International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 34, No. 2 (2020) 2059005 (20 pages)

DOI: 10.1142/S0218001420590053

An Improved Group Similarity-Based Association Rule Mining Algorithm in Complex Scenes

Guiduo Duan[*,†,‡,§,**], Xiaotong Wang[*,¶], Tianxi Huang[‖,††]
and Jürgen Kurths[†,‡]

[*]*School of Computer Science and Engineering*
University of Electronic
Science and Technology of China
Chengdu 611731, P. R. China

[†]*Potsdam Institute for Climate Impact Research*
Potsdam 14473, Germany

[‡]*Department of Physics, Humboldt University of Berlin*
Berlin 12489, Germany

[§]*Institute of Electronic and Information*
Engineering of UESTC in Guangdong
Guangdong 511700, P. R. China

[¶]*CECT Ocean Information Co., Ltd*
Beijing 100043, P. R. China

[‖]*Department of Fundamental Courses*
Chengdu Textile College, Chengdu, P. R. China
[**]*Guiduo.Duan@uestc.edu.cn*
[††]*huang_tianxi@163.com*

Received 12 December 2018
Accepted 28 March 2019
Published 14 June 2019

Association rule (AR) mining in complex scene has attracted extensive attention of researchers in recent years. Typically, many researchers focused on an algorithm itself and ignored a generalization method to improve the performance of AR mining. Tuna *et al.*, presented a general data structure Speeding-Up AR Structure with Inverted Index Compression (SAII) which could be utilized in most of the existing algorithms to improve their performance *IEEE Trans. Cybern.* **46**(12) (2016) 3059–3072. However, we found that this algorithm consumes a lot of time in reordering data because a one-to-one comparison method is used in this process, which is the main reason that the speeding-up structure is difficult to establish when coping with much more large amount of data. To overcome these problems, this paper aims to propose an improved speeding-up AR algorithm based on group similarity and Apache Spark framework to further reduce the memory requirements and runtime. Our simulation results on the police business big dataset make clear that our improved approach performs well and is more suitable for a big data environment.

Keywords: Association rule mining; speeding up; business big data; apache spark.

[††]Corresponding author.

1. Introduction

Association rule (AR) analysis algorithm is one of the most important algorithms in the field of machine learning, which has attracted the attention of many researchers. AR learning is a classic machine learning method, which is widely used to discover relationships between different variables in large databases. By using some measures of interestingness, it could identify strong rules hidden in databases.[8] In the past decades, many researchers have proposed several AR mining algorithms, such as the Apriori algorithm,[2] Eclat algorithm,[21] FP-growth algorithm,[9,19] DP-Apriori,[18] Node-set-based algorithms[7,16] and context-based ARs.[10] It is the common characteristic of the classical algorithm that people scan the original set of transaction statistics frequent itemsets and then discover the rules. Much time is consumed there on the statistical frequent itemsets. With the rapid increase in the volume of datasets, as a new challenge has appeared to extract patterns such as the I/O overhead and computational cost. Therefore, classical methods cannot meet the large-scale data processing requirements. In recent years, many researchers have intended to speed up algorithms in ARs to meet the needs in largescale database. The existing speeding-up association analysis algorithms could be divided into vertical, horizontal and new general speeding-up algorithm.

Vertical speeding-up algorithms[11,15–17] refer to the improvement of AR algorithms in a serial mode. The goal is to reduce the time of discovering frequent itemsets by improving the execution flow of the algorithm. Tian *et al.*[17] proposed a new AR mining method applied into privacy-preserving field, which chose different data transition strategies to find frequent itemsets. Experiments showed that this method is effective when the parameters chosen are reasonable. In Ref. 11, Lee *et al.* proposed an algorithm for mining approximate weighted maximal frequent patterns (AWMFPs). This method used the error tolerance to selectively extract representative patterns from the large-scale database. Mohamed *et al.*[15] presented methods called CountTableFI and BinaryCountTableF to achieve better performance than Apriori algorithm and some other related algorithms. Pradhan *et al.*[16] presented an effective method to discover hybrid-multidimensional associative rules in medical sensor database. This technique achieved higher-confident ARs than a bottom-up computation approach for multidimensional pattern mining. However, the performance of the vertical speeding-up algorithm is limited when coping with a large dataset. Horizontal speeding-up algorithms[3,5,20] refer to parallel algorithms using cluster platforms. Based on this parallel algorithm, some instructions could be executed simultaneously on cluster devices. Then the result can be produced by combining all the individual outputs. Nowadays, the improvement of the parallel algorithm for a specific big data platform is much more popular in big data research and its application. Chen *et al.*,[5] proposed the FP_Growth algorithm based on a Boolean matrix which converted the transaction dataset into a Boolean matrix storage and nonrecursive construction of FP-tree, to reduce the space-time overhead of the algorithm. In Ref. 20, a parallel frequent itemset mining algorithm using

FiDoop not FP trees under the MapReduce model was presented. This method solved the problem that the existing parallel mining algorithms cannot be automatically parallelized. Based on FP-growth and Apache Spark framework, Addi Ait-Mlouk *et al.*[3] developed a learning model to extract the ARs. Experiments on big data through the road accident showed that this approach can efficiently achieve higher accuracy ARs and reduce the response time for the proposed algorithm.

The above methods only focused on one of the algorithm's designs that was implemented through a parallelization strategy, or a candidate set a pruning strategy to improve the performance. However, in the real world complex system scenarios, such as the police business system, the data size is extremely large and the business model is complex. Hence, a single algorithm cannot achieve the overall performance improvement. In Ref. 14, the authors put forward a general data structure Speeding-Up AR Structure with Inverted Index Compression (SAII). In contrast to the above methods, SAII did not perform any improved operation on the algorithm itself but proposed a speeding-up structure. Experiments showed that the proposed data structure reduced both the auxiliary and the main memory requirements. However, we found that a lot of time is consumed in re-ordering data because a one-to-one comparison method is used in this process in this algorithm,[14] which is the main reason that the speeding-up structure is difficult to establish when coping with a much larger amount of data. On the other side, with the advent of the Big Data era, especially in the police business system, the amount of data is extremely huge and the business model is complicated. In addition, the association analysis in police business system is used very frequently, such as case correlation analysis, group association analysis, and personnel association analysis. With the continuous growth of police data, it is still hard to do the mining process effectively using the traditional AR algorithm when the number of items and records is extremely high. Furthermore, a single AR acceleration algorithm can only be applied to a certain type of business model and cannot improve the overall performance. Therefore, an improved speeding-up AR mining algorithm is presented in this paper to further reduce the I/O overhead and improve the overall performance.

In this paper, an improved AR mining algorithm based on group similarity to handle a large set of police business data is presented. The paper is based on the motivations as follows: (1) One important step of the shuffling process is to abandon the process of one-by-one comparison. The dynamic grouping strategy of inserting a record based on group clustering is proposed in the process of shuffling. The inserted record only needs to be compared with a small number of similar records to complete the insert sort operation. (2) Secondly, we set different thresholds for different transaction sets. These thresholds are intended to limit the distribution range in the data group, to reduce the number of comparison times, and to achieve the purpose of load balancing inside the cluster. (3) In addition, we utilized the spark framework. Data localization, memory-based parallel computing model and other characteristics provided by Spark framework are used to achieve further improvement of the algorithm.

The remainder of this paper is organized as follows. In Sec. 2, we briefly review the SAII compression. The building process of our improved model including the processes of shuffling is then proposed in Sec. 3. In Sec. 4, we show and analyze the results from our extensive simulations. Finally, we conclude this work and point out some future work directions in Sec. 5.

2. Background of Speeding-up AR Structure with Inverted Index Compression

In Ref. 14, the authors presented a general data structure called SAII which could be used in the speeding-up the AR mining process. The SAII is based on the inverted index structure, and the transaction records are reorganized in the ARs. The main process of SAII consists of two steps: the shuffling strategy and the inverted index mapping strategy,[12] which is divided into an inverted index establishment and a run-length encoding (RLE) compression.[1] Run-length encoding (RLE) [19] is a classic compression algorithm, in which runs of data (same data value occurs in consecutive) are stored as singular value and count representation, rather than as the original run. Figure 1 shows the methodology's workflow which is used to store the data records into the proposed new data structure.

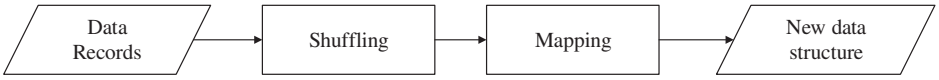


Fig. 1. Workflow for SAII.

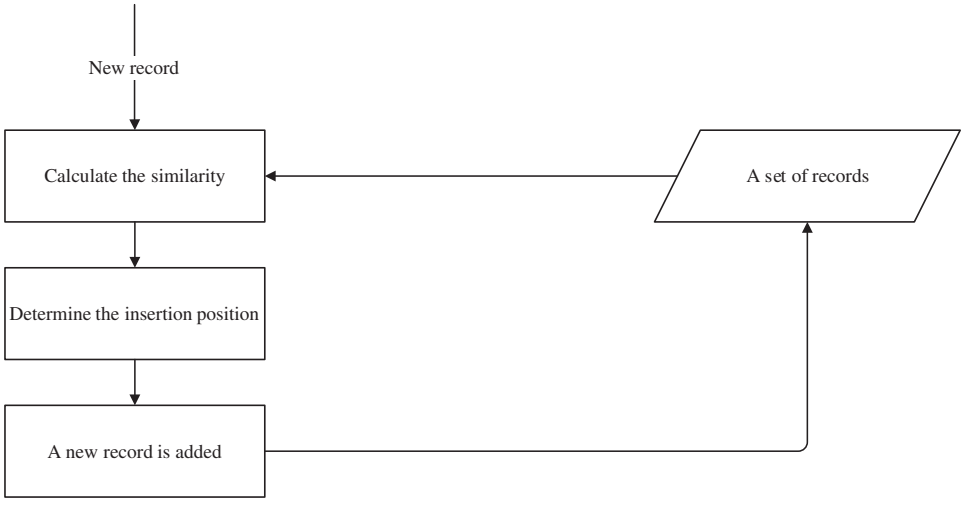


Fig. 2. Workflow of the shuffling process.

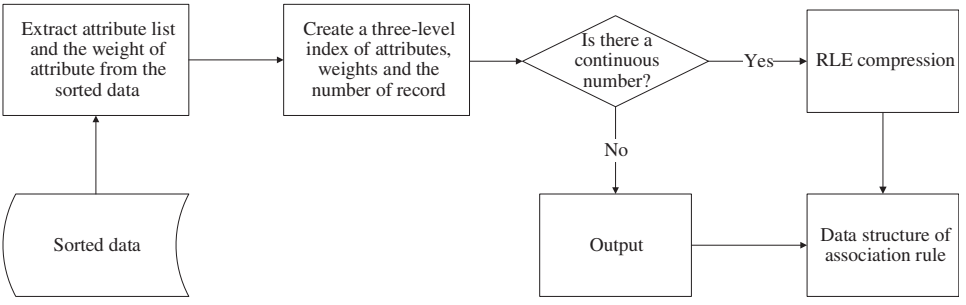


Fig. 3. Workflow of inverted index mapping process.

Shuffling strategy: The data is first re-sorted by similarity among the data records or transactions. Then the highest similarity position is selected for data insertion in turn. The flow of the shuffling is seen in Fig. 2.

Mapping strategy: This process is based on the process of shuffling, and the inverted index structure is constructed on the sorted data with the index attribute values. Then get on the RLE compress according to the inverted index file, and finally the association analysis acceleration structure is formed. This structure is proposed for the association analysis algorithm, which can greatly improve the efficiency of the association analysis algorithm and is universal.

The workflow of inverted index mapping process is seen in Fig. 3.

RLE is very useful on compressing data that contains such runs, for example, simple graphic images, video, etc. For more details about RLE, see Ref. 1.

3. The Improved Group Similarity-Based Association Rule Mining Algorithm

3.1. *Definitions*

In this section, we present the definitions used to describe the process of our shuffling method.

(1) Transaction set

A transaction set A is defined as $A = \{a_1, a_2, \ldots, a_n\}$, where a_i is defined as a transaction record and $a_i \in A, i \in [1, n]$.

$a_i = \{x_{i1}, x_{i2}, \ldots, x_{ij}\}$ denotes that a_i has k transaction attributes, where $x_{ij} \in a_i, i \in [1, n], j \in [1, k]$ and x_{ij} is defined as transaction attribute.

(2) Hamming distance

Let $Q(q_1, q_2, \ldots, q_n)$ and $P(p_1, p_2, \ldots, p_n)$ be two equal length strings. The minimum number of substitutions required to convert one of the strings to another is defined as the Hamming metric between the strings Q and P, and denoted as $\mathrm{HD}(Q, P)$. For example, the Hamming distance (or Hamming metric) between the string "1111" and "1001", $\mathrm{HD}(1111, 1001) = 2$.

(3) Euclidean distance
Let two n dimensional vectors be a and b, a be denoted as $a(a_1, a_2, \ldots, a_n)$, and b denoted as $b(b_1, b_2, \ldots, b_n)$. Then Euclidean distance (or Euclidean metric) between a and b is represented as

$$d(a,b) = \sqrt{\sum_{k=1}^{n} (a_k - b_k)^2}, \quad n \in [1, \infty). \tag{1}$$

(4) r-neighborhood
Let r be a positive integer, D be a dataset. The definition of the r-neighborhood of $p(p \in \mathrm{D})$ is defined as

$$N(p,r) = \{q \in \mathrm{D} | \mathrm{d}(q,p) \leq r\}. \tag{2}$$

(5) k-nearest neighbor distance
Let k be a positive integer. $k\_d(p,o)$ is defined as the k-neighbor distance of k, where $\mathrm{o} \in D, o \neq p$, if it is satisfied

(a) $|\mathrm{N}(p, d(p,o))| \geq k$,
(b) For any $r < d(p,o)$, then $|\mathrm{N}(p,r)| \leq k-1$.

(6) k-nearest neighbor
Let k be a positive integer. The k-neighbor of p consists of all neighboring points satisfying $d(p,q) \leq k\_d(p,q)$ and is defined as

$$N_k(p) = \{q \in \mathrm{D} | d(p,q) \leq k\_d(p,q), q \neq p\}. \tag{3}$$

(7) The center of a group
$G(a_1, a_2, \ldots, a_n)$ is denoted as a data group, where $a_i = \{x_{i1}, x_{i2}, \ldots, x_{ik}\}$, $x_j = \{y_{j1}, y_{j2}, \ldots, y_{jm}\}$, and y_{jm} is the value of the attribute variable. The calculation process of the group center $D(d_j)$ is

$$D(d_j) = \frac{\sum_{j=1}^{k} c_j y_j}{\sum_{j=1}^{k} c_j}, \tag{4}$$

where c_j is the frequency of the attribute value.

3.2. *The improved shuffling algorithm*

The block flow diagram of our algorithm based on group similarity is shown in Fig. 4. The improved speeding-up AR mining algorithm based on the group similarity can be described in two main steps: a shuffling process and a mapping process. The mapping process consists of an inverted index mapping and a compressing process based on RLE. We focus on the improved shuffling strategy in our algorithm.

(1) An improved shuffling algorithm
The shuffle process is to re-order data based on the similarity among records and using a one-by-one comparison method to add a new record to the dataset in Ref. 14.

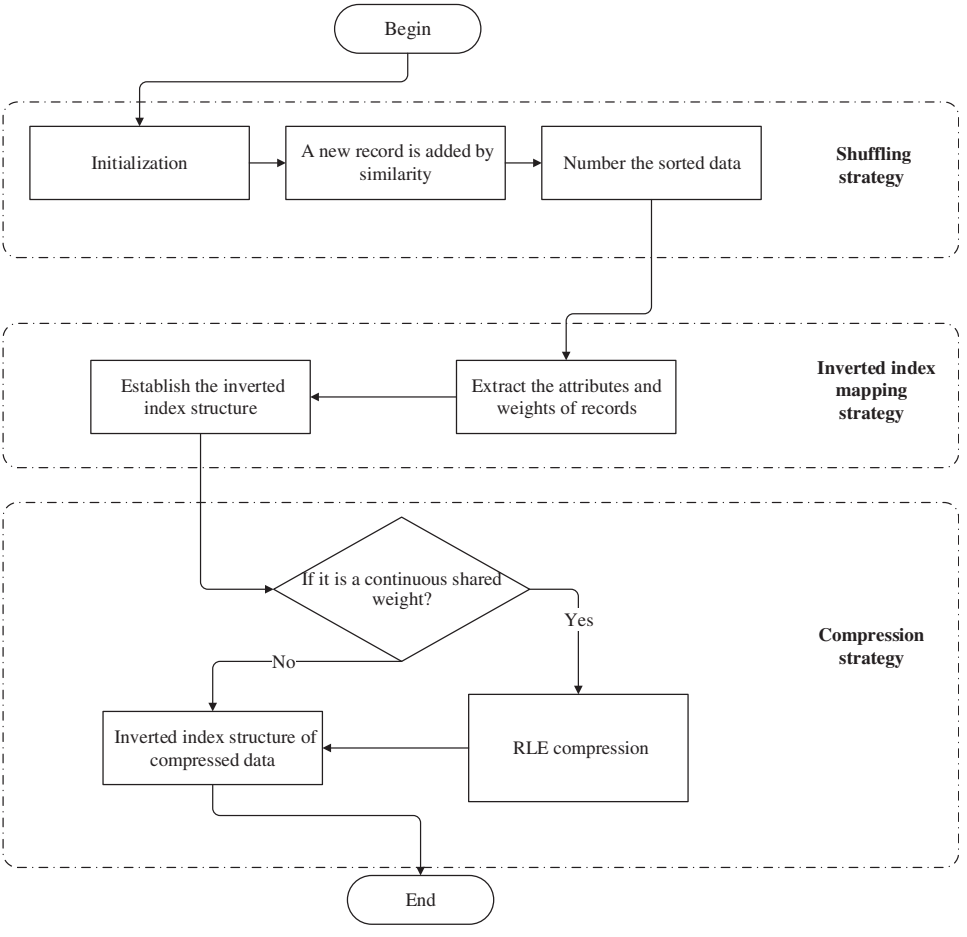


Fig. 4. The block flow diagram of our algorithm based on group similarity.

We found that a large amount of execution time of the algorithm was consumed in the process of data re-ordering and the process used a one-by-one comparison. Thereby, we abandoned the process of one-by-one comparison in our method. In the shuffling process, a dynamic grouping strategy based on clustering insertion data is proposed instead. A new record only needs to be compared with a small amount of similar records to complete the insert sort operation.

The details of this process are shown in Fig. 5.

In Fig. 5, the dataset is first divided into k groups. α denotes the grouping threshold and M the splitting threshold which are all initialized. In our strategy, the k groups are divided into two categories: an external group denoted as G_{ex} and an internal one denoted as G_{in}. The number of G_{ex} is determined by α, and G_{in} is determined by M which reduces the number of insertion comparisons. Each group has a center, and the center of the ith group is denoted as C_i, where $i = 1, 2, \ldots, k$.

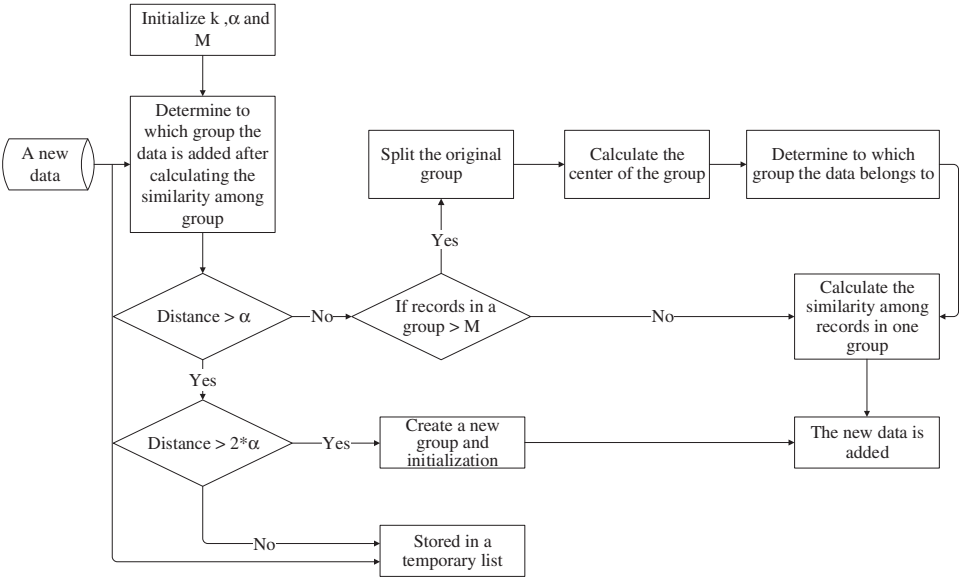


Fig. 5. The process of shuffling based on similarity in a group.

When a new record p is added, we first calculate the similarity among the record and each center of the external group C_i, and the minimum Euclidean distance is achieved. If the distance is greater than 2α, then a new group is created and the data is added into this new group; if the distance is smaller than α, a split operation may be done or not according to the value of M; if the distance belongs to the interval $[\alpha, 2\alpha]$, the record is firstly stored in a temporary list and finally will be added.

When the distance is smaller than α, if the number of record is greater than M, then a split operation is performed. Otherwise, it is not done. Splitting operation refers to the original group further divided again. The split operation is operated as follows

q is defined by the splitting number which is the number of split groups in the splitting operation. The value of q is calculated according to the rate of growth of C_i:

$$q = \begin{cases} \max(g(C_1), g(C_2), \ldots, g(C_k)), & g(C_i) > 2, \\ \min(g(C_1), g(C_2), \ldots, g(C_k)), & g(C_i) \leq 2, \end{cases} \tag{5}$$

where $g(C_i) = \frac{\text{Sorted(records)}}{\sum_{i=1}^{k} C_i}$.

When the original group is determined to split, then the C_i of each sub-group is computed. Finally, the similarity among the records in the new group is calculated to determine the insertion position. The Hamming distance is used here.

An internal grouping G_{in} is equivalent to a two-level group item, which can effectively reduce the times of comparisons before adding. There are two kinds of distance measurements in this model for similarity computation: the Euclidean

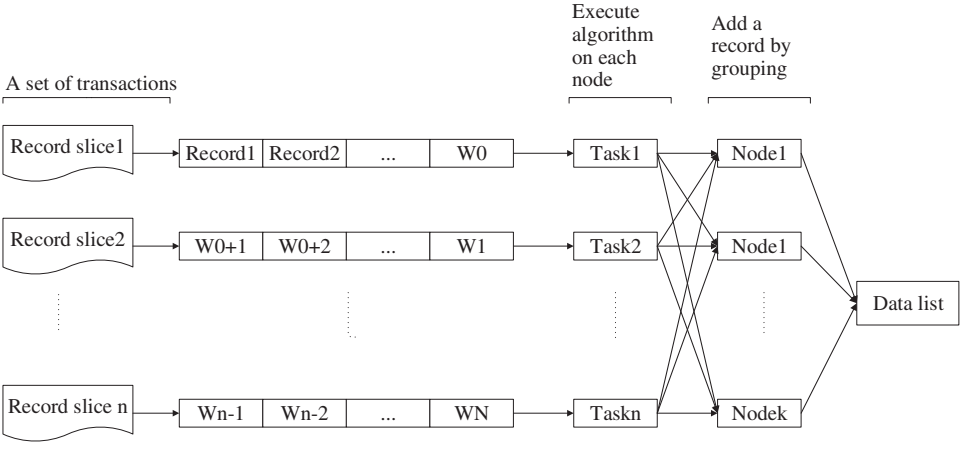


Fig. 6. The block flow of our method using the Spark.

distance metric is used to determine which group the data belongs to and the Hamming distance metric is used to determine the insertion position in a group. The calculation complexity of the Hamming metric is low, hence the total computational complexity is low.

Furthermore, k is equal to the number of cluster executors, and the Spark platform is adopted for the task distribution strategy. Spark's RDDs function as a working set for distributed programs that offers a (deliberately) restricted form of a distributed shared memory. The block flow of our method using the Spark is shown in Fig. 6.

The process of shuffling strategy is done in four steps:

Step 1: Threshold k, α and M are firstly initialized; then the first data is added and initialized into a center of group.
Step 2: Compute the minimum distance between a record p and each of G_{ex} denoted as $d(C_i, p)$ when a record p is added. If $d(C_i, p) < \alpha$, this data belongs to an external group. This external group is the group that calculates the minimum distance from p. After that, if the number of record in this group is greater than M, then split it into G_{in} and compute each center of G_{in}. Insert the record to this G^*_{in} which has the highest similarity. Compute the Hamming distance among p and all the data in the G^*_{in} and then the highest similarity position is selected for this data's final insertion. Otherwise, insert p into G^*_{in} according to the similarity among p and all the data in the G_{ex} directly.
Step 3: If $d(C_i, p) > 2\alpha$, create a new external group and initialize it as the center of the group; if $2\alpha \geq d(C_i, p) \geq \alpha$, put it into the temporary list.
Step 4: Repeat Steps 2 and 3 until the records in the list are finally added.

Algorithm 1. Shuffling process

Input: original_data
Output: sorted_data
procedure:

```
G_alpha ← 0        //group threshold
G_Max ← 0          //split threshold
G_K ← 0            //the number of external groups
G_alpha ← α, G_K ← k, G_Max ← n    // the three variables are initialized, k is the number of cluster Executor α is set by user, n is the maximum number of data in a group;
The number of Groups == G_K, g_k ← 0, groups ← 0, Group_centergroupID ← 0   //Groups is the set of all groups, groups is the set of internal group, g_k is the number of internal groups, and Group_centergroupID is a specified center of a group
  for a record in original_data do
      for Group in Groups do
      distance ← min(similarity(record,group.center))
          if distance > 2α
                Create a new external group, then put it into Groups, and initialize it, G_K+ = 1
            if distance ≥ α && distance ≤ 2α
                list←record
            else
                if Group.records >= M
                   Split the Group, calculate the center of group, and add the record
                            according to the similarity;
                else
                     Add the record directly;
                end if
            end if
          end if
      end for
  end for
  return sorted_data
end procedure
```

3.3. *Inverted index mapping process*

The inverted index mapping is based on the data after shuffling, and the process of inverted index mapping is performed in three steps:

Step 1: Initialize the index file, attribute value index entry, and attribute index entry.

Algorithm 2. Inverted index mapping process

Input: sorted_data
Output: $\text{data}_{\text{index\_based}}$
procedure:

```
1: data_index_based ← Φ
2: index_value ← Φ
3: index_attribute ← Φ
4: list_attributes ← getAttributes(sorted_data) // list_attributes is the list of the
   attributes of the transaction records in the transaction set, getAttributes() is the
   function to get the list of attributes
5: for all attribute in list_attributes do
6:         for all value in attribute do
7:                 for all transaction in sorted_data do
8:                         if value satisfies transaction then
9:                                 index_value ← index_value ∪ transaction
10:                        end if
11:                end for
12:                index_attribute ← index_attribute ∪ index_value
13:                index_value ← Φ
14:        end for
15:                data_index_based ← data_index_based ∪ index_attribute
16:                index_attribute ← Φ
17:   end for
18:   return data_index_based
end procedure
```

Step 2: Extract the list of attributes in the sorted data.
Step 3: According to the three level index, the mapping relation of transaction attribute, transaction attribute value and transaction record is set up, and the process is looped until all data mapping is finished.

More details are seen in Ref. 14.

3.4. *Record compression process*

The transaction records are compressed using the RLE algorithm, which is done in two steps as follows:

Step 1: Initialize the compressed index structure, transaction attribute value index and transaction attribute three variables;
Step 2: Traverse the inverted index mapping structure, compress the continuous record of the shared transaction attribute value.

More details are seen in Ref. 12.

Algorithm 3. RLE (Run Length Encoding) compression process

Input: $\text{data}_{\text{index\_based}}$
Output: $\text{data}_{\text{structure}}$
procedure:

$\text{data}_{\text{structure}} \leftarrow \Phi$
$\text{index}_{\text{value}} \leftarrow \Phi$
$\text{index}_{\text{attribute}} \leftarrow \Phi$
list_attributes $\leftarrow$ getAttributes($\text{data}_{\text{index\_based}}$)
for all attribute in list_attributes do
    for all value in attribute do
        If value comprises consecutive indices then
            $\text{index}_{\text{value}} \leftarrow$ getCompressedIndices(value)
        else
            $\text{index}_{\text{value}} \leftarrow$ getIndices(value)
        end if
        $\text{index}_{\text{attribute}} \leftarrow \text{index}_{\text{attribute}} \cup \text{index}_{\text{value}}$
        $\text{index}_{\text{value}} \leftarrow \Phi$
    end for
    $\text{data}_{\text{structure}} \leftarrow \text{data}_{\text{structure}} \cup \text{index}_{\text{attribute}}$
    $\text{index}_{\text{attribute}} \leftarrow \Phi$
end for
return $\text{data}_{\text{structure}}$

end procedure

4. Experimental Results

All the experiments are run on a Spark cluster of four compute nodes. Each node is configured on an InterPentuim® dual-core ES5300 processor, processor clocked at 2.6 GHz, hard drive capacity 1T, memory 8G, ubuntu12.04 operating system, and gigabit Ethernet. Our comparative results about the running time, the index file size, the ARs and the acceleration effect are discussed in the experiments.

4.1. *Dataset description*

Although the data types of policing data are numerous and the data structure is complicated, the data related to the scene of association analysis are generally personnel data. Therefore, this experiment dataset is from the most representative cases of police business information in the police business system. All the data does not involve any sensitive information. The dataset contains 351 290 records, and each record contains 38 attributes. The main attributes of the data are given in Table 1. Moreover, the size of the test dataset A1 is 100 000 records, A2 is 200 000 and A3 is 300 000.

Table 1. Selected experimental data.

Attribute List About Security Case		Attribute List for People in Case	
Place	A district	Sex	Male
	B district		Female
	C district	Age	<14
	D district		14–16
Time of solving a case	Within one month		16–18
	1–3 months		18–25
	3–6 months		25–30
	6–12 months		30–38
	12–36 months		38–46
	More than 36 months		46–55
			55–65
			>65
The number of suspects	1 person	Ethnic	Han
			Wei
	1–3 people		Zang
			other
	3–10 pepole	Political Status	Communist party member
			League Member
	>10 pepole		Citizen
			Non
Time of the incident	xx-xx-xxxx	Occupation	Construction industry
			Administration
			Service industry
			Self-employed
			Unemployed
			other
		Education Degree	Primary school
			High school
			Secondary school
			College
			Undergraduate
			Master above

4.2. *Model parameter*

In this section, a grouping threshold α, splitting threshold M and splitting number x are used in the process of our shuffling strategy. The optimal thresholds or their ranges are determined by experimental adjustment. The test set A1 is selected in the experiment.

According to our algorithm, α determines the number of external groups. The larger the α is, the larger the average size of G_{ex} is, resulting in a larger amount of temporary list data. The smaller the α is, the smaller the average size of G_{ex} is, the smaller the data in the list is, and the more the number of groups and the times of comparisons are. M determines the number of internal groups. The larger the M is, the less the number of G_{in} is, and the more the number of times of comparisons is. The smaller the M is, the smaller the average size of G_{in} is, and the more the number of times of comparisons is.

Table 3. Different values of M for different results.

M	The Number of Internal Groups	The Average Times of Comparisons
200	12	511
800	3	716
1600	2	1285
2000	1	1855

Table 2. Different values of α for different results.

α	The Number of External Groups	The Number of Data in List
0.35	15	30325
0.15	203	5542
0.12	253	1254
0.08	1254	102

From Table 2, for $\alpha = 0.35$, the number of data in the list is too large, although the number of the outer grouping is less. For $\alpha = 0.12$, there is a good tradeoff between the number of external groups and the temporary list, and a good performance could be achieved. If the value of α decreases gradually, the number of its external groups surges strongly, so a value of $\alpha \in [0.11, 0.13]$ is more reasonable.

The range of M is 200–2000, which is obtained from the average times of comparison using the Euclidean and Hamming distance. From Table 3, for $M = 200$, the number of internal groups is 12 and the average times of comparisons is 511. If the value of M increases gradually, the number of its internal groups decreases, and the average times of comparisons increase strongly. For $M = 800$, there is a good tradeoff between the number of internal groups and the average times of comparisons.

In the following experiments, we take $\alpha = 0.12$ and $M = 800$.

4.3. *Experimental results*

(1) Runtime

The runtime is used to evaluate whether our algorithm (titled SAIIBC) is improved compared to the SAII algorithm. In this experiment, the dataset A3 is used to compare the construction time of the SAII algorithm and the SAIIBC algorithm's acceleration structure on the Spark platform. For this test set, the number of tests is 100 for different data volumes. The average time under each data volume is obtained as the final result of this experiment. The runtime comparison operating on dataset A3 is shown in Fig. 7. In the following experiments, the number of experiments for the different datasets is 100 and the experimental results are the average results.

From Fig. 7, it can be seen that SAII constructs the data structure with less time than SAIIBC when the size of dataset is small. However, as the amount of data increases, the runtime of SAII shows a rapid growth, while the improved algorithm shows a linear growth only.

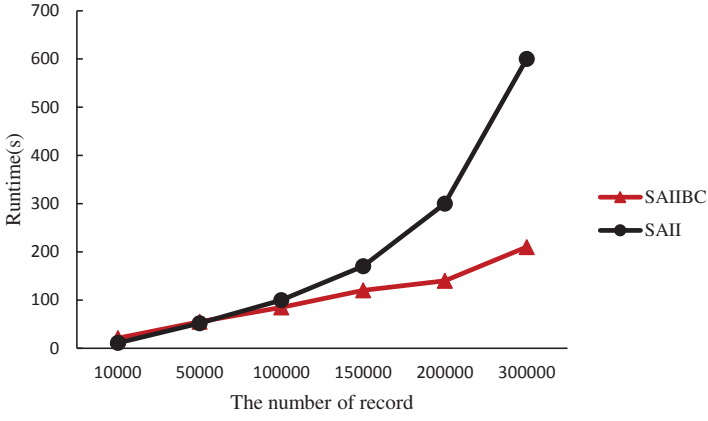


Fig. 7. The comparison results of runtime

The main reason is that for the SAII algorithm, the insertion strategy using the one-by-one comparison when constructing the inverted index compression structure, results in an increase of the amount of data insertion. Much time is spent in calculating and comparing multiple similarities with the previous data when a new record is added. For our algorithm, a new record is added by only comparing with the data in the group but not with all the previous data, thereby it runs in less time.

(2) Complexity Index file size

According to the algorithm described above, the compression efficiency depends on the quality of the shuffle strategy. In this section, we test the performance between our speeding-up structure (SAIIBC) with the general structure (SAII) using the police business datasets. Datasets A1, A2 and A3 are carried out on the experiments. The results are shown in Fig. 8.

In Fig. 8, the size of the index file generated by the two algorithms is basically the same. With the increase of the number of records, the compression rate of the index file of SAIIBC is a little lower than that of the SAII algorithm. This is mainly because SAIIBC uses the grouping method without the global similarity comparison when a new record is to be added, such that some of the data are not clustered, and make the index file compression rate slightly lower. However, from Fig. 8, the difference between both methods is quite small.

(3) Memory requirements

Generally, the low memory usage indicates that the algorithm has higher execution efficiency. This experiment is based on the classical Apriori algorithm[2] with the support of 0.25 and the confidence of 0.5; it tests the running memory usage of this algorithm on SAII, SAIIBC and the original algorithm without the speeding-up process, respectively.

Our experimental results show that compared to the original method, both SAII and SAIIBC can effectively reduce the memory usage of Apriori algorithm in two

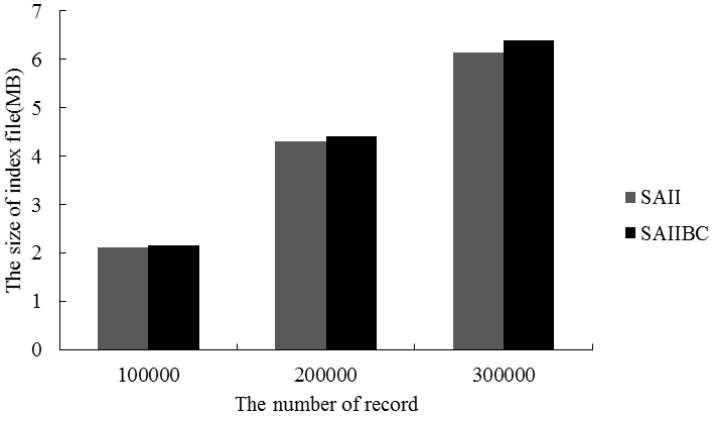


Fig. 8. The comparison results of Index file size.

ways. However, SAIIBC only changes the strategy of the SAII algorithm in building the association acceleration structure, and the experimental results show that the index compression structures constructed in the two ways are almost the same. So, the memory footprint is almost the same. In Fig. 9, the experimental results show that for a small amount of data (within 50 000), there is not much difference among the three ways, even the traditional way is better. The main reason is that the final compression of the acceleration structure does not compress the data effectively when the amount of data is small. When the amount of data increases, the algorithm running on two acceleration structures is better for memory occupation.

(4) Acceleration effect

The acceleration structure could be used by a majority of existing algorithms to enhance the performance. In this experiment, four classical association analysis algorithms are selected, including Apriori algorithm,[2] FP-Growth algorithm,[9]

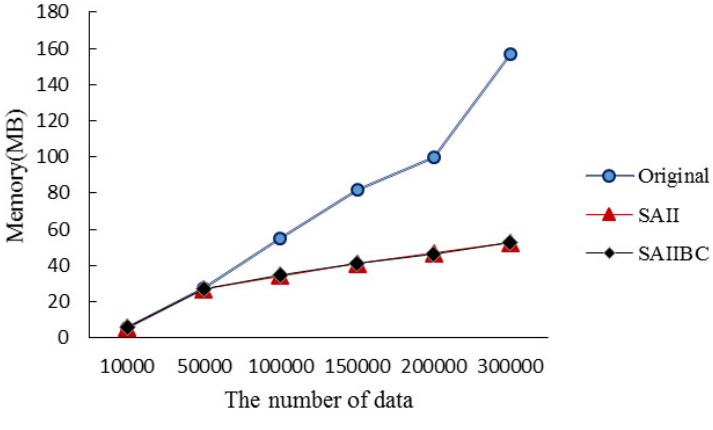


Fig. 9. The comparison results of memory requirement.

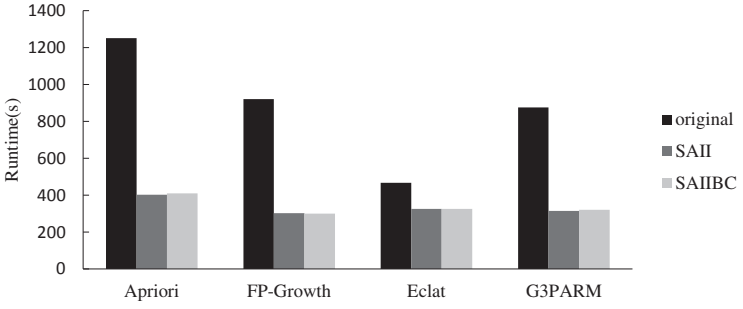


Fig. 10. The comparison results for runtime.

improved Eclat algorithm[4] and G3PARM algorithm based on genetic algorithm theory,[13] with the support of 0.25, and the confidence of 0.5. A1 is used as the dataset. The experimental results are shown in Fig. 10.

The experimental results show that compared to the traditional approach, SAII and SAIIBC are more effective in accelerating the various association analysis algorithms. Moreover, the acceleration effect of Eclat is small due to the original higher efficiency, which also shows that it is difficult to enhance its performance for the efficient algorithm.

SAII has the advantage that it can ensure that the added data is closest to the most similar data. But the shortcomings are more obvious, i.e. the time complexity of SAII is $O(n^2)$ which is higher than ours. In the case of larger scale data, time consumption has an exponential growth trend. Our method is compared with the existing k group, so the time complexity of this section is $O(kn)$. Moreover, the amount of data in a group is limited, which is at most M due to the load balancing. Therefore, our time complexity is $O((k+M)n)$, where k and M are constants.

In summary, the method of similarity in group is adopted in our method through the clustering method to improve the performance of the shuffling process, when facing large-scale datasets.

5. Conclusions

This paper first introduces the existing AR mining algorithm improvement strategy, and discusses the problem of improvement strategies for big data for the police business system. Especially for the general speeding-up AR algorithm, to overcome this drawback, an improved method based on similarity between groups using dynamic data insertion algorithm is proposed. Based on the clustering algorithm, our algorithm flow is improved by thresholds and dynamic grouping strategy to reduce the time and the calculation of the original insertion algorithm. Moreover, Apache Spark is used in the experiment to get more beneficial of the parallel algorithm. The experimental results on police business big data show that our

presented approach SAIIBC has a better performance concerning the speed-up and scale-up of implementations.

This paper only focuses on how to improve the efficiency of AR mining in the police business system. In addition, there are some other problems such as face recognition and keyword search. In future work, a new methodology based on machine learning should be presented to enhance the performance of other types of business issues, using the rapid development of police business affairs big data.

Acknowledgments

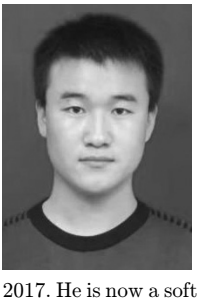
This work is supported by the Ministry of Science and Technology Department Foundation of Sichuan Province (Nos. 2016FZ0108, 2017JY0027, 2018JY0067 and 2017GFW0128) and by the Natural Science Foundation of Guangdong Province, China (No. 2017A030313380) and by the China Scholarship Council.

References

1. D. Abadi, S. Madden and M. Ferreira, Integrating compression and execution in column-oriented database systems, in *Proc. ACM SIGMOD Int. Conf. Manage. Data* (Chicago, IL, USA, 2006), pp. 671–682.
2. R. Agrawal, Mining association rules between sets of items in large databases, *ACM Sigmod Record* **22**(2) (1993) 207–216.
3. A. AitMlouk, T. Agouti and F. Gharnati, Mining and prioritization of association rules for big data: Multicriteria decision analysis approach, *J. Big Data* **4**(42) (2017) 1–21.
4. C. Borgelt, Efficient implementations of Apriori and Eclat, in *Proc. IEEE ICDM Workshop Frequent Item Set Mining Implementations, CEUR Workshop Proceedings* (Florida, USA, 2003), pp. 90–98.
5. X. Chen, S. Zhang, H. Tong and X. Cui, FP-growth algorithm based on boolean matrix and MapReduce, *J. South China Univ. Technol. (Natural Science Edition) (1)* **42** (2014) 135–141 (in Chinese).
6. Z. Deng, Z. Wang and J. Jiang, A new algorithm for fast mining frequent itemsets using N-lists, *Sci. China Inf. Sci.* **55**(9) (2012) 2008–2030.
7. Z. Deng and S. Lv, Fast mining frequent itemsets using Nodesets, *Expert Syst. Appl.* **41**(10) (2014) 4505–4512.
8. P. Gregory, Discovery, analysis, and presentation of strong rules, in *Knowledge Discovery in Databases*, Piatetsky-Shapiro, Gregory and Frawley, William J. eds. (AAAI/MIT Press, Cambridge, MA, 1991).
9. J. Han, J. Pei and Y. Yu *et al.*, Mining frequent patterns without candidate generation: A frequent-pattern tree approach, *Data Mining and Knowl. Discovery* **8**(1) (2004) 53–87.
10. L. Kuang, Z. Liao, W. Feng *et al.*, Multimedia services quality prediction based on the association mining between context and QoS properties, *Signal Process.* **120**(C) (2016) 767–776.
11. G. Lee, U. Yun, H. Ryang *et al.*, Approximate maximal frequent pattern mining with weight conditions and error tolerance, *Int. J. Pattern Recognit. Artif. Intell.* **30**(6) (2016) 1650012.
12. R. W. P. Luk and W. Lam, Efficient in-memory extensible inverted file, *Inf. Syst.* **32**(5) (2007) 733–754.

13. J. M. Luna, J. R. Romero and S. Ventura, Design and behavior study of a grammar-guided genetic programming algorithm for mining association rules, *Knowl. Inf. Syst.* **32**(1) (2012) 53–76.
14. J. M. Luna, A. Cano, M. Pechenizkiy *et al.*, Speeding-up association rule mining with inverted index compression, *IEEE Trans. Cybern.* **46**(12) (2016) 3059–3072.
15. M. H. Mohamed and M. M. Darwieesh, Efficient mining frequent itemsets algorithms, *Int. J. Mach. Learn. Cybern.* **5**(6) (2014) 823–833.
16. G. N. Pradhan and B. Prabhakaran, Association rule mining in multiple, multidimensional time series medical data, *J. Healthc. Inf. Res.* **1**(1) (2017) 92–118.
17. H. Tian and X. Wang, A novel privacy-preserving association rules mining method, *Int. J. Pattern Recognit. Artif. Intell.* **24**(6) (2010) 995–1009.
18. C. Xiang, S. Su, S. Xu *et al.*, DP-Apriori: A differentially private frequent itemset mining algorithm based on transaction splitting, *Comput. Secur.* **50** (2015) 74–90.
19. F. Xu and H. Lu, The application of FP-growth algorithm based on distributed intelligence in wisdom medical treatment, *Int. J. Pattern Recognit. Artif. Intell.* **31**(4) (2017) 1759005.
20. Y. Xun, J. Zhang and Q. Xiao, FiDoop: parallel mining of frequent itemsets using MapReduce, *IEEE Trans. Syst. Man Cybern. Syst.* **46**(3) (2017) 313–325.
21. M. J. Zaki, Scalable algorithms for association mining, *IEEE Trans. Knowl. Data Eng.* **12**(3) (2000) 372–390.

Guiduo Duan received her B.S. degree from the Sichuan Normal University, China in 2003, M.S. and Ph.D. degrees from the University of Electronic Science and Technology of China (UESTC) in 2006 and 2009, respectively. She was in watermarking workshop of the University of Surrey in the UK for research work from October 2007 to October 2008. She is now Associate Professor of UESTC. Currently, she is in the Humboldt University of Berlin as a Visiting Scholar. Her research interests include digital watermarking, machine learning and its applications. She has published more than 20 papers in the international journals and conferences, as well as three books. She has been granted three patents.



Xiaotong Wang received his B.S. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2014 and M.S. degree from the University of Electronic Science and Technology of China (UESTC) in 2017. He is now a software engineer in the CETC Ocean Information Co., Ltd. His research interests are realtime-stream computing, machine learning and big-data processing.

Tianxi Huang received the B.S degree and M.S. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2003 and 2006, respectively. He is now Lecturer in the Department of fundamental Courses, Chengdu Textile College. From 2014 to 2016, he worked in the Huawei Chengdu Institute as delivery project leader and had been responsible for storage product development. From 2006 to 2014, he worked in Hikvision as a project manager and had been responsible for the development of audio and video codec algorithms and algorithm optimization. His research interests are big data, image processing, and video encoding and decoding.

Jürgen Kurths is a German physicist and mathematician. He is a Chair of the research domain Transdisciplinary Concepts of the Potsdam Institute for Climate Impact Research, a Professor of Nonlinear Dynamics at the Institute of Physics at the Humboldt University, Berlin, and a 6th-century chair for Complex Systems Biology at the Institute for Complex Systems and Mathematical Biology at Kings College, Aberdeen University (UK). Kurths studied mathematics at the University of Rostock and was awarded his PhD in 1983 at the GDR Academy of Sciences, followed by his installation in 1991 in theoretical physics at the University of Rostock. His research is mainly concerned with nonlinear physics and complex systems sciences and their applications to challenging problems in Earth system, physiology, systems biology and engineering. He has authored or coauthored more than 500 papers that are cited more than 18,000 times (Hindex: 57).