Peer-to-Peer Networking and Applications (2019) 12:1433–1444
https://doi.org/10.1007/s12083-018-0698-1

PPQAR: Parallel PSO for quantitative association rule mining

Danfeng Yan[1] · Xuan Zhao[1] · Rongheng Lin[1] · Demeng Bai[2]

Received: 7 March 2018 / Accepted: 2 November 2018 / Published online: 13 February 2019

Abstract
Mining quantitative association rules is one of the most important tasks in data mining and exists in many real-world problems. Many researches have proved that particle swarm optimization(PSO) algorithm is suitable for quantitative association rule mining (ARM) and there are many successful cases in different fields. However, the method becomes inefficient even unavailable on huge datasets. This paper proposes a parallel PSO for quantitative association rule mining(PPQAR). The parallel algorithm designs two methods, particle-oriented and data-oriented parallelization, to fit different application scenarios. Experiments were conducted to evaluate these two methods. Results show that particle-oriented parallelization has a higher speedup, and data-oriented method is more general on large datasets.

Keywords Particle swarm optimization · Association rule mining · Spark

1 Introduction

Association rule mining, as one of the main tasks of knowledge discovery in database, was proposed by Agrawal et al. [1] in 1993, in order to extract frequency item sets as well as the hidden correlations, i.e. association rules, among them from transactions in database [2]. Up to now, there have been many algorithms for the task and many improvements have been proposed to optimize the accuracy and effectiveness of these algorithms [3, 4].

In many real world databases, the data may contain both categorical and numerical attributes, and ARM on this kind of data is known as quantitative association rule mining [5, 6]. It becomes complicated and difficult when comes to mine quantitative association rules and some special techniques need to be employed. Recently, evolutionary algorithms, including genetic algorithm (GA), particle swarm optimization algorithm and differential evolution (DE) algorithm, have been well researched and successfully applied to ARM problems [7–9]. Besides them, PSO algorithm has been proved effective and well performed for quantitative association rule mining [10]. However, PSO is a kind of iteration algorithm and might be inefficient on huge datasets. In practical tasks, such as stock market research and transportation volume analysis, datasets are so huge that single computer cannot satisfy the requirements of computing resources. Thus, this paper proposes a parallel PSO algorithm for quantitative association rule mining, i.e. PPQAR, which can run on a distribute cluster and get a remarkable efficiency improvement than the serial version.

Some previous works paid attention to parallelizing classic ARM algorithms like Apriori and FP-Growth based on Hadoop Map-Reduce model or Spark RDD framework. For example, Yu H et al. [11] proposed an improved Apriori algorithm based on the boolean matrix and Hadoop. She X and Zhang L [12] also proposed an Apriori parallel improved algorithm based on Map-Reduce distributed architecture. Joy R and Sherly K [13] introduced a parallel frequent itemset mining approach with spark RDD framework for disease prediction. Zhang D et al. [14] proposed a parallel FP-Growth method for query recommendation, which is included in the Spark's machine learning library MLlib. Rathee S and

This article is part of the Topical Collection: *Special Issue on Big Data and Smart Computing in Network Systems*
Guest Editors: Jiming Chen, Kaoru Ota, Lu Wang, and Jianping He

✉ Danfeng Yan
yandf@bupt.edu.cn

Xuan Zhao
zhaoxuan@bupt.edu.cn

Rongheng Lin
rhlin@bupt.edu.cn

Demeng Bai
baidemeng0119@126.com

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China

[2] State Grid ShanDong Electric Power Research Institute, Jinan, China

Kashyap A [15] studied exploiting Apache Flink's iteration capabilities for distributed Apriori, and took community detection problem as an example. However, these methods were all designed for problems with boolean or categorical data such as market-basket analysis. When facing the quantitative association rule mining problems, these methods need to discretize the numerical attributes in the preprocessing step, and different discretization strategies may lead to completely different results.

PSO algorithm is able to find the proper interval of each attribute by learning from the given dataset, without the need for an apriori discretization [16, 17]. Besides, as a multi-objective optimization algorithm, it could find the appropriate candidate association rules without specifying the minimum support, minimum confidence or other metrics [18]. In recent years, many PSO based quantitative association rule mining methods have been proposed and applied to solving real-world problems [19]. Most of these approaches tried to optimize the mining results and the effectiveness of the algorithm [20–22]. Feng et al. [23] utilized the particle swarm optimization algorithm to improve the performance of association rule features in data mining. Jiang et al. [24] introduced a multi-objective PSO approach to generate association rules that depict the relationship between affective dimensions and design attributes based on online customer reviews. Qian et al. [25] proposed a new improved PSO based ARM method to avoid the local optimal solution by controlling the particle velocity. However, few of them focused on the optimization of the computational process. Since the evolutionary algorithms are all iterative and hard to parallelize, there are few previous works focused on the parallelization of these algorithms. Thus, this paper worked to parallelize the PSO algorithm for ARM to make the algorithm more scalable, more efficient and be able to deal with huge datasets in practical problems.

The main contributions of this paper includes:

(1) We utilize the PSO algorithm to mine quantitative association rules and take the ARM as a multi-objective optimization problem. In this way, the threshold of the optimization objectives need not to be specified and the numerical attributes need not to be discretized first.
(2) We design the parallel optimization of the PSO algorithm for quantitative association rule mining. Two strategies, called particle-oriented and data-oriented respectively, are proposed for different application scenarios.
(3) We design the proposed method based on Spark framework.

The rest of the paper is organized as follows. Section 2 introduces the background. Section 3 presents the design and implementation of the proposed approach in detail. Section 4 introduces the conducted experiments and analyzes the results. Finally, section 5 concludes the paper.

2 Background

Before introducing the proposed parallel algorithm, some basic concepts, including the definitions of association rule, origin PSO algorithm and its usage in multi-objective optimization, are briefly restated in this section.

2.1 Association rule

Let $I = \{i_1, i_2, ..., i_m\}$ be the set of m different items. A set $X \subseteq I$ is termed an itemset or pattern. In particular, an itemset X containing k items is called a k-itemset. Let $T = \{t_1, t_2, ..., t_n\}$ be the set of all possible transactions. A transaction database D is a set of transactions, such that $D \subseteq T$. Each transaction has a unique transaction ID (*tid*). A transaction $t = (tid, X)$ is a tuple (or record) where X is an itemset. A transaction $t = (tid, X)$ is termed to include itemset Y if $Y \subseteq X$. An association rule is an implication in form of $X \rightarrow Y$, where $X, Y \subset I$ and $X \cap Y = \phi$, and X is called the antecedent and Y is termed of consequent, and the rule means X implies Y.

The two fundamental parameters used in conventional ARM methods are support and confidence. Support of an association rule $X \rightarrow Y$ is obtained as the percentage of transactions in D that contain $X \cup Y$ to the entire number of transactions. Support indicates the statistical importance of association rules. The confidence of the association rule is characterised as the percentage of transactions in D containing $X \cup Y$ to the total number of transactions that contain X.

Particularly, in quantitative association rule mining problems, attributes are numerical so that items stand for the specific intervals of each attribute. One of the main tasks is to find the proper interval for each numerical attribute. There are two major approaches to solve this problem. One of the approach is to discretize the numerical attributes firstly, then take each interval as the item and employ the traditional association rule mining algorithm such as Apriori and FP-Growth. The commonly used discretization methods include equal width method, equal depth method, clustering based discretization algorithm [26] and so on. One of the drawbacks of this approach is separating the discretization and the association rule mining into two processes, which is not efficient and may not obtain the best results. The other approach is to employ the evolutionary algorithms, including GA, PSO and DE, and find the appropriate boundary of the interval automatically during the iterations. Since the discretization affects the results of association rules mining to a great extent, the second approach is more commonly used and has been proved to be effective [7, 9, 10].

2.2 Particle swarm optimization

The particle swarm optimization algorithm is a stochastic optimization technique proposed by Eberhart and Kennedy in

1995 [27], which optimizes an objective function by improving the candidate solutions iteratively. In PSO algorithm, each particle has two attributes, the position and the velocity, and represents a candidate solution to the problem. During the search process, each particle moves to its next position according to its own experience and the experience of the whole particles in the swarm.

Let $D \in R^N$ be the search space with N dimensions and k be the number of particles. The main process of PSO algorithm is as follows:

(1) For each particle, initialize its position with an N-dimensional vector $X_i = (x_{i1}, x_{i2}, ..., x_{iN})$, $i = 1, 2, ..., k$, $x_{in} \in [L_n, U_n]$, $1 \leq n \leq N$, where L_n and U_n are the lower and upper bounds of the n-th dimension, and set X_i as the particle's local best position $lbest_i \leftarrow X_i$. Then calculate the fitness value for all particles and initialize the global best position of the swarm $gbest \leftarrow X_i$, if $f(gbest) \geq f(X_i)$ for all values of $i \in [1, k]$, where f is the objective function to be maximized. Meanwhile, initialize the particle's velocity with zero vector.

(2) Update the iteration counter, until meet the predefined maximum iterations.

(3) Update the position and the velocity of each particle according to the following equations:

$$V_i(t+1) = w^* V_i(t) + C_1{}^* r_1{}^* (lbest_i(t) - X_i(t)) + C_2{}^* r_2{}^* (gbest(t) - X_i(t)) \quad (1)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2)$$

where $1 \leq i \leq k$, w is termed the inertia weight, C_1 and C_2 are two constant values called cognitive and social factors, r_1 and r_2 are picked from uniform distribution [0,1], aiming to bring stochastic state to the algorithm.

(4) For each value of $i \in [1, k]$, update the local best position $lbest_i$ if $f(lbest_i) \leq f(X_i)$, and update the global best position $gbest$ if $\exists j \in [1, k]$, $f(gbest) \leq f(X_j)$ and $f(X_j) \geq f(X_i)$ for all values of $i \in [1, k]$.

(5) Back to step (2).

2.3 Multi-objective optimization

PSO algorithm is also usually employed in multi-objective optimization problems. In this situation, more than one objectives need to be optimized and some special techniques have to be taken to obtain the trade-off between different goals.

There are mainly three approaches to realize the optimization of multi-objectives, that is the weighted sum method, the lexicographic method and the Pareto method [16]. In the weighted sum method, different objectives are weighted linear summed and the result is used for comparison. The lexicographic method prioritizes the optimization objectives firstly, then each objective is compared sequentially. However, these two methods have some drawbacks. The weighted sum method sets the weight coefficient for each optimization objective, and the value of these parameters is usually determined according to experts' previous experience and lacks of objectivity. This shortcoming also exists in the lexicographic method. It is difficult to determine the priority of different optimization objectives and different priority orders may lead to completely different comparison results. By contrast, the Pareto method overcomes the limitation and obtains a trade-off among different optimization objectives.

In the Pareto method, a candidate solution is considered to dominate another candidate solution if it is better for at least one optimization objective and not worse for all the other objectives at the same time [28, 29]. Figure 1 illustrates an example of the Pareto method, where O_1 and O_2 are two objectives to be optimized, and c_1 to c_4 are four candidate solutions. According to the definition, we can conclude that c_1 is dominated by c_2 and c_3 is dominated by c_4. Since neither c_1 nor c_4 is dominated by the other candidate solution, they are termed non-dominated candidate solutions and both accepted as Pareto Optimal Front.

In multi-objective PSO algorithm, the comparison of the objective function often takes Pareto dominance into account when moving the particles, and the non-dominated candidate rules are stored in the local best position set and the global best position set. When applying PSO algorithm to mining quantitative association rules, two key issues are how to represent the candidate rule with the particle and how to design the fitness function. The particle encoding methods and the commonly used metrics are detailedly illustrated in next section.

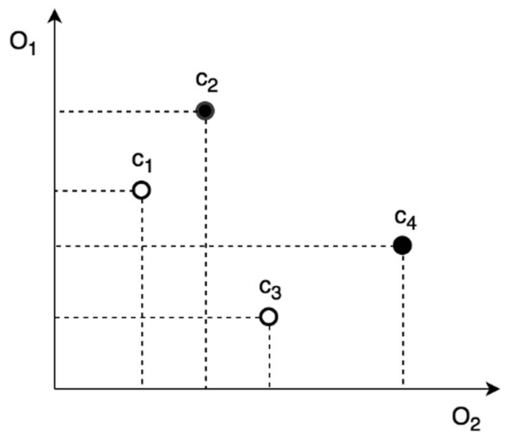

Fig. 1 Example of the pareto method

3 PPQAR

3.1 Rule representation

PSO algorithm can be applied to mining association rules by designing the particle position vector and the fitness function. There are two methods to represent a rule as particle position, which are inspired from chromosome representations in the genetic algorithms. The first is Pittsburgh approach, where each particle or chromosome represents a set of rules. The second is called Michigan method, in which each particle represents a separate rule. These two methods have different application scenarios, and decisions should be made depends on the main task of the problem. The Pittsburgh method pays attention to the features of the whole rules set instead of evaluating the quality of a single rule. On the contrary, the Michigan method has simpler and shorter encoding form, and is easier to calculate optimization objectives on each individual rule. In this work, the Michigan approach is followed and there is an one-to-one correspondence between candidate rule and particle position.

There are also different approaches to encode a single rule as a particle. Binary encoding and attribute interval representation are two commonly used method. The first approach is more suitable to handle categorical data and single-dimension ARM problems like market-basket analysis. By contrast, the second approach is used to process the fixed-length data with aptotic attributes, and is more suitable for numerical attributes. In this paper, the attribute interval representation method introduced by Beiranvand V et al. [16]. is used to encode the rule to a particle.

In detail, as shown in Table 1, each attribute in the particle consists of three parts: the mark field (Mark), the lower bound (L) and the upper bound (U). The first part of an attribute is a mark indicates which part of the rule the current attribute belongs to: the antecedent part, the consequent part, or none of them which means this attribute does not appear in the rule. This part of the attribute takes the values in three separate intervals: if $Mark_i \in [0,\ 0.33]$, the attribute is involved in the antecedent of the rule; if $Mark_i \in [0.34,\ 0.66]$, the attribute is involved in the consequent of the rule; and if $Mark_i \in [0.67,\ 1.00]$, the attribute does not appear in the rule, at all. The antecedent section of a rule includes all those attributes in the particle containing a value in the range of [0.00, 0.33] as the first part of the attribute, while the consequent section includes all the attributes containing a value in the range of [0.34, 0.66] as the first part of the attribute. The second and the third parts of an attribute in the particle representation specify the lower bound L_i and the upper bound U_i values of that attribute in the dataset. As the PSO algorithm evolves these intervals are adjusted so as to find the most appropriate interval for each attribute [17, 18].

Table 1 Rule representation as a particle

Attribute_1			Attribute_2			… Attribute_i …			Attribute_n		
Mark_1	L_1	U_1	Mark_2	L_2	U_2	Mark_i	L_i	U_i	Mark_n	L_n	U_n

3.2 Fitness function

The proposed method uses four measures as optimization objectives to evaluate the quality of each particle, viz., candidate solution, including support, confidence, comprehensibility and interestingness.

As mentioned above, when it comes to numerical attributes, support of $A \in [L_A,\ U_A]$ where L and U are the lower and upper bounds of specified interval is defined in (3). Accordingly, the confidence of rule $A \rightarrow C$ is defined in (4).

$$Support(A) = \frac{number\ of\ records\ where\ A \in [L_A U_A]}{number\ of\ records\ in\ the\ database} \tag{3}$$

$$Confidence = Support(A \cup C)/Support(A) \tag{4}$$

Many research results have shown that the classical framework performs well in most cases [30–32], but there is still room for improvement. The support and confidence measures guarantee the objectivity and reliability of a rule, in addition, a rule should also be comprehensible and interesting. However, comprehensibility and interestingness of a rule are very abstract and hard to quantify. According to Ghosh and Nath's research [33], the less number of conditions in the antecedent part of a rule would be, the more comprehensible or understandable is that rule. Meanwhile, the research used a method based on the support of both antecedent and consequent parts to measure the interestingness of the rule [34]. These two measures are calculated by (5) and (6).

$$Comprehensibility = log(1 + |C|)/log(1 + |A \cup C|) \tag{5}$$

$$Interestingness = [Sup(A \cup C)/Sup(A)] \\ {}^{*}[Sup(A \cup C)/Sup]*[1 - Sup(A \cup C)/|D|] \tag{6}$$

where Sup is the support count of the conditions and $|D|$ is the total number of records in the database.

All the four objectives evaluate a candidate rule from different angles. In summary, the paper designs the metric vector shown in (7) as the fitness of PSO algorithm, where Sup, Con, Com, Int are short for Support, Confidence, Comprehensibility and Interestingness, respectively.

$$Fitness = (Sup, Con, Com, Int) \tag{7}$$

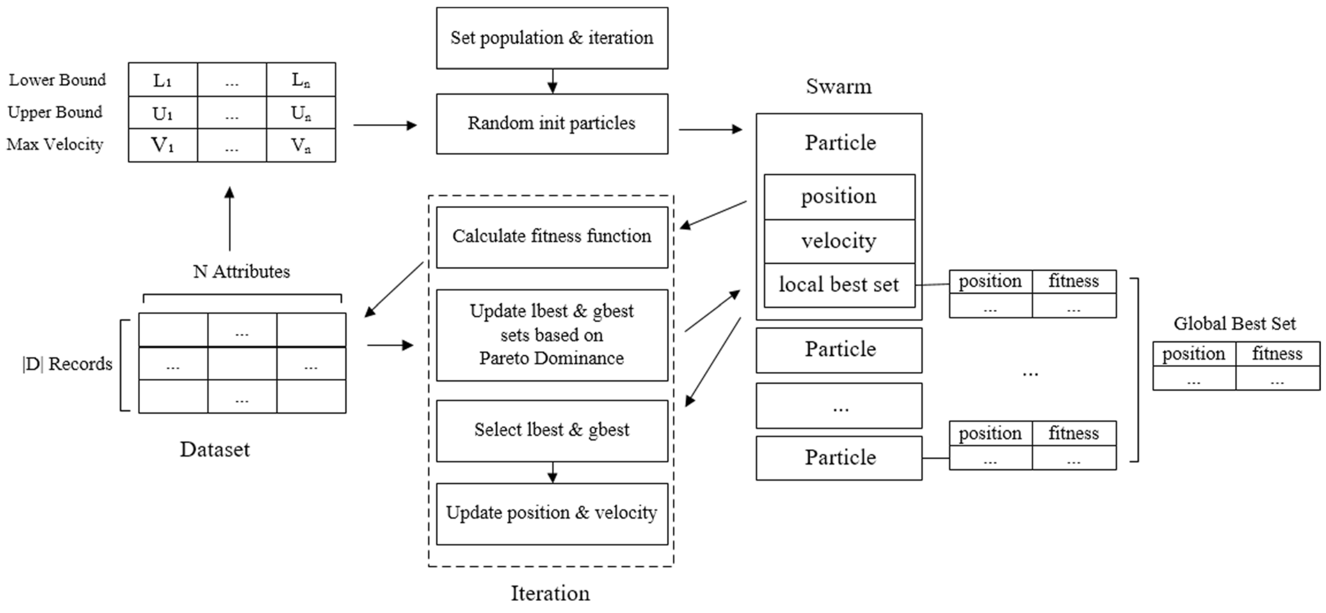

Fig. 2 The particle-oriented parallelization method

3.3 PSO for ARM

The overall framework of PSO algorithm for quantitative association rule mining problems is shown in Fig. 2.

Firstly, the value of two parameters, i.e. particle population and maximum iteration, are set. Then, records are imported from the database and for each attribute in the record, calculate the lower and upper bounds as the limit of the particle position.

Then the PSO algorithm begins iterating until meet the stop criteria, i.e. maximum iterations. Since the PSO algorithm is easy to fall into local optimum, the particles are initialized randomly in the range of attribute values, instead of assigning the same initial position.

During each iteration, the fitness function vector consists of four objectives is calculated for all the particles. Then both local best value set lbset and global best value set gbest are updated based on Pareto dominance. More concretely, if the fitness f_i of the particle's current position p_i is higher than the fitness f_j of some best known position p_j in the local or global best set on all the four metrics, which means the candidate rule represented by p_i dominates the rule represented by p_j, then delete p_j from the local or global best set and add p_i as the new best known position. By contrast, if f_i is lower than the fitness f_j of some best known position p_j in the local or global best set on all the four metrics, indicating that the candidate rule represented by p_i is dominated by the rule represented by p_j, then just finish the comparison and move to next iteration. Otherwise, the particle's current position p_i is a new non-dominated candidate solution and should be added to the local or global best set.

To update the particles' velocity and position, certain items in the local best set and global best set have to be selected to be put into the formula, and there the Roulette wheel selection is applied. Specifically, for a non-dominated best known position p_i, the probability of being selected is calculated by (8), where f_i is the fitness value of position p_i and the denominator is the sum of the fitness value of particle positions stored in the local best set and global best set. Since the fitness is a four-dimensional vector, the product of the four measures is used as the fitness value.

$$Probability(p_i) = \frac{f_i}{\sum_{p_j \ in \ set} f_j} \tag{8}$$

The Roulette wheel selection method ensures the position with better fitness value has higher chance of being selected.

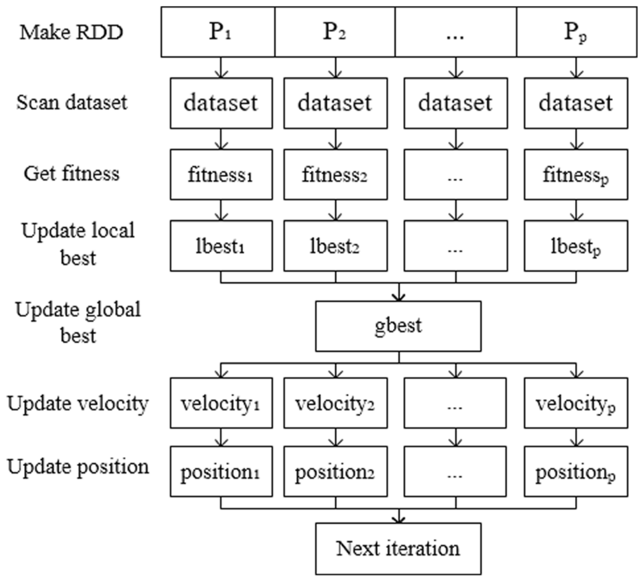

Fig. 3 The particle-oriented parallelization method

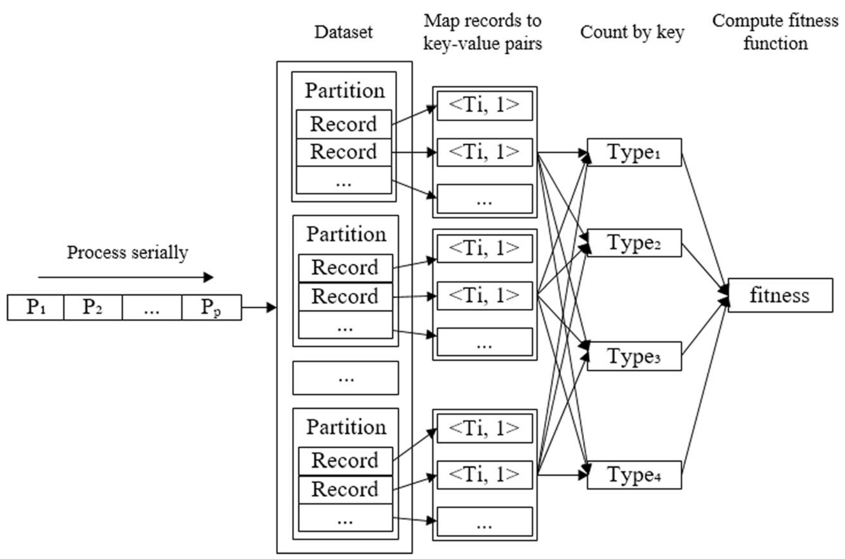

Fig. 4 The data-oriented parallelization method

In this way, the particle as well as the whole swarm can converge to the optimum value more quickly.

Moreover, to control excessive roaming of particles outside the search space, the value of each component in the velocity vector is clamped to the predefined range. Let $V_{max} = \{v_{max1}, v_{max2}, \ldots, v_{maxn}\}$ be the maximum velocity, where v_{maxi}, $i \in [1, n]$ represents the velocity component on attribute $a_i \in [L_i, U_i]$ and is calculated by (8).

$$v_{maxi} = (U_i - L_i)/K \tag{9}$$

Here K is termed the limitation factor, which is a constant value set by user according to the specific data feature. When the velocity component v_i exceeds the limit, it is rectified according to (10).

$$v_i = r^* v_{maxi} \tag{10}$$

where r is randomly picked from uniform distribution [0,1]. Similarly, the particle position is also fixed when exceeds the bound during each iteration.

3.4 Parallel optimization

As an iterative algorithm, PSO algorithm requires previous results to finish calculation in each iteration, so the algorithm must execute in serial between iterations. In this situation, this paper aims at parallelizing the stages in each iteration and reducing time consumption.

Table 2 The statistics of the test dataset

Records number	Indexes number	Type
20 million	12	numerical

Previous section and Fig. 2 show that many stages of the algorithm can be executed in parallel. For example, procedures like calculating the fitness function and updating the position, velocity and local best set value of each particle can be processed separately. We propose the particle-oriented approach for large number of particles and relatively small datasets. In view of huge datasets, the data-oriented parallelization method is present to split the whole dataset into partitions. So the two proposed methods in this paper are adopted in different application scenarios and we will discuss it in detail.

Let $|D|$ be the number of records in the database, N be the number of attributes, the complexity of the algorithm is equal to $O(N^*|D|)$. Moreover, with specific particle population P and iteration number I, the complexity is equal to $O(N^*|D|^*P^*I)$, which grows linearly with each parameter increasing. The fitness function calculation step needs to scan the whole dataset to get values of optimization objects, so its complexity depends on the scale of dataset. Since the complexities of other steps are only related to the scale of swarm and iterations, it is obvious that the fitness calculation step is the main part of the whole time consumption in each iteration. Therefore, this paper focuses on computing the fitness function in parallel and proposes two strategies, termed particle-oriented and data-oriented parallelization, respectively. Brief illustration of these two methods are shown in Fig. 3 and Fig. 4.

As shown in Fig. 3, the particle-oriented method treats each particle as the computing unit and calculate each particle's fitness function in parallel. For each particle, the algorithm scan the dataset and run just like the serial version.

Based on Spark RDD framework, the algorithm create an RDD *particleRDD* = $[P_1, P_2, \ldots, P_p]$ to save the

Table 3 Sample records

Sample number	P	Q	I	WDIRECTION	WSPEED	GDIRECTION	GSPEED	HUMIDITY	TEMPERATURE	PRESSURE	MDIRECTION	MSPEED
1	301.55	−56.65	314.80	12	6.4	343	9.1	67	26.2	990.8	359	9.3
2	310.69	−54.82	325.35	67	5.1	45	8.5	61	30.1	996.4	63	5.2
3	−555.0	6.09	573.18	111	2.6	94	3.9	45	31.6	999.7	97	2.3

whole swarm, where p is the particles number and P_i, $1 \le i \le p$ is a particle. When the algorithm begins, it firstly loads the dataset into the memory from file on each worker node, and the dataset is resident during the whole process. During each iteration, the algorithm distributes the particles to different worker nodes, and all the computing tasks of each particle is finished on the same node. Algorithm 1 presents the pseudo code of this method.

```
Algorithm 1 Particle-Oriented Parallelization
Input: $D$, dataset of fixed-length records with aptotic
        attributes
Input: $p$, particles number
Input: $I$, maximum iterations
Output: Association rules set

Begin
  Load $D$ into memory;
  Make RDD $particleRDD=[P_1, P_2, ..., P_p]$
  for $1 \leqslant i \leqslant p$ do init $P_i$;
  while $i < I$ do
    for $P_j$ in $particleRDD$ do in parallel
      Scan $D$ and compute fitness;
        for $lbest_j^k$ in $lbest_j$ do
          if $P_j$ Pareto dominates $lbest_j^k$ then
            Update $lbest_j$ set;
    Collect $particleRDD \rightarrow particles$;
      for $P_j$ in $particles$ do
        for $lbest_j^k$, $gbest_{k'}$ in $lbest_j$ and $gbest$ do
          if $lbest_j^k$ Pareto dominates $gbest_{k'}$ then
            Update $gbest$ set;
      Broadcast $gbest$ set;
      for $P_j$ in $particleRDD$ do in parallel
        Update $velocity_j$ and $position_j$;
      $i \leftarrow i + 1$;
  Output($gbest$)
End
```

Assume that there are enough computing resources, including CPU cores and memory, this method can reduce the complexity from $O(N*|D|*P)$ to $O(N*|D|)$ theoretically. Moreover, the particle-oriented approach has strong flexibility and can be applied on clusters with different number of worker nodes by adjusting the number of partitions in the particle RDD.

The particle-oriented parallelization method is more suitable for large particles number and relatively small datasets. Since only particles information is transmitted in network in the following iterations, this method has very small network communication overhead. Besides, by caching the dataset in memory, there is no I/O cost during iterations, which makes it also applicable for large iteration number.

The particle-oriented approach can get a linear speed-up compared to the origin serial algorithm. However, there are some deficiencies in this method. When it comes to huge datasets, the single worker node may not be able to load the whole dataset into the memory. Moreover, the particle-oriented method is unable to make full use of the computing resources when the particle number is close to or even less than the number of worker nodes. Therefore, a more scalable and general method is proposed to overcome these disadvantages.

The data-oriented parallelization method updates particles' positions, velocities and local best sets in the same way as the particle-oriented parallelization method. But in the fitness function computing step, this method splits the whole dataset into partitions and each partition is treated as a computing unit. The swarm is processed serially. For each particle, the algorithm computes the fitness function on every partitions in parallel, then results on every partition are collected and merged to get the final fitness value.

In this method, the records in the dataset are organized as an RDD: $dataRDD = [R_1, R_2, ..., R_d]$, where d is the number of records in dataset, and R_i, $1 \le i \le d$ is a record. The relation between each record and the rule

Table 4 The parameters settings

Parameter	Value	Description
w	0.8	Inertia weight
C_1, C_2	2	Cognitive and social factors
K	20	Limitaton factor

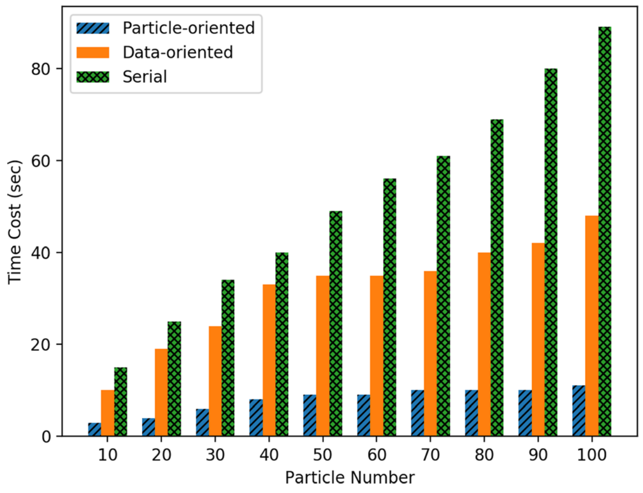

Fig. 5 Time consumption of three methods

represented by the particle's position can be divided into four conditions: the record satisfies both antecedent and consequent conditions, only satisfies antecedent condition, only satisfies consequent condition and satisfies neither antecedent nor consequent condition. The algorithm calculates the optimization objectives by counting the number of records of these four types. Let T_i, $1 \leq T_i \leq 4$ indicates the relation type, map each record R_i to a key-value pair $<T_i, 1>$, then it is able to get records number of each type by applying the reduce action. Figure 4 demonstrates the fitness calculation step and Algorithm 2 describes the data-oriented method in detail.

Therefore, the particle-oriented parallelization method is efficient when the datasets are small and the number of particles is large, while the data-oriented

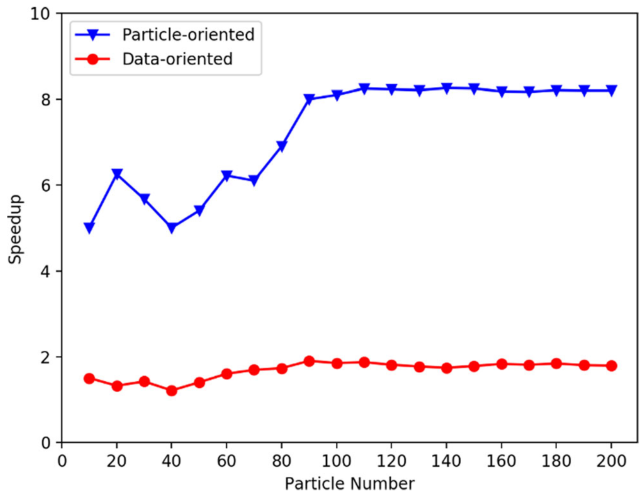

Fig. 6 Speedup comparison between two proposed methods

method has a better performance on the huge datasets. Meanwhile, it has a fine scalability because the efficiency can be raised by simply adding more compute nodes to the cluster.

Algorithm 2 Data-Oriented Parallelization

```
Input: $D$, dataset of fixed-length records with aptotic
       attributes
Input: $p$, particles number
Input: $I$, maximum iterations
Output: Association rules set

Begin
  Make RDD $dataRDD=[R_1, R_2, ..., R_d]$, $1 \leqslant d \leqslant |D|$;
  for $1 \leqslant i \leqslant p$ do init $P_i$;
  while $i < I$ do
    for $1 \leqslant k \leqslant p$ do
      for $R_j$ in $dataRDD$ do in parallel
        Compare $R_j$ with $P_k$'s position $\rightarrow$ $T_j$;
        Map $R_j \rightarrow <T_j, 1>$;
      Reduce by key $<T_j, 1> \rightarrow sum$;
      Compute optimization objectives→fitness;
      for $lbest_k^l$ in $lbest_k$ do
        if $P_k$ Pareto dominates $lbest_k^l$ then
          Update $lbest_k$ set;
          for $gbest_{l'}$ in $gbest$ do
            if $P_k$ Pareto dominates $gbest_{l'}$ then
              Update $gbest$ set;
    for $1 \leqslant k \leqslant p$ do
      Update $velocity_k$ and $position_k$;
    $i \leftarrow i + 1$;
  Output($gbest$)
End
```

4 Experiments and discussion

In this section, a series of experiments were conducted to evaluate the proposed algorithm PPQAR. All experiments run on a Spark cluster with one master and six worker nodes. Each worker node has four CPU cores and 4GB RAM, with an environment of 64bit centOS 7 and Spark 2.0.0.

This paper used equipment defects records in power grid as the test dataset. This dataset is collected from the actual production environment and recorded twelve indexes such as environment temperature, humidity, dissolved gas in oil etc. when device failure happened.

Table 5 Comparison between two methods

Particle number	Time cost (sec)			Speedup	
	Serial	Particle-oriented	Data-oriented	Particle-oriented	Data-oriented
10	15	3	10	5.00	1.50
20	25	4	19	6.25	1.32
30	34	6	24	5.67	1.42
40	40	8	33	5.00	1.21
50	49	9	35	5.40	1.40
60	56	9	35	6.22	1.60
70	61	10	36	6.10	1.69
80	69	10	40	6.90	1.73
90	80	10	42	8.00	1.90
100	89	11	48	8.10	1.85
110	99	12	53	8.25	1.87
120	107	13	59	8.23	1.81
130	115	14	65	8.21	1.77
140	124	15	71	8.26	1.74
150	132	16	74	8.25	1.78
160	139	17	76	8.18	1.83
170	147	18	81	8.17	1.81
180	156	19	85	8.21	1.84
190	164	20	91	8.20	1.80
200	172	21	96	8.20	1.79

And all attributes are numerical. As shown in Table 2, there are 20 million records in test dataset. Table 3 illustrates three sample records.

In our experiments, the hyper parameters in PSO algorithm are set to the values shown in Table 4.

Firstly, we compared two proposed methods with various particles numbers ranging from 10 to 200, and serial version algorithm was used as the benchmark. Since both particle-oriented and data-oriented method parallize the stages in a single iteration, the time consumption of one iteration was recorded. These experiments used dataset with a scale of 2 million lines. Moreover, to avoid randomness, each test was repeated 10 times and took the average of the results. The results are shown in Fig. 5, Fig. 6 and Table 5.

Figure 5 shows that the time costs of all the three methods keep near-linearly increasing with the particle number growing, which is consistent with the theoretical complexity. According to the calculation, we can conclude that the data-oriented parallelization method gets a 180% speedup approximately and the particle-oriented method achieves 600% speedup in average. The results shown in Fig. 6 indicate that the particle-oriented method is more than three times the performance of data-oriented method. Therefore, when the dataset size is not huge or there is enough memory on each compute node, the particle-oriented parallelization approach should be the first choice.

Figure 6 also shows that the speedup of the data-oriented parallelization method is relatively stable on different particle number. It means that the performance of the method has little correlation with the swarm size. By comparison, the particle-oriented method works better on large particle number. This is because there are more particles in a single partition when the particle number grows, and the computation takes higher proportion of the total time cost compared with the communication loss. The result verifies that the particle-oriented approach is more fit for applications with large particle number.

Table 6 The memory consumption of three method

Serial	Particle-oriented	Data-oriented
221 MB	1256 MB	223 MB

Table 7 Time consumption on different dataset sizes

Size (million lines)	2	4	6	8	10	12	14	16	18	20
time cost (sec)	0.22	0.40	0.77	1.31	1.48	1.74	2.16	2.67	2.71	2.95

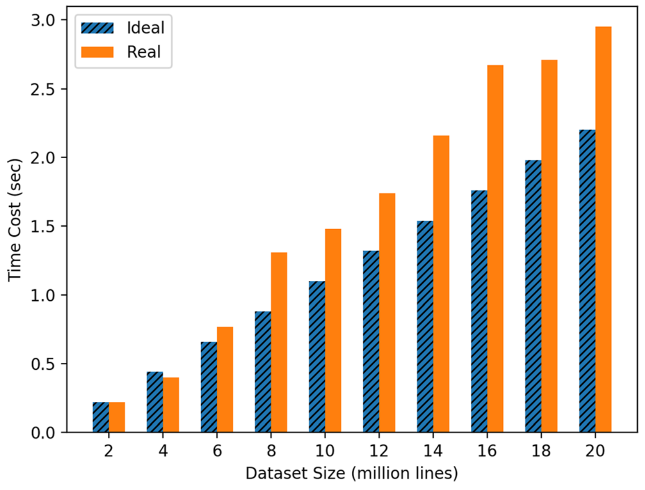

Fig. 7 Experiments on different dataset sizes

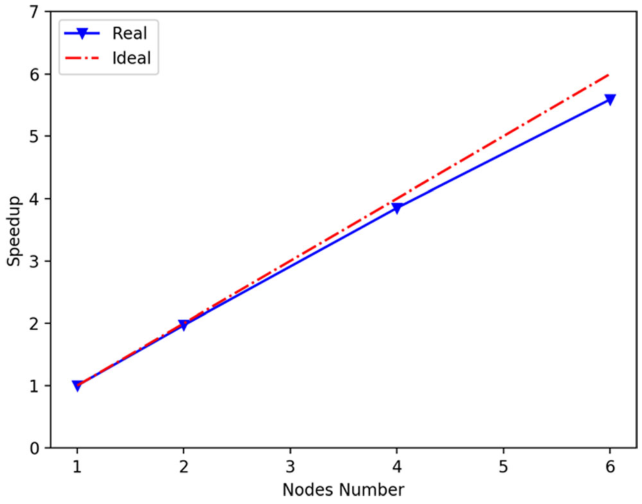

Fig. 8 The speedup on different number of worker nodes

We also record the memory consumption of three methods and the results are shown in Table 6. We can see that the memory cost of serial and the data-oriented method is basically the same. But the particle-oriented method uses almost six times of memory, because it needs to broadcast the dataset to every worker and six worker nodes are utilized in the experiment.

Secondly, we conducted the experiments to evaluate the performance of the data-oriented method. The data-oriented parallelization method is designed for huge datasets which could not be loaded into memory at once. Thus, in this part, the algorithm was tested on different dataset sizes ranging from 2 million to 20 million lines. Time cost of the fitness computing step of one particle was recorded and results are shown in Table 7. The result shows that the algorithm took 0.22 s to calculate the fitness for one particle on dataset with 2 million lines. Assume this as the time consumption unit, a baseline is shown in Fig. 7 and compared with the real time cost. It indicates that with dataset size increasing, the real time consumption has a deviation trend from the ideal baseline. This is because when dataset size grows, the algorithm generates more key-value pairs, which makes the shuffle and reduce steps take more time. This is an optimization goal and would be focused on in future work.

Finally, we validated the scalability of the particle-oriented method. In this part, we used the dataset of 20 million lines and tested the algorithm on clusters consisting of different number of computing nodes. The two main parameters, particle number and iteration, are set to 30 and 100 respectively, and the experiment result is shown in Table 8 and Fig. 8. The result shows that the speedup of the particle-oriented method is approximately proportional to the number of computing nodes. It means that we can improve the efficiency of the algorithm by deploying large scale clusters. Moreover, from Fig. 7, we can find that there is also a small gap between the real speedup and the ideal curve due to the communication loss, and the deviation is more obvious as the cluster scale increasing.

Table 8 The speedup of the particle-oriented method

Nodes number	Time cost (sec)	Speedup (%)
1	3426	100
2	1737	197
4	889	385
6	613	559

Table 9 shows a rule case of the dataset. The mark of attribute P, I is between 0.00 and 0.33, so P and I is the antecedent part; The mark of attribute WSPEED, GDIRECTION and PRESSURE is between 0.33 and 0.66, so these attributes is the consequent part. Other attributes are not contained in the rule. Specially, the rule can be shown as follows:

$$P[-38.8, 671.8], I[356.9, 681.7] \rightarrow WSPEED[0, 3.3], \quad GDIRECTION[0, 99.9], PRESSURE[994.1, 1003.9] \tag{11}$$

Table 9 A Rule case

Attribute	Mark	Lower bound	Upper bound
P	0.05	−38.8	671.8
Q	0.84	−37.8	35.7
I	0.24	356.9	681.7
WDIRECTION	0.71	1	560
WSPEED	*0.55*	*0*	*3.3*
GDIRECTION	*0.59*	*0*	*99.9*
GSPEED	0.68	1.3	6.68
HUMIDITY	0.74	64.2	100
TEMPERATURE	0.70	22.6	28.3
PRESSURE	*0.43*	*994.1*	*1003.9*
MDIRECTION	0.84	4876	9999
MSPEED	0.87	4.5	9.3

where P[−38.8, 671.8] means that the value of attribute P is between −38.8 and 671.8. The support, confidence, comprehensibility and interestingness of the rule are 0.31, 0.63, 0.86, 0.28.

5 Conclusion

This paper presents a parallel PSO algorithm for quantitative association rule mining problems. According to previous approaches, PSO algorithm is used to mine quantitative association rules by encoding the particles' position and defining the optimization objectives as fitness function. In order to improve the efficiency, two parallelization strategies, called particle-oriented and data-oriented respectively, are designed for different application scenarios. Experiments show that the particle-oriented parallelization is more efficient and the data-oriented method is more general and scalable to process huge datasets. Both methods have an obvious speedup compared to the serial algorithm. The algorithm will be deployed to clusters with more computing nodes and applied to large scale analysis on defects records in power grid in future work.

Acknowledgments This paper is supported by National 863 project (No.2015AA050204), State Grid science and technology project (No.520626170011), and Natural Science Foundation of China (No.61532006, No.61320106006).

References

1. R. Agrawal, T. Imielin ski, and A. Swami. 1993. Mining association rules between sets of items in large databases *In Proceedings of the 1993 ACM SIGMOD international conference on management of data*
2. R. Agrawal and R. Srikant. 1994. Fast algorithms for mining association rules in large databases. *In Proceedings of the 20th International Conference on Very Large Data Bases*. 487–499
3. Han J, Pei J, Yin Y, Mao R (2004) Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Min Knowl Disc 8:53–87
4. Manjit Kaur and Urvashi Grag. 2014. ECLAT Algorithm for Frequent Itemsets Generation. *International Journal of Computer Systems* 1
5. Han J, Kamber M (2006) Data mining concepts and techniques. Morgan Kaufmann
6. Srikant R, Agrawal R (1996) Mining quantitative association rules in large relational tables. *In Acm Sigmod Record* 25:1–12
7. Dehuri S, Jagadev A, Ghosh A, Mall R (2006) Multi-objective genetic algorithm for association rule mining using a homogeneous dedicated cluster of workstations. Am J Appl Sci 3:2086–2095
8. Alatas B, Akin E, Karci A (2008) MODENAR: multi-objective differential evolution algorithm for mining numeric association rules. Appl Soft Comput 8:646–656
9. Mata J, Alvarez JL, Riquelme JC (2002) Discovering numeric association rules via evolutionary algorithm. In: *In Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, Berlin Heidelberg, pp 40–51
10. T. Bharathi and P. Krishnakumari. 2016. A comparative analysis on efficiency of contemporary association rule mining algorithm. *In 2016 3rd International Conference on Advanced Computing and Communication System*
11. Yu H, Wen J, Wang H, Jun L (2011) An improved Apriori algorithm based on the Boolean matrix and Hadoop. Procedia Engineering 15(1):1827–1831
12. Xiangyang S and Ling Z. Apriori Parallel Improved Algorithm Based on MapReduce Distributed Architecture. 2016. IEEE sixth international conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), 517–521
13. Joy R and Sherly K K. Parallel frequent itemset mining with spark RDD framework for disease prediction. 2016. *IEEE International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, 1–5
14. Li H, Wang Y and Zhang D, et al. Pfp: parallel fp-growth for query recommendation. 2008. *ACM Proceedings of the 2008 ACM conference on Recommender systems*, 107–114
15. Rathee S and Kashyap A. Exploiting Apache Flink's iteration capabilities for distributed Apriori: Community detection problem as an example. 2016. *IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 739–745
16. Beiranvand V, Mobasher-Kashani M, Bakar AA (2014) Multi-objective PSO algorithm for mining numerical association rules without a priori discretization. Expert Syst Appl 41:4259–4273
17. Álvarez VP, Vázquez JM (2012) An evolutionary algorithm to discover quantitative association rules from huge databases without the need for an apriori discretization. Expert Syst Appl 39:585–593
18. Qodmanan HR, Nasiri M, Minaei-Bidgoli B (2011) Multi objective association rule mining with genetic algorithm without specifying minimum support and minimum confidence. Expert Syst Appl 38: 288–298
19. Dhanalaxmi B, Naidu GA, Anuradha K (2015) Adaptive PSO based association rule mining technique for software defect classification using ANN. Procedia Comput Sci 46:432–442
20. Agrawal M, Mishra M, and Kushwah S P S. 2015. Association rules optimization using particle swarm optimization algorithm with mutation. *International Journal of Soft Computing and Engineering (IJSCE)* 5
21. Kuo RJ, Chao CM, Chiu YT (2011) Application of particle swarm optimization to association rule mining. Appl Soft Comput 11:326–336
22. Agrawal M, Mishra M, and Kushwah S P S. 2015. Association rules optimization using improved PSO algorithm. *In International Conference on Communication Networks (ICCN)*. 395–398
23. Feng, H., Liao, R., Liu, F., Wang, Y., Yu, Z., Zhu, X. (2018, February). Optimization algorithm improvement of association rule mining based on particle swarm optimization. *In Measuring Technology and Mechatronics Automation (ICMTMA), 2018 10th International Conference on (pp. 524-529).* IEEE
24. Jiang H, Kwong CK, Park WY, Yu KM (2018) A multi-objective PSO approach of mining association rules for affective design based on online customer reviews. J Eng Des:1–23

25. Qianxiang, S., Ping, W. (2017, September). Association rules mining based on improved PSO algorithm. *In Computational Intelligence and Applications (ICCIA), 2017 2nd IEEE International Conference on* (pp. 145-149). IEEE
26. Gupta A, Mehrotra KG, Mohan C (2010) A clustering-based discretization for supervised learning. Statistics & Probability Letters 80(9):816–824
27. J Kennedy and R Eberhart. 1995. Particle swarm optimizaiton. *In IEEE International Conference on Neural Networks*, Vol 4. 1942–1948
28. Horn J, Nafpliotis N, Goldberg D E. A niched Pareto genetic algorithm for multiobjective optimization. *Evolutionary Computation, 1994. Proceedings of IEEE World Congress on Computational Intelligence.* 2002:82–87 vol.1
29. Luna JM, Romero JR, Ventura S (2013) Grammar-based multi-objective algorithms for mining association rules. Data Knowl Eng 86(4):19–37
30. Patel Tushar S, Mayur P, Dhara L, Jahnvi K, Piyusha D, Ashish P, Dhara L (2013) An analytical study of various frequent itemset mining algorithms. Res J Computer & IT Sci 1(1):6–9
31. Mittal A, Nagar A, Gupta K, Nahar R (2015) Comparative study of various frequent pattern mining algorithms. International Journal of Advanced Research in Computer and Communication Engineering 4(4):550–553
32. Nithya P, Lakshmipriya G (2015) An overview of data mining and data warehousing-architecture techniques and applications. International journal of computer science tends and technology (ijcst) 3(1)
33. Ghosh A, Nath B (2004) Multi-objective rule mining using genetic algorithms. Inf Sci 163:123–133
34. Liu B, Hsu W, Chen S, Ma Y (2000) Analyzing the subjective interestingness of association rules. IEEE Intelligent Systems and their Applications 15:47–55

Danfeng Yan professor in State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications (BUPT). She obtained a doctor degree of computer science in BUPT. Her current research interests include Data Mining, Big Data and Analytics.