IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 9, 2020, accepted June 28, 2020, date of publication July 3, 2020, date of current version July 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3007028

Projection Based Large Scale High-Dimensional Data Similarity Join Using MapReduce Framework

YOUZHONG MA[1,2], (Member, IEEE), RUILING ZHANG[1], ZHANYOU CUI[1], AND CHUNJIE LIN[1]
[1]School of Information and Technology, Luoyang Normal University, Luoyang 471934, China
[2]Henan Key Laboratory for Big Data Processing and Analytics of Electronic Commerce, Luoyang 471934, China

Corresponding author: Youzhong Ma (ma_youzhong@163.com)

This work was supported in part by the Training Plan for Young Backbone Teachers of Colleges and Universities in Henan under Grant 2017GGJS134, in part by the Science and Technology Research Plan Project of Henan Province under Grant 202102210357, in part by the Science and Technology Opening Up Cooperation Project of Henan Province under Grant 172106000077 and Grant 152106000048, in part by the National Natural Science Foundation of China under Grant 61602231, in part by the Outstanding Talents of Scientific and Technological Innovation in Henan under Grant 184200510011, and in part by the National Key Research and Development Program under Grant 2016YFE0104600.

ABSTRACT Similarity join has been widely used in many data analysis and data mining applications, we mainly focus on the scalability and performance problem of similarity join query on massive high-dimensional data set. *p*-stable distribution based projection scheme can implement dimension reduction effectively. Three novel approaches based on projection scheme are proposed to deal with massive high-dimensional data similarity join problem: Single projection method, Multiple projection method and Projection space partitioning method. Comprehensive experimental tests were performed to evaluate the performance of the above approaches. The experimental results show that the proposed approaches in this paper have good performance and scalability.

INDEX TERMS Similarity join, MapReduce framework, high-dimensional data, *p*-stable distribution, multiple projections.

I. INTRODUCTION

With the development of data acquisition technology and data acquisition equipment, data size, data precision and data dimension are increasing rapidly in an unprecedented way.The dimensions of many types of data can reach thousands or ten thousands of dimensions, such as image, video, trajectory, time series and so on. High-dimensional data similarity join can figure out all the similar data pairs whose distance is not bigger than the predefined distance threshold from the massive high-dimensional data set, which plays an important role in many fields, such as image clustering, document de-duplication, similarity video detection, etc.

The calculation cost of the similarity join on large scale high-dimensional data is always very expensive. With the increasing of data size and dimensionality, the traditional centralized processing method and index-based algorithm can no longer satisfy the performance requirements.

MapReduce [1] was first proposed by Google as a distributed and parallel computing model with high scalability, fault tolerance and high availability which is used to deal with massive data analysis and processing attracting more and more attention from academia and industry. We try to deal with the similarity join problem on massive high-dimensional data by using MapReduce framework and provide the following contributions:

- Multiple projections based filtering scheme was proposed which can make sure that the recall and the filtering effect are both relative higher;
- Three novel approaches based on projection scheme were proposed which can deal with massive high-dimensional data similarity join problem efficiently: Single projection method, Multiple projection method and Projection space partitioning method.
- Detailed experiments were performed to evaluate the performance of the proposed approaches in this paper, the results show that our proposed methods have better performance and scalability compared with other existing methods.

This paper is organized as the following: Section 2 makes a detailed survey about the related works on similarity

The associate editor coordinating the review of this manuscript and approving it for publication was Ali Kashif Bashir.

join, and analyzes their advantages and disadvantages; The third section gives the definition of high-dimensional data similarity join, introduces some relevant basic knowledge and proves relevant theorems; Three novel similarity join algorithms are proposed respectively in section 4, 5 and 6 which are Single Projection based Similarity Join Algorithm Using MapReduce, Multiple Projections based Similarity Join Algorithm Using MapReduce, Projection Space Partitioning based Similarity Join Algorithm Using MapReduce; Section 7 conducts comprehensive experiments; In Section 8, some conclusions and expectations about the work are made.

II. RELATED WORKS

Similarity join is an important operation which is widely used in many applications, the scholars have conducted comprehensive research works on this problem. Pang *et al.* [2], Lin and Wang [3] and Yu *et al.* [4] mainly review the research works of similarity join in centralized environment. In view of the performance and scalability problems faced by large scale data similarity join, some research works attempted to use MapReduce framework to solve them. Pang *et al.* [5] summarizes the works of massive data similarity join based on MapReduce framework, Silva *et al.* [6] and Kimmett *et al.* [7] make experimental analysis and comparison of the typical similarity join algorithms based on MapReduce framework. Similarity join problem can be divided into the following categories according to the different types of data processed: set similarity join, vector similarity join, spatial data similarity join, probabilistic data similarity join, string similarity join and graph data similarity join.

A. SET SIMILARITY JOIN

Lin [8] firstly exploited the similarity join problems based on MapReduce framework and proposed brute force algorithm and index-based algorithm. Each set pair needs to be compared once according to the brute force algorithm because it didn't adopt any filtering scheme. Index-based algorithm can achieve a certain degree of filtering, but the filtering effect is not ideal and there are many duplicated calculations. Vernica *et al.* [9] proposed a prefix filtering based massive set similarity join approach. The improvement of literature [10] compared to literature [9] is that besides prefix filtering scheme, literature [10] also proposed length filtering scheme, so the filtering effect is further enhanced. Rong *et al.* [11] proposed a multiple prefix filtering technique which can further reduce the number of the candidate pairs. A cost model was also proposed to decide the prefix number. The scheme based on prefix filtering technology also has some shortcomings. Firstly, network communication cost is relative high: each set has to be replicated many times, the number of replications equals the length of the prefix, so for a longer set, the data replication rate is too high, which will lead to a higher network communication cost; secondly, there exist too many duplicated comparisons: for any two sets, if there are k common items in their prefix, the comparison will be repeated k times.

Elsayed *et al.* [12] made full use of the characteristics of MapReduce framework and the structure of Word-Count MapReduce program, proposed a document similarity join approach, which can effectively deal with the problem of duplicated comparisons. Literature [13] extended the types of data objects so that it can process set, multiple set and vector. Similarity measures can be inner product, cosine similarity and Jacquard similarity. Both of the solutions can effectively avoid duplicated comparisons, do not need to transmit the document itself, only transmit the corresponding weight of each word, which can greatly reduce the cost of network communication. However, there still exist some limitations in this kind of scheme. Any two documents containing only one common element need to be compared once, and the filtering function of similarity threshold and prefix are not utilized, so many unnecessary candidate pairs will be generated. Baraglia *et al.* [14] proposed a hybrid solution by combining the advantages of prefix filtering scheme with Word-Count-Like scheme. This solution can effectively reduce the number of candidate pairs and avoid duplicated computation. Each pair is calculated only once, but additional data transmission is needed.

Rong *et al.* [15] and Deng *et al.* [16] proposed partition-based similarity join approached for set data. Based on the traditional Locality Sensitive Hashing technology, PLSH [17] proposed a new banding technique using flexible thresholds, which can greatly reduce the number of false positive examples, improve the computational efficiency, and achieve the balance between false positive examples and false negative examples. Amagata *et al.* [18] proposed a Local-Index-based dynamic set kNN selfjoin approach(LI-DSN-Join) which can deal with the KNN Join problem on dynamic set more efficiently. Bellas and Gounaris [19] conducted a comprehensive presentation and comparative evaluation of GPU accelerated set similarity joins which provided a good reference for other following related works.

B. VECTOR SIMILARITY JOIN

Luo *et al.* [20] proposed a novel dimension reduction scheme called Basic Dimension Aggregation Approximation(abbreviated as BDAA) which was motivated by Pairwise Aggregation Approximation(abbreviated as PAA), the distance of the DAA representations is the lower bound of the original distance of the vectors, so DAA can prune unqualified pairs without calculating their original distances. Luo *et al.* [20] also designed a parallel similarity join algorithm based on MapReduce framework using BDAA which can filter effectively at a lower cost, and the time complexity is still $O(n^2)$, that is, any two vectors need to be compared once in low-dimensional space.

Seidl *et al.* [21] proposed a massive vector similarity join approach by using grid partitioning. This method has good parallel characteristics and can be easily implemented in MapReduce framework. The disadvantage of this method is that it is only suitable for the situation of low dimension. Once the dimension is high, its performance will degrade severely.

Fries *et al.* [22] proposed a novel MapReduce based similarity join method called PHiDJ which can improve the similarity join speed through grouping dimensions and variable length grid dividing. Seidl *et al.* [21] adopted uniform gird dividing method (equal width), while Fries *et al.* [22] adopted variable length grid dividing method, which has better adaptability and filtering effect. Another similarity join algorithm called *MRSJ_IDS* which can support incremental data sets was proposed in [23].

SAX-Based HDSJ [24] conducted dimension reduction for high-dimensional vectors using Piecewise Aggregate Approximation technique, the original high-dimensional vector can be converted into PAA vector.The PAA vector is converted into SAX string by using Symbolic Aggregate Approximation(SAX) which can be used to filter effectively. Ma *et al.* [25] proposed a multi-PAA based similarity join approach called MP-V-SJQ which can further increase the filtering effect and reduce the filtering cost on the basis of SAX-Based HDSJ [24]. In order to reduce unnecessary comparisons and achieve load balancing among computing nodes, Grid-Based SJ [26] proposed a similarity join approach based on dynamic grid partition.

Zhang *et al.* [27] were the first to study the problem of KNN join based on MapReduce framework. In order to reduce the cost of the comparisons and network transimission,Zhang *et al.* [27] proposed a novel KNN join method called zKNNJ which can return approximate results. Lu *et al.* [28] proposed an exact KNN similarity join approach, which partitions data mainly based on the Voronoi Diagram. Dai and Ding [29] proposed exact KNN simialrity join algorithm and the approximate KNN similarity join algorithm based on the nested loop join framework.

Kim and Shim [30] and Ma and Ci [31] proposed Top-k similarity join solutions respectively for massive high-dimensional vectors using MapReduce framework. Chen *et al.* [32] proposed a distance based on LSH for high-dimensional data, and converted the distance based on LSH into hamming distance of high-dimensional data signature. On this basis, it designed a top-k similarity join algorithm using Spark. Compared with Hadoop based solutions, Chen *et al.* [32] has faster computing speed and better scalability. Rong *et al.* [33] proposed a new similarity join algorithm called symbolic aggregation and vertical decomposition(SAVD) using Spark.

C. SPATIAL DATA SIMILARITY JOIN

Zhang *et al.* [34] did some research works on spatial data similarity join problem and proposed Spatial Join with MapReduce(SJMR) algorithm which can divide the data uniformly. A method based on pivot point is designed to ensure that a pair of spatial data can only be compared once at most. Liu *et al.* [35] proposed a novel Top-k spatial join algorithm using MapReduce(TKSJMR) which can obtain the k spatial objects with the largest overlapping number with other spatial objects. Liu *et al.* [36] proposed a Map-Reduce-Filter-Merge (MRFM) method under MapReduce framework which can solve the problem of parallel spatial join aggregation under non-index conditions. Liu *et al.* [37] proposed the parallel R-tree index construction method based on MapReduce framework, and then proposed the KNN similarity join algorithm based on MapReduce by using R-tree index. A novel ''controllable-replication'' framework for spatial join problems was proposed by Gupta *et al.* [38] which can reduce the network transmission costs between cluster nodes, and deal with the spatial join problems effectively based on ''overlap'' and ''inclusion'' predicates. Zhang *et al.* [39] studied the spatial keyword join query problem under MapReduce framework, proposed the spatial text object filtering algorithm based on the combination of prefix filtering and grid partitioning technology, and proposed two optimization methods which can improve the performance of spatial keyword join query. Dan *et al.* [40] mainly focused on spatial-temporal trajectory similarity join problem and proposed a novel two-level grid index which takes both spatial and temporal information into account. Zhu *et al.* [41] were the first to exploit the Spatial Visual Similarity Join problem for Geo-Multimedia aiming to find similar geo-image pairs in both aspects of geo-location and visual content. Wan *et al.* [42] proposed hierarchical indexing structures and Voronoi-based methods to deal with spatial range query which maybe useful to solve the spatial data similarity join problem.

D. SIMILARITY JOIN ON OTHER DATA TYPES

Lei *et al.* [43] and Huang *et al.* [44] proposed similarity join algorithms based on EMD (Earth Mover's Distance) Distance, mainly aiming at dealing with large-scale probabilistic data. Ma and Meng [45] proposed two parallel similarity join methods based on MapReduce framework for large-scale probabilistic set data: Map side filtering based similarity join and Reduce side filtering based similarity join.

Rheinlander and Leser [46], Deng *et al.* [47], Lin *et al.* [48] and Li *et al.* [49] mainly conducted research works on scalable similarity join problem on massive string data based on MapReduce framework. Rheinlnder and Leser [46] proposed a new index structure called PeARL based on trie tree structure with edit distance as the similarity measure between strings. Deng *et al.* [47] mainly extended the signature mechanism based on partition to support string join based on set similarity measure(Jaccard similarity measure). Lin *et al.* [48] extended PassJoin [49] algorithm, proposed a faster algorithm PassJoinK, and combined with MapReduce framework, parallelized PassJoinK algorithm, proposed a new algorithm called PassJoinKMRS which can deal with scalable string similarity join problem.

In order to deial with massive graph similarity similarity join problem, Pang *et al.* [50] proposed a scalable prefix filtering scheme which can reduce the number of comparisons. Based on MapReduce framework, an extensible graph data similarity join algorithm was designed. Chen *et al.* [51] mainly studied graph similarity join problem based on edit distance. The author mainly proposed an algorithm called

MGSJoin based on "filtering and verification" mechanism, adopted bloom filter technology to reduce redundant computing and network transmission cost, and integrated multi-way join strategy to enhance the efficiency of the verification stage. Zhang *et al.* [52] did some research works on large scale RDF data similarity join problem. The similarity join on the uncertain graph database usually has more practical application value and has higher time complexity compared with the certain graph database. Miu *et al.* [53] and Miu and Wang [54] have done some research works on the similarity join problems on the uncertain graph database using MapReduce.

In recent years, many other research works have been done on different similarity join problems, such as similarity join on time series [55], [56], approximate KNN similarity join [57], [58], similarity join on data stream [59].

III. PRELIMINARIES

A. NOTATIONS

Table 1 describes the notations which are used in this paper:

TABLE 1. Notations.

Notations	Descriptions
n_1, n_2	the cardinality of the data sets.
ϵ	the predefined Euclidean distance threshold.
d	the data point's dimensionality.
$dist(v_1, v_2)$	the Euclidean distance between v_1 and v_2.
$g(v)$	$g(v) = a \cdot v$, each element of a is a random variable satisfied with *p-stable distribution*.
$\pi_m(v)$	$\pi_m(v) = \langle g_1(v), g_2(v), \ldots, g_m(v)\rangle$.
$\Delta_m(v_1, v_2)$	the distance between $\pi_m(v_1)$ and $\pi_m(v_2)$ that is $\Delta_m(v_1, v_2) = dist(\pi_m(v_1), \pi_m(v_2))$.
$\chi^2(m)$	Chi-squared distribution, the degree of freedom is m .

B. PROBLEM DEFINITION

High-Dimensional Data Similarity Join Query is defined as the following:

Definition 1 (High-Dimensional Data Similarity Join Query(HDSJ)): Given two data sets $\mathbf{Q}$ *and* $\mathbf{R}$, $\mathbf{Q} = \{q_1, q_2, \ldots, q_{n_1}\}$, $\mathbf{R} = \{r_1, r_2, \ldots, r_{n_2}\}$, q_i *is* i_{th} *data point from* $\mathbf{Q}$, $q_i = \langle q_{i1}, q_{i2}, \ldots, q_{id}\rangle$, r_j *is* j_{th} *data point from* $\mathbf{R}$, $r_j = \langle r_{j1}, r_{j2}, \ldots, r_{jd}\rangle$, q_i *and* r_j *are d-dimensional vector, that is* $q_i \in \Re^d$, $r_j \in \Re^d$. $dist(.)$ *represents the Euclidean distance and* ϵ *refers to the distance threshold, then the high-dimensional data similarity join query on* $\mathbf{Q}$ *and* $\mathbf{R}$ *can figure out all the similar data pairs whose distance is less than or equal to the predefined threshold* ϵ. *That can be recorded as:* $HDSJ(\mathbf{Q} \bowtie \mathbf{R}) = \{\langle q_i, r_j\rangle | q_i \in \mathbf{Q}, r_j \in \mathbf{R}, dist(q_i, r_j) \leq \epsilon\}$. *The Euclidean distance between* q_i *and* r_j *can be calculated as the following*:

$$dist(q_i, r_j) = \sqrt{\sum_{m=1}^{d}(q_{im} - r_{jm})^2} \tag{1}$$

C. THEOREMS

Theorem 1: Given two d-dimensional vectors v_1 *and* v_2, *then* $g(v_1) - g(v_2) \sim \mathbf{N}(0, dist^2(v_1, v_2))$.

Theorem 2: Given two d-dimensional vectors v_1 *and* v_2, $v_1, v_2 \in \Re^d$, $\frac{\Delta_m^2(v_1,v_2)}{dist^2(v_1,v_2)} \sim \chi^2(m)$.

Theorem 1, 2 can be derived based on the properties of *p-stable distribution* and *Chi-squared* distribution, and they have been proofed in our previous work [60]. As described in Table 1, $g(v) = a \cdot v$, each element of a is a random variable satisfied with *p-stable distribution*. We can project d-dimensional vector v into m-dimensional space by using $\pi_m(v) = \langle g_1(v), g_2(v), \ldots, g_m(v)\rangle$ so as to achieve the goal of dimension reduction.

Theorem 3: If $\Delta_m(v_1, v_2) > k\epsilon$, *then the probability that* $dist(v_1, v_2)$ *will be bigger than* ϵ *is bigger than* $1 - P(\chi^2 > k^2)$, *that is:* $P(dist(v_1, v_2) > \epsilon | \Delta_m(v_1, v_2) > k\epsilon) > 1 - P(\chi^2 > k^2)$.

Proof:

$$P(dist(v_1, v_2) > \epsilon | \Delta_m(v_1, v_2) > k\epsilon)$$
$$= \frac{P(dist(v_1, v_2) > \epsilon \text{ and } \Delta_m(v_1, v_2) > k\epsilon)}{P(\Delta_m(v_1, v_2) > k\epsilon)}$$
$$\because \Delta_m(v_1, v_2) > k\epsilon \text{ is given.}$$
$$\therefore P(\Delta_m(v_1, v_2) > k\epsilon) = 1$$
$$\because dist(v_1, v_2) \geq 0$$
$$\therefore P(dist(v_1, v_2) > \epsilon | \Delta_m(v_1, v_2) > k\epsilon)$$
$$= P(dist(v_1, v_2) > \epsilon \text{ and } \Delta_m(v_1, v_2) > k\epsilon)$$
$$= P(dist^2(v_1, v_2) > \epsilon^2 \text{ and } \Delta_m(v_1, v_2) > k\epsilon)$$
$$= P\left(\frac{1}{dist^2(v_1, v_2)} < \frac{1}{\epsilon^2} \text{ and } \Delta_m(v_1, v_2) > k\epsilon\right)$$
$$\because \Delta_m(v_1, v_2) \geq 0$$
$$\therefore P(dist(v_1, v_2) > \epsilon | \Delta_m(v_1, v_2) > k\epsilon)$$
$$= P\left(\frac{\Delta_m^2(v_1, v_2)}{dist^2(v_1, v_2)} \leq \frac{\Delta_m^2(v_1, v_2)}{\epsilon^2} \text{ and } \Delta_m(v_1, v_2) > k\epsilon\right)$$
$$= 1 - P\left(\frac{\Delta_m^2(v_1, v_2)}{dist^2(v_1, v_2)} > \frac{\Delta_m^2(v_1, v_2)}{\epsilon^2} \text{ and } \Delta_m(v_1, v_2) > k\epsilon\right)$$
$$\because \Delta_m(v_1, v_2) > m\epsilon \text{ and } \frac{\Delta_m^2(v_1, v_2)}{dist^2(v_1, v_2)} \sim \chi^2(k)$$
$$\therefore P\left(\frac{\Delta_m^2(v_1, v_2)}{dist^2(v_1, v_2)} \text{ and } \Delta_m(v_1, v_2) > k\epsilon\right)$$
$$< P\left(\chi^2 > \frac{m^2\epsilon^2}{\epsilon^2}\right) = P\left(\chi^2 > k^2\right)$$
$$\Rightarrow 1 - P\left(\frac{\Delta_m^2(v_1, v_2)}{dist^2(v_1, v_2)} \text{ and } \Delta_m(v_1, v_2) > k\epsilon\right)$$

$$> 1 - P\left(\chi^2 > k^2\right)$$
$$\Rightarrow P(dist(v_1, v_2) > \epsilon | \Delta_m(v_1, v_2) > k\epsilon) > 1 - P(\chi^2 > k^2)$$

■

According to the above theorems, we can find that if the Euclidean distance between $\pi_m(v_1)$ and $\pi_m(v_2)$ is grater than $k\epsilon$, then the Euclidean distance between v_1 and v_2 will be greater than ϵ with the probability which is greater than $1 - P(\chi^2 > m^2)$. Based on this property, we can filter out the dissimilar vectors at low computation cost by computing the low dimensional space distance.

Theorem 4: If $dist(v_1, v_2) > \epsilon$, then the probability that $\Delta_m(v_1, v_2)$ will be bigger than $k\epsilon$ is bigger than $P(\chi^2 > k^2)$, that is: $P(\Delta_m(v_1, v_2) > k\epsilon | dist(v_1, v_2) > \epsilon) > P(\chi^2 > k^2)$.

Proof:

$$\because \Delta_m(v_1, v_2) \geq 0$$
$$\therefore P(\Delta_m(v_1, v_2) > k\epsilon | dist(v_1, v_2 > \epsilon))$$
$$= P(\Delta_m^2(v_1, v_2) > k^2\epsilon^2 | dist(v_1, v_2) > \epsilon)$$
$$= P\left(\frac{\Delta_m^2(v_1, v_2)}{dist^2(v_1, v_2)} > \frac{k^2\epsilon^2}{dist^2(v_1, v_2)} | dist(v_1, v_2) > \epsilon\right)$$
$$= \frac{P\left(\frac{\Delta_m^2(v_1, v_2)}{dist^2(v_1, v_2)} > \frac{k^2\epsilon^2}{dist^2(v_1, v_2)} \text{ and } dist(v_1, v_2) > \epsilon\right)}{P(dist(v_1, v_2) > \epsilon)}$$
$$\because P(dist(v_1, v_2) > \epsilon) = 1$$
$$\therefore P(\Delta_m(v_1, v_2) > k\epsilon | dist(v_1, v_2 > \epsilon))$$
$$= P\left(\frac{\Delta_m^2(v_1, v_2)}{dist^2(v_1, v_2)} > \frac{k^2\epsilon^2}{dist^2(v_1, v_2)} \text{ and } dist(v_1, v_2) > \epsilon\right)$$
$$\because \frac{\Delta_m^2(v_1, v_2)}{dist^2(v_1, v_2)} \sim \chi^2(m)$$
$$\therefore P(\Delta_m(v_1, v_2) > k\epsilon | dist(v_1, v_2 > \epsilon))$$
$$> P\left(\frac{\Delta_m^2(v_1, v_2)}{dist^2(v_1, v_2)} > \frac{k^2\epsilon^2}{\epsilon^2}\right)$$
$$= P\left(\frac{\Delta_m^2(v_1, v_2)}{dist^2(v_1, v_2)} > k^2\right)$$
$$= P\left(\chi^2 > k^2\right)$$
$$\Rightarrow P(\Delta_m(v_1, v_2) > k\epsilon | dist(v_1, v_2 > \epsilon)) > P(\chi^2 > k^2)$$

■

Theorem 4 implies that if the distance between v_1 and v_2 is greater than ϵ, then the distance between $\pi_m(v_1)$ and $\pi_m(v_2)$ in the reduced dimensional space will be greater than $k\epsilon$ with the probability that is greater than $P(\chi^2 > k^2)$.

IV. SINGLE PROJECTION BASED SIMILARITY JOIN ALGORITHM USING MapReduce

In order to deal with the massive high-dimensional data similarity join more efficiently, we propose a novel parallel similarity join approach according to the theorem 3 by using MapReduce paradigm. Figure 1 displays the general framework of the Single Projection based Similarity Join Algorithm Using MapReduce(SPSJ), Algorithm 1 describes the detailed procedure. Single projection means that we project d-dimensional vector v into m-dimensional space by using $\pi_m(v) = \langle g_1(v), g_2(v), \ldots, g_m(v)\rangle$ and obtain a m-dimensional vector $\pi_m(v)$.

The Nested Loop Join approach is used to perform the similarity join query. All the vectors in data set Q are evenly divided into c partitions, all the partitions need to be compared with each other, so each vector has to be replicated c times.

Algorithm 1 has two phases: map phase and reduce phase. The main task of the map phase is to divide all the vectors in Q evenly into c partitions and assign a random partition id($\leq c$) to each vector(line 2), $\pi_m(value)$ represents the m-dimensional projection of each vector. Then construct the newKey (line 5, 9) and the newValue(line 6, 10) by combining pid, $\pi_m(value)$ and $value$, finally emit the $\langle newKey, newValue\rangle$ pairs(line 7, 11) for c times totally(line 4-11). In the reduce phase, we firstly obtain the Euclidean distance between $\pi_m(v_1)$ and $\pi_m(v_2)$, recorded as $temp$(line 14), if $temp$ is bigger than ϵ, we can filter out $\langle v_1, v_2\rangle$ safely in advance(line 15), otherwise, the Euclidean distance between v_1 and v_2 in d-dimensional space needs to be computed again, recorded as $dist$(line 16), if $dist \leq \epsilon$, $\langle v_1, v_2\rangle$ is the final answer, and will be emitted(line 17-18).

Algorithm 1 Single Projection Based Similarity Join Algorithm Using MapReduce

```
1  map(key, value)//each value is a vector
2  pid ← Math.abs(random.nextInt())%c +1;//assigning a
   random partition id to each vector.
3  π_m(value) ← Mapping(value);//π_m(value) =
   ⟨g_1(value), g_2(value), ..., g_m(value)⟩.
4  for i = 1; i ≤ pid; i++ do
5      newKey ← "p" + i + "p" + pid;
6      newValue ← ⟨"p" + pid, π_m(value), value⟩
7      output(newKey, newValue);
8  for i = pid + 1; i ≤ c; i++ do
9      newKey ← "p" + pid + "p" + i;
10     newValue ← ⟨"p" + pid, π_m(value), value⟩
11     output(newKey, newValue);
12 reduce(newKey, newValues)
13 foreach vector pairs ⟨v_1, v_2⟩ ∈ newValues do
14     temp ← Δ_m(v_1, v_2);
15     if temp ≤ kε then
16         dist ← dist(v_1, v_2);
17         if dist ≤ ε then
18             output(⟨v_1, v_2⟩, dist);
```

V. MULTIPLE PROJECTIONS BASED SIMILARITY JOIN ALGORITHM USING MapReduce

According to Theorem 3 and 4, the lower bound of the recall and filter effect of the single projection based similarity join algorithm are $1 - P(\chi^2 > k^2)$ and $P(\chi^2 > k^2)$ respectively.

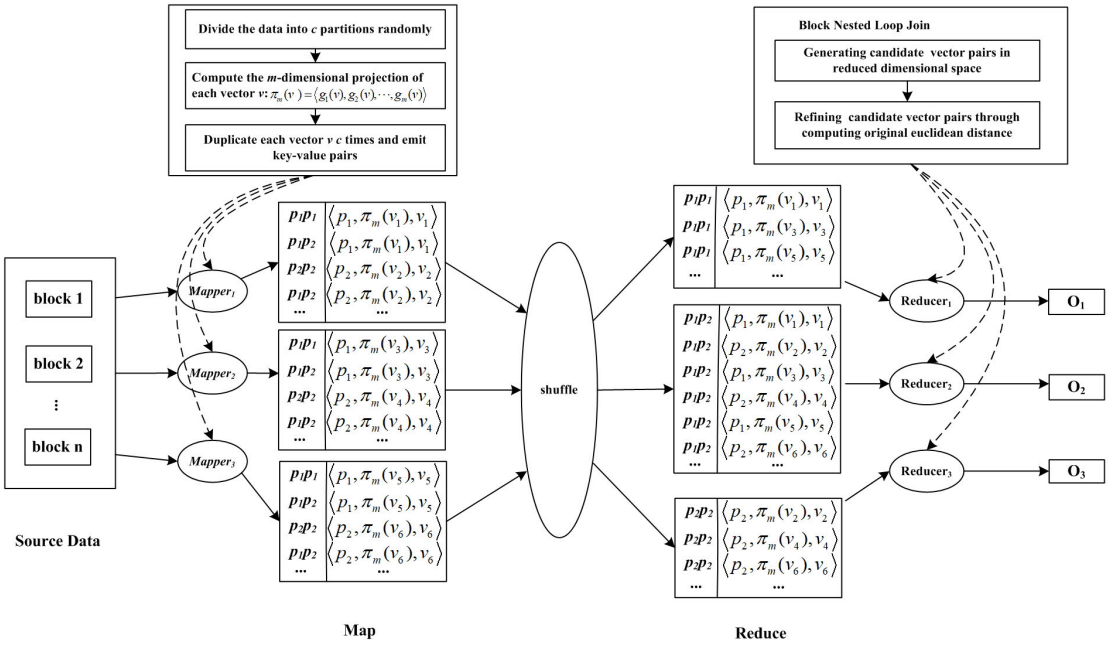


FIGURE 1. General framework of single projection based similarity join algorithm.

The filter effect will decrease as the recall increases, so if we want to obtain high recall, the filter effect may be low, for example, if the lower bound of the recall is 0.9, that is $1 - P(\chi^2 > k^2) = 0.9$, then the lower bound of the filter effect will be 0.1, that is $P(\chi^2 > k^2) = 0.1$. In such case, the performance of the single projection based similarity join algorithm will be affected. In most cases, we want to make sure that the recall and the filter effect are both relative higher. A novel similarity join approach based on multiple projections according to theorem 5 and theorem 6 can achieve the objective. Multiple projections mean that we project d-dimensional vector v into multiple m-dimensional vectors by using $\pi_m^i(v) = \langle g_1^i(v), g_2^i(v), \ldots, g_m^i(v) \rangle$, and obtain several m-dimensional vector $\pi_m^1(v), \pi_m^2(v), \ldots, \pi_m^k(v)$.

Theorem 5: If $\Delta_m^1(v_1, v_2) > k\epsilon$ *or* $\Delta_m^2(v_1, v_2) > k\epsilon$ *or* $\Delta_m^3(v_1, v_2) > k\epsilon$, *then the probability that* $dist(v_1, v_2)$ *will be bigger than* ϵ *is greater than* $3(1 - P(\chi^2 > k^2)) - 3(1 - P(\chi^2 > k^2)^2 + (1 - P(\chi^2 > k^2)^3$, *that is:* $P(dist(v_1, v_2) > \epsilon | \Delta_m^1(v_1, v_2) > k\epsilon$ *or* $\Delta_m^2(v_1, v_2) > k\epsilon$ *or* $\Delta_m^3(v_1, v_2) > k\epsilon) > 3(1-P(\chi^2 > k^2)) - 3(1-P(\chi^2 > k^2)^2 + (1-P(\chi^2 > k^2)^3$.

Proof: Supposing that **A** represents $dist(v_1, v_2)$, $\Delta_m^1(v_1, v_2) > k\epsilon$, $\Delta_m^2(v_1, v_2) > k\epsilon$ and $\Delta_m^3(v_1, v_2) > k\epsilon$ are recorded as **B**,**C** and **D** respectively. According to theorem 3, we can obtain, $P(A|B), P(A|C), P(A|D), P(B), P(C)P(D), P(A|B \cup C \cup D), \because P(B), \therefore P(AB), \therefore P(AC), \therefore P(AB), \therefore P(ACD), P(ABD), P(ABC), P(ABCD), \because P(A|B)$, and $\therefore P(A|B \cup C \cup D)$, as shown at the bottom of the next page. ■

Theorem 6: If $dist(v_1, v_2) > \epsilon$, *then* $P(\Delta_m^1(v_1, v_2) > k\epsilon \cup \Delta_m^2(v_1, v_2) > k\epsilon \cup \Delta_m^3(v_1, v_2) > k\epsilon | dist(v_1, v_2) > \epsilon) > 3P(\chi^2 > k^2) - 3P(\chi^2 > k^2)^2 + P(\chi^2 > k^2)^3$

The proof procedure of theorem 6 is the same as that of theorem 5.

According to theorem 5 and theorem 6, we can obtain that the lower bound of the recall and filter effect are $3(1-P(\chi^2 > k^2)) - 3(1 - P(\chi^2 > k^2)^2 + (1 - P(\chi^2 > k^2)^3$ and $3P(\chi^2 > k^2) - 3P(\chi^2 > k^2)^2 + P(\chi^2 > k^2)^3$ respectively. Supposing that $P(\chi^2 > k^2) = 0.3$, so the lower bound of the recall will be $3*(1-0.3) - 3*(1-0.3)^2 + (1-0.3)^3$, that is 0.973. and the lower bound of the filter effect will be $3*0.3 - 3*0.3^2 + 0.3^3$, that is 0.657. Compared with the previous algorithm, the higher filter effect can be obtained when the recall is approximately the same. We propose Multiple Projection based Similarity Join Algorithm Using MapReduce(MPSJ) based on theorem 5 and theorem 6, Figure 2 displays the general framework of MPSJ.

There are two main improvements compared to SPSJ algorithm, the first improvement is: at map phase, we generate multiple m-dimensional projections for each vector v(supposing that l times) which are $\pi_m^1(v), \pi_m^2(v), \cdots, \pi_m^l(v)$ (line 3), then combing the partition id, l times m-dimensional projections and the original vector v together as the *value* of the output of map phase(line 4 - 11), that is: $\langle pid, \pi_m^1(v), \pi_m^2(v), \cdots, \pi_m^l(v), v \rangle$. the second improvement is: at reduce phase, Block Nested Loop Join approach is adopted to conduct the similarity join, for each vector pair $\langle v_i, v_j \rangle$, we compute the distance of their m-dimensional projections, if $dist(\pi_m^1(v_i), \pi_m^1(v_j)) > k\epsilon$, $\langle v_i, v_j \rangle$ can be filtered out in advance, otherwise, the distance of the next m-dimensional projections will be figured out again, and so on(line 15 - 18). If $dist(\pi_m^1(v_i), \pi_m^1(v_j))$, $dist(\pi_m^2(v_i), \pi_m^2(v_j)), \ldots, dist(\pi_m^l(v_i), \pi_m^l(v_j))$ are all less than or equal to $k\epsilon$, $\langle v_i, v_j \rangle$ will be the candidate pair. Finally we

need to verify the original Euclidean distance between v_i and v_j, if $dist(v_i, v_j) \leq \epsilon$, $\langle v_i, v_j \rangle$ will be the final similar vector pair(line 19 - 22).

VI. PROJECTION SPACE PARTITIONING BASED SIMILARITY JOIN ALGORITHM USING MapReduce

SPSJ algorithm and MPSJ algorithm can filter out some vector pairs which are impossible similar through computing the m-dimensional distance, every pair of the vectors still needs to be compared in m-dimensional space, the time complexity is $\mathcal{O}(n^2)$. A novel parallel similarity join algorithm based on Projection Space Partitioning scheme(PSPSJ) is proposed which can reduce the comparison times effectively, Figure 3 shows the General Framework of PSPSJ Algorithm.

Firstly, figuring out the 1-dimensional projection and m-dimensional projection of the original d-dimensional vectors respectively using p-stable distribution, recorded as $\pi_1(v)$ and $\pi_m(v)$. Then partitioning the data set in 1-dimensional space, the partitioning scheme is as the following: ϵ represents the d-dimensional distance threshold, then the distance threshold in 1-dimensional space is $\epsilon_1 = k_1\epsilon$, the distance threshold in m-dimensional space is $\epsilon_m = k_m\epsilon$. All the vectors can be partitioned into several sub partitions which have the same width (ϵ_1) according to their projected value in 1-dimensional space.

For each sub partition S_i, the vectors in itself and its two adjacent sub partitions(S_{i-1} and S_{i+1}) are more likely to be similar with the vectors in S_i, but the vectors in other sub partitions are not, that is: S_i only needs to join with $\tilde{S}_i$, $\tilde{S}_i = S_{i-1} \bigcup S_i \bigcup S_{i+1}$. Then for each vector pair $\langle v_1, v_2 \rangle, v_1 \in S_i, v_2 \in \tilde{S}_i$, the distance of $\pi_m(v_1)$ and $\pi_m(v_2)$ is figured out firstly, recorded as $\Delta_m(v_1, v_2)$, if $\Delta_m(v_1, v_2) > k_m\epsilon$,

$$
\begin{aligned}
P(A|B) &> 1 - P(\chi^2 > k^2) \\
P(A|C) &> 1 - P(\chi^2 > k^2) \\
P(A|D) &> 1 - P(\chi^2 > k^2) \\
P(B) &= 1 \\
P(C) &= 1 \\
P(D) &= 1 \\
P(A|B \cup C \cup D) &= \frac{P(A(B \cup C \cup D))}{P(B \cup C \cup D)} \\
&= \frac{P(AB \cup A(C \cup D))}{P(B) + P(C \cup D) - P(B(C \cup D))} \\
&= \frac{P(AB) + P(A(C \cup D)) - P(ABA(C \cup D))}{P(B) + P(C \cup D) - P(B(C \cup D))} \\
&= \frac{P(AB) + P(AC \cup AD) - P(ABC \cup ABD))}{P(B) + P(C) + P(D) - P(CD) - P(BC \cup BD)} \\
&= \frac{P(AB) + P(AC) + P(AD) - P(ACD) - P(ABC) - P(ABD) + P(ABCD)}{P(B) + P(C) + P(D) - P(CD) - P(BC) - P(BD) + P(BCD)} \\
\because P(B) &= 1, \quad P(C) = 1, \ P(D) = 1 \\
\therefore P(AB) &= P(B)P(A|B) = P(A|B) \\
\therefore P(AC) &= P(C)P(A|C) = P(A|C) \\
\therefore P(AB) &= P(D)P(A|D) = P(A|D) \\
\because B, C, D &\ \textit{are independent of each other} \\
\therefore P(ACD) &= P(CD)P(A|CD) = P(C)P(D)P(A|C)P(A|D) = P(A|C)P(A|D) \\
P(ABD) &= P(BD)P(A|BD) = P(B)P(D)P(A|B)P(A|D) = P(A|B)P(A|D) \\
P(ABC) &= P(BC)P(A|BC) = P(B)P(C)P(A|B)P(A|C) = P(A|B)P(A|C) \\
P(ABCD) &= P(B)P(C)P(D)P(A|B)P(A|C)P(A|D) = P(A|B)P(A|C)P(A|D) \\
&\Rightarrow P(A|B \cup C \cup D) \\
&= \frac{P(A|B) + P(A|C) + P(A|D) - P(A|C)P(A|D) - \ldots + P(A|B)P(A|C)P(A|D)}{P(B) + P(C) + P(D) - P(B)P(C) - P(B)P(D) - P(C)P(D) + P(B)P(C)P(D)} \\
&= \frac{P(A|B) + P(A|C) + P(A|D) - P(A|C)P(A|D) - \ldots + P(A|B)P(A|C)P(A|D)}{1 + 1 + 1 - 1 * 1 - 1 * 1 - 1 * 1 + 1 * 1 * 1} \\
\because P(A|B) &> 1 - P(\chi^2 > k^2), P(A|C) > 1 - P(\chi^2 > k^2), P(A|D) > 1 - P(\chi^2 > k^2) \\
\therefore P(A|B \cup C \cup D) &> 3(1 - P(\chi^2 > k^2)) - 3(1 - P(\chi^2 > k^2)^2 + (1 - P(\chi^2 > k^2)^3
\end{aligned}
$$

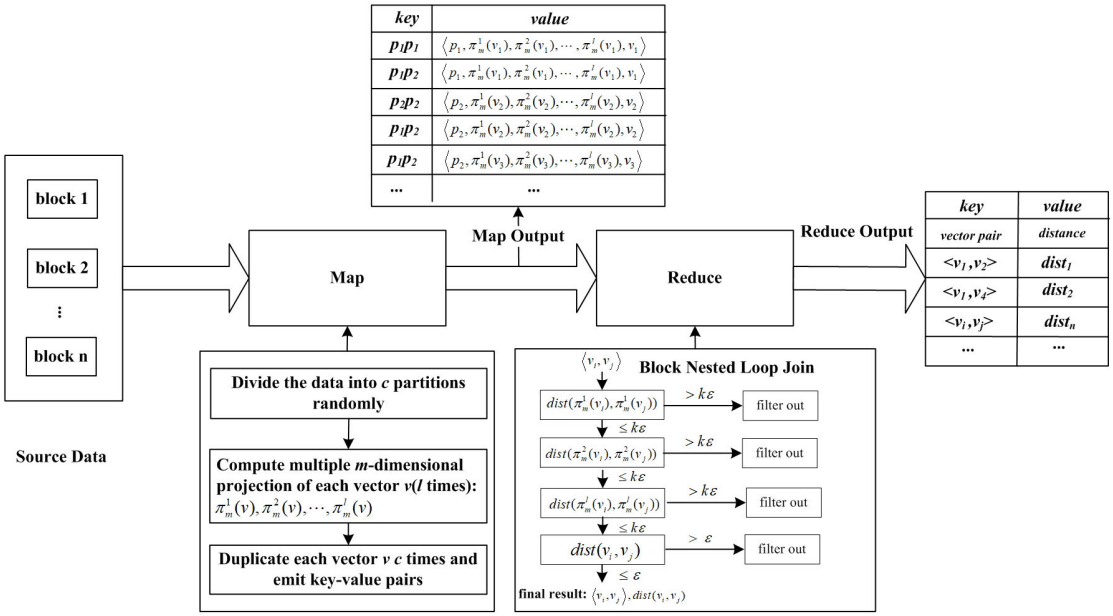


FIGURE 2. General framework of multiple projection based similarity join algorithm.

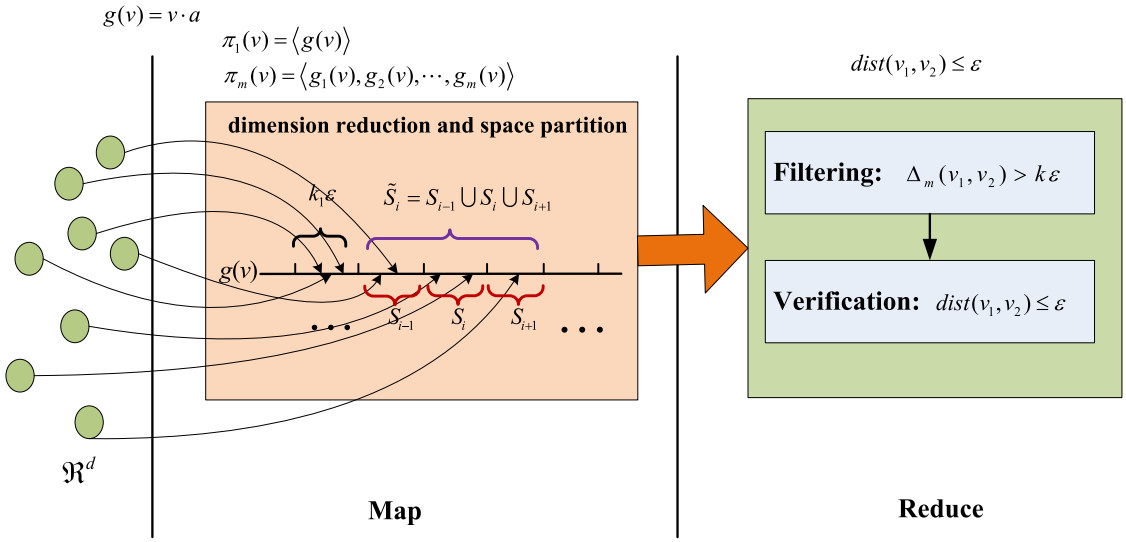


FIGURE 3. General framework of projection space partitioning based similarity join algorithm.

$\langle v_1, v_2\rangle$ can be filtered out. Otherwise, $\langle v_1, v_2\rangle$ will become the candidate vectors pair. Finally, we need to verify the candidate vectors pair $\langle v_1, v_2\rangle$ by computing their original d-dimensional distance.

Algorithm 3 displays the detailed procedure of the similarity join algorithm using MapReduce based on projection space partitioning scheme. In the map phase, we firstly obtain the 1-dimensional projection $\pi_1(value) = g(value)$ and m-dimensional projection $\pi_m(value) = \langle g_1(value), g_2(value), \ldots, g_m(value)\rangle$ for each vector respectively(line 2-3), $\pi_1(value)$ is used to divide the vectors and $\pi_m(value)$ is used to filter the vectors. A partition id recorded as $pid = \lceil \pi_1(value)/\epsilon_1 \rceil$ will be created for each vector making sure that all the partitions have the same with ϵ_1. Line 5-21 can make sure that each sub partition S_i needs only to be compared with itself and its adjacent sub partitions(S_{i-1} and S_{i+1}). The main task of the reduce phase is to conduct secondary filtering and verification for each candidate vector pair $\langle v_1, v_2\rangle$, the distance of $\pi_m(v_1)$ and $\pi_m(v_2)$ is firstly computed(line 24), if $\Delta_m(v_1, v_2)$ is bigger than $\epsilon_m = k_m\epsilon$, then $\langle v_1, v_2\rangle$ can be dismissed safely(line 25). Otherwise, the original d-dimensional distance recorded as $dist$ needs to be computed again(line 26), if $dist \leq \epsilon$, then $\langle v_1, v_2\rangle$ will be the final answer(line 27-28).

Algorithm 2 Multiple Projection Based Similarity Join Algorithm Using MapReduce

map(key, value)//each value is a vector
pid ← Math.abs(random.nextInt())%c +1;//assigning a random partition id to each vector.
generating l times m-dimensional projections for each vector *value*: $\pi_m^1(value)$, $\pi_m^2(value)$, $\cdots$, $\pi_m^l(value)$//$\pi_m^i(value) = \langle g_1^i(value), g_2^i(value), \ldots, g_m^i(value)\rangle$.
for $i = 1; i \leq pid; i++$ **do**
    newKey ← "p" + i + "p" + pid;
    newValue ← ⟨"p" + pid, $\pi_m^1(value)$, $\pi_m^2(value)$, $\cdots$, $\pi_m^l(value)$, $value$⟩
    output(newKey, newValue);
for $i = pid + 1; i \leq c; i++$ **do**
    newKey ← "p" + pid + "p" + i;
    newValue ← ⟨"p" + pid, $\pi_m^1(value)$, $\pi_m^2(value)$, $\cdots$, $\pi_m^l(value)$, $value$⟩
    output(newKey, newValue);
reduce(newKey, newValues)
foreach *vector pairs* $\langle v_1, v_2\rangle \in newValues$ **do**
    $i \leftarrow 1$;
    for $i = 1; i \leq l; i++$ **do**
        $temp \leftarrow \Delta_m^i(v_1, v_2)$;
        if $temp > k\epsilon$ **then**
            break;
    if $i > l$ **then**
        $dist \leftarrow dist(v_1, v_2)$;
        if $dist \leq \epsilon$ **then**
            output($\langle v_1, v_2\rangle$, $dist$);

Algorithm 3 Projection Space Partitioning Based HDSJ

map(key, value)//each value is a vector
$\pi_1(value) \leftarrow Mapping_1(value)$;//$\pi_1(value) = g(value)$. Being used to divide the vectors into several partitions with equal-sized width in the projected 1-dimensional space.
$\pi_m(value) \leftarrow Mapping_m(value)$;//$\pi_m(value) = \langle g_1(value), g_2(value), \ldots, g_m(value)\rangle$.
$pid \leftarrow \lceil \pi_1(value)/\epsilon_1 \rceil$;
if $pid == 1$ **then**
    newValue ← ⟨pid, ⟨$\pi_m(value)$, $value$⟩⟩
    newKey ← "S" + pid;
    output(newKey, newValue);
    $pidRight \leftarrow pid + 1$;
    newKey ← "S" + pid + "S" + $pidRight$;
    output(newKey, newValue);
else
    newValue ← ⟨pid, ⟨$\pi_m(value)$, $value$⟩⟩
    newKey ← "S" + pid;
    output(newKey, newValue);
    $pidRight \leftarrow pid + 1$;
    newKey ← "S" + pid + "S" + $pidRight$;
    output(newKey, newValue);
    $pidLeft \leftarrow pid - 1$;
    newKey ← "S" + $pidLeft$ + "S" + pid;
    output(newKey, newValue);
reduce(newKey, newValues)
foreach *vector pairs* $\langle v_1, v_2\rangle \in newValues$ **do**
    $\Delta_m(v_1, v_2) \leftarrow dist(\pi_m(v_1), \pi_m(v_2))$;
    if $\Delta_m(v_1, v_2) \leq \epsilon_m$ **then**
        $dist \leftarrow dist(v_1, v_2)$;
        if $dist \leq \epsilon$ **then**
            output($\langle v_1, v_2\rangle$, $dist$);

VII. RESULTS

Detailed experiments are performed to validate our proposed approaches' performance: Single Projection based Similarity Join(SPSJ), Multiple Projections based Similarity Join(MPSJ) and Projection Space Partitioning based Similarity Join(PSPSJ).

A. EXPERIMENTAL SETUP

The experiments are implemented on Hadoop-2.7.3, the cluster contains 11 nodes in which one node is Master node and the other 10 nodes are slave nodes. the configuration for each node is described in Table 2. The distance thresholds are 0.1, 0.2, 0.3 0.4 and 0.5 respectively. The data sets used in our experiments are the same with [20] which can be downloaded from the internet,[1] table 3 describes the details of the datasets.

B. PERFORMANCE VS. DISTANCE THRESHOLD

Figure 4 shows the performance of BDAA, SPSJ, MPSJ and PSPSJ on image-128-5 which contains 500 thousand vectors with 128 dimensions under different distance threshold. Although BDAA, SPSJ and MPSJ adopt respective dimension reduction techniques, every pair of the vectors needs to be compared once, so the time complexity is square. While PSPSJ can divide the data set into several partitions, it is easy to determine which partitions need to be compared that some vectors pais can be filtered out in advance at a lower cost. So the performance of PSPSJ is the best when the distance threshold is less than 0.3. However, when the distance threshold $\epsilon > 0.3$, the number of the partitions which can be divided into will be small, each partition contains more vectors accordingly. On the one hand, the filtering effect will decrease, on the other hand, every partition pair needs to be processed by one Map task, because the partition pais number becomes smaller, the parallelism decreases and the computing power of the cluster can not be fully utilized. So when the distance threshold $\epsilon > 0.3$, the performance of PSPSJ decreases dramatically. In conclusion, when the

[1] *http : //corpus − texmex.irisa.fr/*

TABLE 2. Cluster configuration.

Node Name	CPU	Memory	OS	Disk	Map Task	Reduce Task
Master	Intel I5-6500 3.2GHz,4 cores	8GB	Ubuntu 16.04	1TB	-	-
Slave1	Intel I7-6700 3.4GHz,8 cores	36GB	Ubuntu 16.04	1TB	8	8
Slave2	Intel I7-6700 3.4GHz,8 cores	20GB	Ubuntu 16.04	1TB	6	6
Slave3	Intel I7-6700 3.4GHz,8 cores	36GB	Ubuntu 16.04	1TB	8	8
Slave4	Intel I5-6500 3.2GHz,4 cores	10GB	Ubuntu 16.04	1TB	4	4
Slave5	Intel I5-6500 3.2GHz,4 cores	10GB	Ubuntu 16.04	1TB	4	4
Slave6	Intel I5-6500 3.2GHz,4 cores	10GB	Ubuntu 16.04	1TB	4	4
Slave7	Intel I5-6500 3.2GHz,4 cores	10GB	Ubuntu 16.04	1TB	4	4
Slave8	Intel I5-6500 3.2GHz,4 cores	10GB	Ubuntu 16.04	1TB	4	4
Slave9	Intel G3220, 3.0GHz, 2 cores	4GB	Ubuntu 16.04	1TB	2	2
Slave10	Intel G3220, 3.0GHz, 2 cores	4GB	Ubuntu 16.04	1TB	2	2

TABLE 3. Datasets details.

Dataset	Vector Num.($\times 10^3$)	Dimensionality	Data Size
Image-128-5	500	128	423M
Image-128-4	400	128	348M
Image-128-3	300	128	263M
Image-128-2	200	128	176M
Image-128-1	100	128	85M
Image-960-5	500	960	3.92G
Image-960-4	400	960	3.28G
Image-960-3	300	960	2.46G
Image-960-2	200	960	1.64G
Image-960-1	100	960	0.85G
Image-256	500	256	1.09G
Image-512	500	512	2.19G

distance threshold is bigger than 0.3, MPSJ has the best performance, otherwise, PSPSJ has the best performance.

Figure 5 shows the performance of the above proposed methods on image-960-5 which contains 500 thousand vectors with 960 dimensions under different distance threshold. Because the dimensionality of image-960-5 is bigger than that of image-128-5, the time cost of the above proposed methods on image-960-5 is higher than which on image-128-5. However, the trend of the proposed methods with the distance threshold on image-128-5 and image-960-5 is the same.

C. PERFORMANCE VS. DATA SIZE

Figure 6 shows the performance of BDAA, SPSJ, MPSJ and PSPSJ on image-128-2, image-128-3, image-128-4 and image-128-5 whose dimensionality is 128 and the distance threshold is set to 0.1. The experimental results show that the run time of BDAA, SPSJ, MPSJ and PSPSJ increase approximate linearly with the data size increasing, the growth rate of PSPSJ is minimal and it has the best performance among BDAA, SPSJ, MPSJ and PSPSJ. The main reason is that all the projected vectors still need to be compared with each other when using algorithm BDAA, SPSJ and MPSJ,

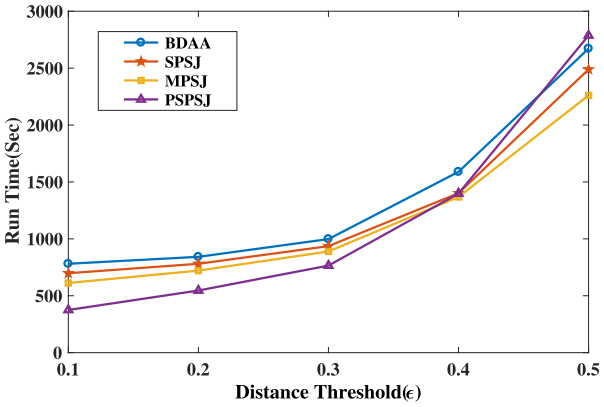


FIGURE 4. Performance with different threshold ϵ (dim = 128, data size = 500 × 10^3).

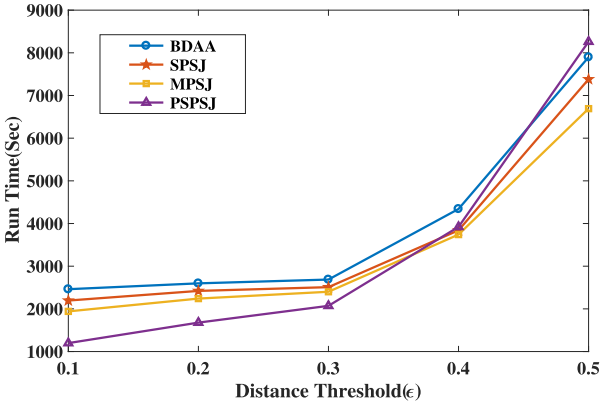


FIGURE 5. Performance with different threshold ϵ (dim = 960, data size = 500 × 10^3).

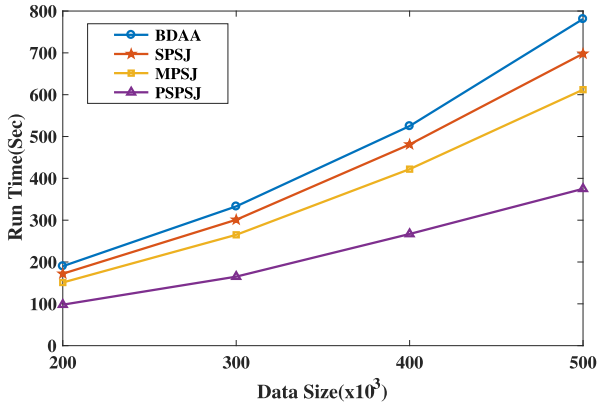


FIGURE 6. Performance with different data size (dim = 128, ϵ = 0.1).

the time complexity is $\mathcal{O}(n^2)$. However, PSPSJ approach can divide the vectors into several disjoint partitions at a lower cost, the vectors coming from two non-adjacent partitions can be filtered in advance according to Theorem 3, so it can reduce the comparison times effectively. Figure 7 displays the experimental results when the distance threshold is set to 0.3, and the results are like with the case in Figure 6.

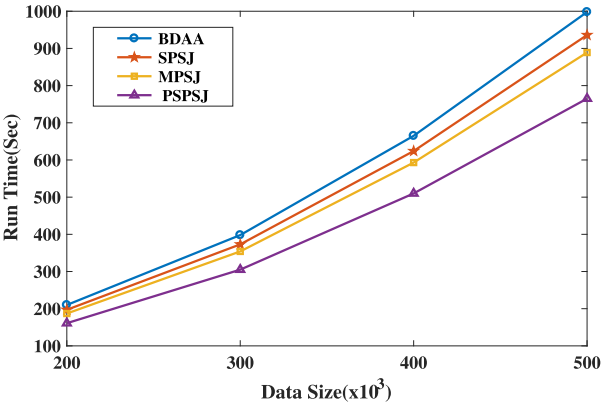


FIGURE 7. Performance with different data size (dim = 128, ϵ = 0.3).

D. PERFORMANCE VS. DIMENSION

Figure 8 and Figure 9 display the performance of BDAA, SPSJ, MPSJ and PSPSJ on data set image-128-5, image-256-5, image-512 and image-960-5 which all contain 500 thousand vectors. The experimental results show that the performance of PSPSJ is the best under different dimensions. The run time of the above methods increase as the data

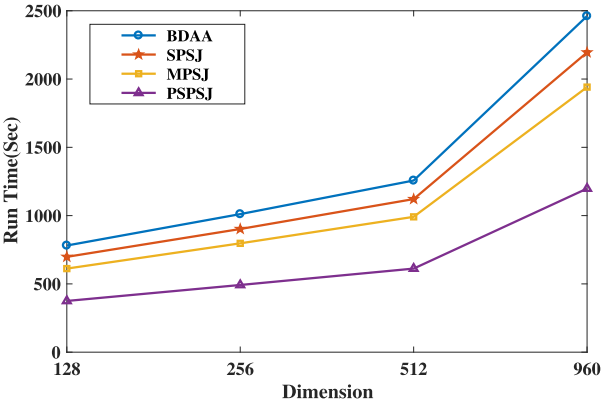


FIGURE 8. Performance with different dimension (data size = 500 × 10^3, ϵ = 0.1).

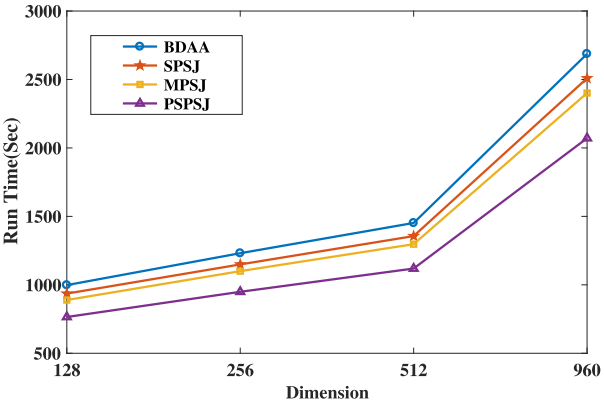


FIGURE 9. Performance with different dimension ϵ (data size = 500 × 10^3, ϵ = 0.3).

size increases, and the run time increases dramatically when the dimensionality is more than 512, while the growth rate of PSPSJ is minimal. The main reason is that the dimensionality of the projected vectors will increase as the dimensionality of the original vectors increases, however, the projected vectors coming from two non-adjacent partitions do not need to be compared with each other, so the size of the dimensions has a relatively small impact on time performance when using PSPSJ algorithm.

E. ALGORITHM PERFORMANCE ANALYSIS

The above experimental results show that the performance of different algorithms is related to the distance threshold ϵ, The algorithm PSPSJ works best when the distance threshold $\epsilon \leq 0.3$, however, when the distance threshold ϵ is over 0.3, the algorithm MPSJ becomes the best one. Given the fixed threshold ϵ, the data size and the data dimensionality have little influence on the performance of different algorithms.

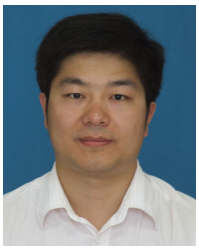
VIII. CONCLUSION

In this paper we mainly conduct research on large scale high-dimensional data similarity joins using *p*-stable based projection scheme and propose parallel approaches under MapReduce framework in order to improve the efficiency. We perform enough experiments to verify the performance of our proposed methods, the experimental results prove that our proposed methods have better performance and scalability. In the future, we plan to extend our work to deal with other complicated data types, such as graph similarity join, time series similarity join and trajectory similarity join. We also plan to exploit the similarity join under Spark paradigm.

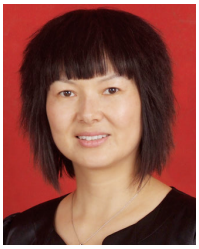
REFERENCES

[1] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *Proc. 6th USENIX Symp. Oper. Syst. Des. Implement. (OSDI)*. San Francisco, CA, USA, 2004, pp. 137–150.
[2] J. Pang, Y. Gu, J. Xu, and G. Yu, "Research advance on similarity join queries," *Frontiers Comput. Sci. Techn.*, vol. 7, no. 1, pp. 1–13, 2013.
[3] X.-M. Lin and W. Wang, "Set and string similarity queries: A survey," *Chin. J. Comput.*, vol. 34, no. 10, pp. 1853–1862, Oct. 2011.
[4] M. Yu, G. Li, D. Deng, and J. Feng, "String similarity search and join: A survey," *Frontiers Comput. Sci.*, vol. 10, no. 3, pp. 399–417, Jun. 2016.
[5] J. Pang, G. Yu, J. X, and Y. Gu, "Similarity joins on massive data based on mapReduce framework," *Comput. Sci.*, vol. 42, no. 1, pp. 1–5, 2015.
[6] Y. Silva, J. Reed, B. K. W. , and C. Rong, "An experimental survey of mapReduce-based similarity joins," in *Proc. SISAP*, 2016, pp. 181–195.
[7] B. Kimmett, V. Srinivasan, and A. Thomo, "Fuzzy joins in mapReduce: An experimental study," in *Proc. VLDB Endowment*, vol. 8, no. 12, pp. 1514–1517, 2015.
[8] J. Lin, "Brute force and indexed approaches to pairwise document similarity comparisons with mapReduce," in *Proc. 32nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, New York, NY, USA, 2009, pp. 155–162.
[9] R. Vernica, M. J. Carey, and C. Li, "Efficient parallel set-similarity joins using mapReduce," in *Proc. SIGMOD*, 2010, pp. 495–506.
[10] L. Shen and Q. X. Peng, "Near duplicated text detection based on mapReduce," *Appl. Mech. Mater.*, vols. 427–429, pp. 2618–2621, Sep. 2013.
[11] C. Rong, W. Lu, X. Wang, X. Du, Y. Chen, and A. K. H. Tung, "Efficient and scalable processing of string similarity join," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2217–2230, Oct. 2013.
[12] T. Elsayed, J. Lin, and D. W. Oard, "Pairwise document similarity in large collections with mapReduce," in *Proc. 46th Annu. Meeting Assoc. Comput. Linguistics Hum. Lang. Technol. Short Papers*, 2008, pp. 265–268.
[13] A. Metwally and C. Faloutsos, "V-SMART-join: A scalable mapReduce framework for all-pair similarity joins of multisets and vectors," *Proc. VLDB Endowment*, vol. 5, no. 8, pp. 704–715, Apr. 2012.
[14] R. Baraglia, G. De Francisci Morales, and C. Lucchese, "Document similarity self-join with mapReduce," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 731–736.
[15] C. Rong, C. Lin, Y. N. Silva, J. Wang, W. Lu, and X. Du, "Fast and scalable distributed set similarity joins for big data analytics," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 1–12.
[16] D. Deng, G. Li, H. Wen, and J. Feng, "An efficient partition based method for exact set similarity joins," *Proc. VLDB Endowment*, vol. 9, no. 4, pp. 360–371, 2015.
[17] J. Wang and C. Lin, "MapReduce based personalized locality sensitive hashing for similarity joins on large scale data," *Comput. Intell. Neurosci.*, vol. 2015, Oct. 2015, Art. no. 217216.
[18] D. Amagata, T. Hara, and C. Xiao, "Dynamic set kNN self-join," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 818–829.
[19] C. Bellas and A. Gounaris, "An empirical evaluation of exact set similarity join techniques using GPUs," *Inf. Syst.*, vol. 89, Mar. 2020, Art. no. 101485.
[20] W. Luo, H. Tan, H. Mao, and L. M. Ni, "Efficient similarity joins on massive high-dimensional datasets using MapReduce," in *Proc. IEEE 13th Int. Conf. Mobile Data Manage.*, Jul. 2012, pp. 1–10.
[21] T. Seidl, S. Fries, and B. Boden, "MR-DSJ: Distance-based self-join for large-scale vector data analysis with mapReduce," in *Proc. BTW*, 2013, pp. 37–56.
[22] S. Fries, B. Boden, G. Stepien, and T. Seidl, "PHiDJ: Parallel similarity self-join for high-dimensional vector data with mapReduce," in *Proc. IEEE 30th Int. Conf. Data Eng.*, Mar. 2014, pp. 796–807.
[23] Y. Xu and H. Chen, "MapReduce-based similaroty join for incremental data set," *Appl. Res. Comput.*, vol. 31, no. 11, pp. 3369–3384, 2014.
[24] Y. Ma, X. Meng, and S. Wang, "Parallel similarity joins on massive high-dimensional data using MapReduce," *Concurrency Comput., Pract. Exper.*, vol. 28, no. 1, pp. 166–183, Jan. 2016.
[25] Y. Ma, S. Jia, and Y. Zhang, "'A novel approach for high-dimensional vector similarity join Query," *Concurrency Comput., Pract. Exper.*, 2017, vol. 29, no. 5, pp. 1–12.
[26] M. Jiang, Y. Song, and J. Chang, "A density-aware similarity join Query processing algorithm on mapReduce," *Advanced Multimedia and Ubiquitous Engineering* (Lecture Notes in Electrical Engineering), vol. 393. Singapore: Springer, 2016, pp. 469–475.
[27] C. Zhang, F. Li, and J. Jestes, "Efficient parallel kNN joins for large data in MapReduce," in *Proc. 15th Int. Conf. Extending Database Technol.*, 2012, pp. 38–49.
[28] W. Lu, Y. Shen, S. Chen, and B. C. Ooi, "Efficient processing of k nearest neighbor joins using MapReduce," *Proc. VLDB Endowment*, vol. 5, no. 10, pp. 1016–1027, Jun. 2012.
[29] J. Dai and Z. Ding, "MapReduce Based Fast kNN Join," *Chin. J. Comput.*, vol. 38, no. 1, pp. 99–108, 2015.
[30] Y. Kim and K. Shim, "Parallel Top-K similarity join algorithms using MapReduce," in *Proc. IEEE 28th Int. Conf. Data Eng.*, Apr. 2012, pp. 510–521.
[31] Y. Ma and X. Ci, "Parallel top-K join on massive high-dimensional vectors.," *Chin. J. Comput.*, vol. 38, no. 1, pp. 86–98, 2015.
[32] D. Chen, C. Shen, J. Feng, and J. Le, "An efficient parallel top-K similarity join for massive multidimensional data using spark," *Int. J. Database Theory Appl.*, vol. 8, no. 3, pp. 57–68, Jun. 2015.
[33] C. Rong, X. Cheng, Z. Chen, and N. Huo, "Similarity joins for high–dimensional data using spark," *Concurrency Comput., Pract. Exper.*, vol. 31, no. 20, Oct. 2019, Art. no. e5339.
[34] S. Zhang, J. Han, Z. Liu, K. Wang, and Z. Xu, "SJMR: Parallelizing spatial join with MapReduce on clusters," in *Proc. IEEE Int. Conf. Cluster Comput. Workshops*, Oct. 2009, pp. 1–8.
[35] Y. Liu, L. Chen, N. Jing, and L. Liu, "Parallel top-K spatial join Query processing on massive spatial data," *J. Comput. Res. Develop.*, vol. 48, no. 3, pp. 163–172, 2011.
[36] Y. Liu, L. Chen, N. Jing, and W. Xiong, "MRFM: An efficient approach to spatial join aggregate," in *Proc. WAIM Workshops*, 2012, pp. 140–150.
[37] Y. Liu, N. Jing, L. Chen, and W. Xiong, "Algorithm for processing k-Nearest join based on R-Tree in MapReduce," *J. Softw.*, vol. 24, no. 8, pp. 1836–1851, Jan. 2014.
[38] H. Gupta, B. Chawda, S. Negi, T. A. Faruquie, L. V. Subramaniam, and M. Mohania, "Processing multi-way spatial joins on map-reduce," in *Proc. 16th Int. Conf. Extending Database Technol.*, 2013, pp. 113–124.

[39] Y. Zhang, Y. Ma, and X. Meng, "Efficient spatio-textual similarity join using MapReduce," in *Proc. IEEE/WIC/ACM Int. Joint Conferences Web Intell.*, Aug. 2014, pp. 52–59.
[40] T. Dan, C. Luo, Y. Li, B. Zheng, and G. Li, "Spatial Temporal Trajectory Similarity Join," in *Proc. APWeb-WAIM*, 2019, pp. 251–259.
[41] L. Zhu, W. Yu, C. Zhang, Z. Zhang, F. Huang, and H. Yu, "SVS-JOIN: Efficient spatial visual similarity join for geo-multimedia," *IEEE Access*, vol. 7, pp. 158389–158408, 2019.
[42] S. Wan, Y. Zhao, T. Wang, and Z. Gu, "Multi-dimensional data indexing and range Query processing via Voronoí diagram for Internet of Things," *Future Gener. Comput. Syst.*, vol. 91, pp. 382–391, Oct. 2019.
[43] B. Lei, J. Xu, Y. Gu, and G. Yu, "Parallel top-K similarity join algorithm on probabilistic data based on Earth mover's distance," *J. Softw.*, vol. 24, no. 2, pp. 188–199, 2013.
[44] J. Huang, R. Zhang, R. Buyya, and J. Chen, "MELODY-JOIN: Efficient Earth Mover's distance similarity joins using MapReduce," in *Proc. IEEE 30th Int. Conf. Data Eng.*, Mar. 2014, pp. 808–819.
[45] Y. Ma and X. Meng, "Set similarity join on massive probabilistic data using MapReduce," *Distrib. Parallel Databases*, vol. 32, no. 3, pp. 447–464, Sep. 2014.
[46] A. Rheinlander and U. Leser, "Scalable sequence similarity search and join in main memory on multi-cores," in *Proc. Euro-Par Workshops*, 2011, pp. 13–22.
[47] D. Deng, G. Li, S. Hao, J. Wang, and J. Feng, "MassJoin: A mapreduce-based method for scalable string similarity joins," in *Proc. IEEE 30th Int. Conf. Data Eng.*, Mar. 2014, pp. 340–351.
[48] C. Lin, H. Yu, W. Weng, and X. He, "Large scale similarity join with edit-distance constraints," in *Proc. DASFAA*, 2014, pp. 328–342.
[49] G. Li, D. Deng, J. Wang, and J. Feng, "Pass-join: A partition-based method for similarity joins," *Proc. VLDB Endowment*, vol. 5, no. 3, pp. 253–264, Nov. 2011.
[50] J. Pang, Y. Gu, J. Xu, Y. Bao, and G. Yu, "Efficient graph similarity join with scalable prefix-filtering using mapReduce," in *Proc. WAIM*, 2014, pp. 415–418.
[51] Y. Chen, X. Zhao, B. Ge, C. Xiao, and C.-H. Chi, "Practising scalable graph similarity joins in mapReduce," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2014, pp. 112–119.
[52] X. Zhang, L. Chen, and M. Wang, "Towards efficient join processing over large RDF graph using mapReduce," in *Proc. SSDBM*, 2012, pp. 250–259.
[53] F. Miu, H. Wang, and Q. Ruan, "Methodof similarityJoinon uncertain graphs using mapReduce," *Comput. Sci.*, vol. 45, no. 12, pp. 299–307, 2018.
[54] F. Miu and H. Wang, "Method for similarity join on uncertain graph database," *J. Softw.*, vol. 29, no. 10, pp. 3150–3163, 2018.
[55] N. Zhou, X. Zhang, C. Liu, and S. Wang, "Similarity join on time series under dynamic time warping," *Chin. J. Comput.*, vol. 41, no. 8, pp. 1798–1813. 2018.
[56] G. Chatzigeorgakidis, K. Patroumpas, D. Skoutas, S. Athanasiou, and S. Skiadopoulos, "Scalable hybrid similarity join over geolocated time series," in *Proc. 26th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, Nov. 2018, pp. 119–128.
[57] S. Ferrada, B. Bustos, and N. Reyes, "An efficient algorithm for approximated self-similarity joins in metric spaces," *Inf. Syst.*, vol. 91, Jul. 2020, Art. no. 101510.
[58] P. Čech, J. Lokoč, and Y. N. Silva, "Pivot-based approximate K-NN similarity joins for big high-dimensional data," *Inf. Syst.*, vol. 87, Jan. 2020, Art. no. 101410.
[59] J. Xu, C. Song, P. Lv, and T. Li, "Distributed similarity join over data streams based on Earth mover's distance," *Chin. J. Comput.*, vol. 42, no. 8, pp. 1779–1796, 2019.
[60] Y. Ma, S. Jia, and Y. Zhang, "Chi-square distribution based similarity join Query algorithm on high-dimensional data," *J. Comput. Appl.*, vol. 7, pp. 1993–1997, Oct. 2016.



YOUZHONG MA (Member, IEEE) received the Ph.D. degree in computer software and theory from the Renmin University of China, in June 2014. He is currently an Associate Professor with the Academy of Information Technology, Luoyang Normal University, Henan, China. His research interests include big data management and analysis, and Web data integration.



RUILING ZHANG received the M.S. degree from Northwestern Polytechnical University, Xi'an, China, in 2007. She is currently a Professor with Luoyang Normal University. Her research interests include intelligent information processing, data mining, and rough set.

ZHANYOU CUI received the Ph.D. degree in mechanical engineering from Xi'an Jiaotong University. He is currently an Associate Professor with the Academy of Information Technology, Luoyang Normal University, Henan, China. His research interests include simulation algorithm and big data mining.

CHUNJIE LIN received the M.S. degree in computer application technology from the Henan University of Science and Technology, in June 2011. He is currently a Lecturer with the Academy of Information Technology, Luoyang Normal University, Henan, China. His research interests include data mining and semantic network analysis.