shuffle operation cost increases, which leads to the increase of the whole distributed runtime and time ratio. At the same time, with the increase of the number of computing nodes, the network transmission time is increased, which leads to more additional time-consuming. Therefore, the time ratio of POD algorithm is significantly increased.

3) SCALABILITY

In this experiment, we used artificial data sets S1, S3 and S5 with 200000, 600000 and 1000000 objects respectively. Each data object contains 50 attributes, and the experimental results are shown in Fig.11. Fig.11(a) shows the impact of the number of computing nodes on mining efficiency. The less the number of computing nodes, the more time is consumed for a large data set. The reasons are as follows: firstly, the calculation of outlier factor of each data object can be parallelized; secondly, the calculation of Z-value is not affected by the number of nodes; finally, the construction of the corresponding weighted k nearest neighbor candidate set is also independent of the number of nodes. Therefore, the running time of the algorithm is linear with the number of data objects. At the same time, with the increase of computing nodes, the amount of network transmission will increase, and the time consumed in shuffle phase will be affected. The parallel effect of the algorithm is weakened, so the change tends to be slow.

Fig.11(b) shows the trend of speedup in various data sets when the number of nodes changes. The more the number of computing nodes, the greater the speedup; and the less the amount of data, the greater the speedup. The overall trend of the curve is gradually lower than that of the linear curve. When the amount of data is small, the number of data blocks in HDFS file system will be reduced, which will lead to less time to query the weighted k nearest neighbors of all objects. In addition, cluster I / O takes a small amount of time, which has little impact on the speedup. Therefore, the speedup ratio is increasing. At the same time, the more the number of cluster computing nodes, the shorter the running time of the algorithm. Theoretically, the speedup ratio should be linear with the number of computing nodes, but in the actual operation process, the increase of the number of computing nodes brings the increase of network transmission, so the effect of parallelization will gradually decrease.

VII. CONCLUSION AND FUTURE WORK

In this paper, we propose a parallel outlier detection algorithm POD based on k nearest neighbours. The algorithm uses information entropy to calculate each attribute weight, and then uses Z-order curve to encode high-dimensional data into Z-value. After getting the weighted k nearest neighbor, POD algorithm detect outlier by using minimum distance and average distance between each object and its weighted kNN. In order to better adapt to the needs of massive data, the future work is that our proposed POD algorithm is implemented in clusters with more nodes.

REFERENCES

[1] X. Yi, R. Paulet, E. Bertino, and V. Varadharajan, "Practical approximate K nearest neighbor queries with location and query privacy," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1546–1559, Jun. 2016.

[2] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query processing in spatial network databases," in *Proc. 29th VLDB Conf.*, San Francisco, CA, USA, 2003, pp. 802–813.

[3] D. A. Adeniyi, Z. Wei, and Y. Yongquan, "Automated Web usage data mining and recommendation system using K-nearest neighbor (KNN) classification method," *Appl. Comput. Informat.*, vol. 12, no. 1, pp. 90–108, Jan. 2016.

[4] L. P. Zhang, H. D. Jing, L. I. Song, and H. Y. Cui, "K nearest neighbor query based on Voronoi diagram for obstructed spaces," *Comput. Sci.*, vol. 43, no. 5, pp. 174–178, 2016.

[5] J. Cai, H. Wei, H. Yang, and X. Zhao, "A novel clustering algorithm based on DPC and PSO," *IEEE Access*, vol. 8, pp. 88200–88214, 2020.

[6] X. Zhao, Y. Rao, J. Cai, and W. Ma, "Abnormal trajectory detection based on a sparse subgraph," *IEEE Access*, vol. 8, pp. 29987–30000, 2020.

[7] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest neighbor queries," *ACM SIGMOD Rec.*, vol. 24, no. 2, pp. 71–79, Jun. 1995.

[8] Y. Chen, L. Zhou, Y. Tang, J. P. Singh, N. Bouguila, C. Wang, H. Wang, and J. Du, "Fast neighbor search by using revised K-D tree," *Inf. Sci.*, vol. 472, pp. 145–162, Jan. 2019.

[9] Y. Yang, J. Cai, H. Yang, J. Zhang, and X. Zhao, "TAD: A trajectory clustering algorithm based on spatial-temporal density analysis," *Expert Syst. Appl.*, vol. 139, Jan. 2020, Art. no. 112846.

[10] Y. Li, J. Cai, H. Yang, J. Zhang, and X. Zhao, "A novel algorithm for initial cluster center selection," *IEEE Access*, vol. 7, pp. 74683–74693, 2019.

[11] J. Roshanian, S. Yazdani, and M. Ebrahimi, "Star identification based on euclidean distance transform, Voronoi tessellation, and K-nearest neighbor classification," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 52, no. 6, pp. 2940–2949, Dec. 2016.

[12] S. Yang, Z. He, and Y.-P.-P. Chen, "Workload-based ordering of multi-dimensional data," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 3, pp. 831–844, Mar. 2016.

[13] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. 30th Annu. ACM Symp. Theory Comput. (STOC)*, New York, NY, USA, Aug. 1998, pp. 604–613.

[14] K.-I. Lin and C. Yang, "The ANN-tree: An index for efficient approximate nearest neighbor search," in *Proc. 7th Int. Conf. Database Syst. Adv. Appl. (DASFAA)*, Apr. 2001, pp. 174–181.

[15] S. Zhang, D. Cheng, Z. Deng, M. Zong, and X. Deng, "A novel KNN algorithm with data-driven K parameter computation," *Pattern Recognit. Lett.*, vol. 109, pp. 44–54, Jul. 2018.

[16] J. Gou, W. Qiu, Z. Yi, Y. Xu, Q. Mao, and Y. Zhan, "A local mean representation-based K-nearest neighbor classifier," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 3, p. 1–25, 2019.

[17] J. Gou, W. Qiu, Z. Yi, X. Shen, Y. Zhan, and W. Ou, "Locality constrained representation-based K-nearest neighbor classification," *Knowl.-Based Syst.*, vol. 167, pp. 38–52, Mar. 2019.

[18] R. Agrawal, "Integrated parallel K-nearest neighbor algorithm," in *Proc. 2nd Int. Conf. SCI*, vol. 1, 2019, pp. 479–486.

[19] X. Zhao, J. Zhang, X. Qin, J. Cai, and Y. Ma, "Parallel mining of contextual outlier using sparse subspace," *Expert Syst. Appl.*, vol. 126, pp. 158–170, Jul. 2019.

[20] J. Zhang, X. Yu, Y. Xun, S. Zhang, and X. Qin, "Scalable mining of contextual outliers using relevant subspace," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 3, pp. 988–1002, Mar. 2020.

[21] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA, 2000, pp. 427–438.

[22] F. Angiulli and C. Pizzuti, "Outlier mining in large high-dimensional data sets," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 2, pp. 203–215, Feb. 2005.

[23] R. Xiao, J. Su, X. Du, J. Jiang, X. Lin, and L. Lin, "SFAD: Toward effective anomaly detection based on session feature similarity," *Knowl.-Based Syst.*, vol. 165, pp. 149–156, Feb. 2019.

[24] Y. Liu, Z. Li, C. Zhou, Y. Jiang, J. Sun, M. Wang, and X. He, "Generative adversarial active learning for unsupervised outlier detection," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 8, pp. 1517–1528, Aug. 2020.

[25] D. Chakraborty, V. Narayanan, and A. Ghosh, "Integration of deep feature extraction and ensemble learning for outlier detection," *Pattern Recognit.*, vol. 89, pp. 161–171, May 2019.

[26] G. Feng, Z. Li, W. Zhou, and S. Dong, "Entropy-based outlier detection using spark," *Cluster Comput.*, vol. 23, no. 2, pp. 409–419, Jun. 2020.

[27] J. Li, J. Zhang, X. Qin, and Y. Xun, "Feature grouping-based parallel outlier mining of categorical data using spark," *Inf. Sci.*, vol. 504, pp. 1–19, Dec. 2019.

[28] X. Zhang, C. Mei, D. Chen, and J. Li, "Feature selection in mixed data: A method using a novel fuzzy rough set-based information entropy," *Pattern Recognit.*, vol. 56, pp. 1–15, Aug. 2016.

[29] D. Liu, P. Hu, T. Li, and C. Jiang, "An approach for attribute weights acquisition based on rough sets theory and information gain," *Int. J. Comput. Intell. Syst.*, vol. 33, no. 10, pp. 1296–1302, 2007.

[30] W. Jin, A. K. H. Tung, and J. Han, "Mining top-n local outliers in large databases," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2001, pp. 293–298.

[31] S. J. Lou, J. F. Zhang, and A. Q. Liu, "A outlier mining algorithm based on p weights," *J. Chin. Comput. Syst.*, vol. 35, no. 1, pp. 55–59, 2014.

[32] Y. Ma, J.-F. Zhang, J.-H. Cai, H.-F. Yang, and X.-J. Zhao, "Parallel extraction and analysis of abnormal features of QSO spectra based on sparse subspace," *Spectrosc. Spectral Anal.*, vol. 41, no. 4, pp. 1086–1091, 2021.

[33] D. M. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, 2011.

[34] D. Xu, Y. Wang, Y. Meng, and Z. Zhang, "An improved data anomaly detection method based on isolation forest," in *Proc. 10th Int. Symp. Comput. Intell. Design (ISCID)*, Dec. 2017, pp. 287–291.

[35] C. Fauconnier and G. Haesbroeck, "Outliers detection with the minimum covariance determinant estimator in practice," *Stat. Methodol.*, vol. 6, no. 4, pp. 363–379, Jul. 2009.

YANG MA received the M.S. degree in computer science and technology from the Taiyuan University of Science and Technology, Taiyuan, China, in 2009, where she is currently pursuing the Ph.D. degree in big data and intelligent manufacturing. Her current research interests include data mining and artificial intelligence.

XUJUN ZHAO received the M.S. degree in computer science and technology and the Ph.D. degree from the Taiyuan University of Science and Technology (TYUST). He is currently a Professor with the School of Computer Science and Technology, TYUST. His research interests include data mining and parallel computing. He is a Member of the China Computer Federation (CCF).