Neural Comput & Applic (2013) 22 (Suppl 1):S351–S364
DOI 10.1007/s00521-013-1343-9

ORIGINAL ARTICLE

Privacy preserving association rule mining over distributed databases using genetic algorithm

Bettahally N. Keshavamurthy · Asad M. Khan · Durga Toshniwal

Received: 16 July 2011 / Accepted: 25 January 2013 / Published online: 23 February 2013

Abstract Privacy preservation in distributed database is an active area of research. With the advancement of technology, massive amounts of data are continuously being collected and stored in distributed database applications. Indeed, temporal associations and correlations among items in large transactional datasets of distributed database can help in many business decision-making processes. One among them is mining frequent itemset and computing their association rules, which is a nontrivial issue. In a typical situation, multiple parties may wish to collaborate for extracting interesting global information such as frequent association, without revealing their respective data to each other. This may be particularly useful in applications such as retail market basket analysis, medical research, academic, etc. In the proposed work, we aim to find frequent items and to develop a global association rules model based on the genetic algorithm (GA). The GA is used due to its inherent features like robustness with respect to local maxima/minima and domain-independent nature for large space search technique to find exact or approximate solutions for optimization and search problems. For privacy preservation of the data, the concept of trusted third party with two offsets has been used. The data are first anonymized at local party end, and then, the aggregation and global association is done by the trusted third party. The proposed algorithms address various types of partitions such as horizontal, vertical, and arbitrary.

Keywords Privacy preservation · Distributed databases · Genetic algorithm · Association rules mining · Trusted third-party computation

B. N. Keshavamurthy (✉) · A. M. Khan · D. Toshniwal
Department of Electronics and Computer Engineering,
Indian Institute of Technology, Roorkee, Uttarakhand, India
e-mail: bnkeshav123@gmail.com; kesavdec@iitr.ernet.in

A. M. Khan
e-mail: asadmuec@iitr.ernet.in

D. Toshniwal
e-mail: durgafec@iitr.ernet.in

1 Introduction

With the advancement of technology, voluminous data may be easily collected in distributed database applications such as market basket analysis, medical research, etc. In distributed databases, data may be fragmented horizontally, vertically, or arbitrarily. In case of horizontal partition, each site has complete information on a distinct set of entities. An integrated dataset consists of the union of these datasets. In case of vertical partition, each site has partial information on the same set of entities. An integrated dataset would be produced by joining the data from all the sites. Arbitrary fragmentation is a hybrid of the previous two [23].

It is tedious to analyze such voluminous databases manually; thus, data mining plays a significant role. The key algorithms of data mining include classification, clustering, and association rule mining. The association rule mining, used for pattern discovery, has received a greater attention with Rakesh Agarwal's proposal of Apriori algorithm [1]. The association rule mining extracts interesting correlations, frequent patterns, and associations among sets of items in the transaction databases. Association means set of rules which describes how the presence of certain set of items/itemsets influences the presence of other items/itemsets. The key factors in deciding the interestingness of generated association rules are support and confidence.

In recent years, a lot of work has been done on association rule mining over distributed databases. Adding privacy to association rule discovery is an active area of research nowadays. Privacy preserving data mining allows computation of aggregate statistics over an entire dataset without compromising the privacy of private data of the participating data sources.

The important proposals on association rule mining, association rule mining for horizontal partition, and association rule mining for vertical partition are briefed over here. The key literature on association rule mining are [2–4]. In [2], a model of hierarchical management on the cryptogram is put forward by combining the advantages of both RSA public key cryptosystem and homomorphic encryption scheme. A generalized data mining model for association rule mining for distributed database is provided in [3]. Distributed association mining based on an improved Apriori algorithm has been proposed in [4]. It first generates all the local frequent itemsets at each node and then communicates to the general node, producing the global frequent itemset of association rules.

Association rule mining over horizontally partitioned distributed database is addressed in [5]. In [6], an improved algorithm based on the Fast Distributed association rule Mining (FDM) algorithm is proposed. It computes the total support count with the privacy-preserved method and ensures the source of every local large itemset and local support count is covered, so that it reduces the time spent on communication and preserves the privacy of the data distributed at each site. Later, enhanced Hussein et al.'s Scheme (EMHS) for privacy preserving association rules mining on horizontally distributed databases has been proposed in [7]. EMHS is based on the Hussein et al.'s Scheme (MHS) and improves privacy and performance even after increasing the number of sites.

Association rule mining for vertical partitioned distributed database is well addressed in [8]. By applying privacy preservation among multiple parties, a collusion-resistant algorithm for distributed association rule mining based on the Shamir's secret sharing technique has been proposed [9]. Later, vector-based semi-honest third-party product protocol (T-VDC) has been proposed for vertically partitioned distributed database, which overcomes secure two-party vector dot product computation algorithm drawbacks, such as the large communication cost and easy to leak privacy [10]. Secure two-party computation was originally proposed by Yao [11]. An approach for privacy preservation in mining the frequent items over distributed scenario is proposed in [12].

Evolutionary techniques are well known for multi-object research problems. Genetic algorithm is one of the key techniques of evolutionary algorithm [13–15]. The GA is used due to its inherent features like robustness with respect to local maxima/minima and domain-independent nature for large search technique, to find exact or approximate solutions for optimization and search problems. These features make it well suited for solving the problems about which we have less knowledge.

Genetic algorithms for rule mining are broadly classified into Pittsburg approach and Michigan approach, based on the encoding of rules in the population of chromosomes. Michigan approach is more popular [16, 17]. The GA is a multi-objective problem rather than a single object greedy technique [18]. The key proposals on classification and association rule mining using GA are [19–21]. Genetic algorithm-based approach for mining classification rules from large database is provided in [19]. The review of classification using genetic programming is described in [20]. In [21], it gives solutions for new problems which are based on an adaptive variant of genetic programming.

The rest of the paper is organized as follows: Sect. 2 gives the basic definitions, problem definition, and proposed approach. Section 3 includes the discussion and results, and Sect. 4 concludes with references.

2 Preliminaries

The basic definitions of frequent pattern, association rule mining, and secure multi-partition computation will be presented in Sect. 2.1. Section 2.2 covers the problem definition, and the proposed approach will be described in Sect. 2.3.

2.1 Basic definitions

Definition 1 (*Frequent itemset*) Let $I = \{i_1, i_2 \ldots i_n\}$ be a set of n binary attributes called items. Let D, the task-relevant data, be a set of transactions. Let $T = \{t_1, t_2 \ldots t_m\}$ be a set of transactions, where each transaction in T is a set of items such that $t_i \subset I$. Each transaction in D is associated with identifier, called *TID*. The occurrence frequency of an item is the number of transactions that contains that item. An item in I is frequent, if it satisfies the minimum support threshold [12].

Definition 2 (*Association rule*) Let A be a set of items. A transaction t_i is said to contain A if and only if $A \subseteq t_i$. An association rule is an implication rule of the form $A \Rightarrow B$, where $A \subset I$, $B \subset I$, and $A \cap B$. The key factors deciding the interestingness of association rules are support and confidence [5].

$$\text{Support } (A \rightarrow B) = P(A \cup B) \tag{1}$$

$$\begin{aligned}\text{Confidence}(A \rightarrow B) = P(B/A) &= \frac{\text{Support}(A \cup B)}{\text{Support}(A)} \\ &= \frac{\text{Support\_Count}(A \cup B)}{\text{Support\_Count}(A)}\end{aligned} \tag{2}$$

Definition 3 (*Fitness function*) Fitness function for rule discovery is the key to the convergence of *GA*. The basic parameters to measure the fitness function are support and confidence.

Definition 4 (*Secure multiparty computation* (*SMC*)) In distributed database environment, to compute the global patterns from n data sources, secure multiparty computation (SMC) is used. When there are n data sources $DS_0, DS_1, \ldots, DS_{n-1}$, such that each DS_i has a private data item d_i, where $i = 0,\ 1\ \ldots\ n-1$, the parties want to compute $\sum_{i=0}^{n-1} d_i$ privately, without revealing their private data d_i to each other. For solving the above problem, SMC was proposed in [11]:

Assumptions $\sum_{i=0}^{n-1} d_i$ is in the range $[0, m-1]$, and DS_j is the protocol initiator.

1. At the beginning, DS_j chooses a uniform random number r within $[0, m-1]$.
2. Then, DS_j sends the sum $d_j + r\ (mod\ m)$ to the data source $DS_{j+1}\ (mod\ n)$.
3. Each remaining data sources, DS_i, do the following: Upon receiving a value x, data source DS_i sends the sum $d_i + x\ (mod\ m)$ to the data source $DS_{i+1}\ (mod\ n)$.
4. Finally, when party DS_j receives a value from the data source $DS_{j+n-1}\ (mod\ n)$, it will be equal to the total sum $r + \sum_{i=0}^{n-1} d_i$. Since r is only known to DS_j, it can find the sum $\sum_{i=0}^{n-1} d_i$ and distribute this global sum to collaborating data sources.

2.2 Problem description

Let DB be a distributed database, partitioned either horizontally, vertically, or arbitrarily across n sites, S_1, S_2,…., S_n. Let DB_1, DB_2…DB_n be the distributed databases at these sites. In typical scenarios of distributed databases, like retail market basket analysis, the distributed sites will be willing to share their private information in a privacy preserving manner for computing global association rules, viewing the benefits such as customer retention, opening new franchise, etc. The problem of global association rule computation for distributed sites consists of two tasks, frequent itemset mining from aggregated database followed by association rule computation, for those frequent patterns which holds the user-defined minimum threshold.

2.3 Proposed approach

The block diagram for the proposed model of privacy preserving distributed mining using genetic algorithm is shown in Fig. 1. It consists of the following steps:

(a) Local GA-based frequent pattern mining
(b) Privacy preservation of local data at each collaborating parties side and global data at trusted third-party side
(c) Formulation of association rules
(d) Dissemination of global association rules to participating parties.

2.3.1 Local GA-based frequent pattern mining

It is the first stage of proposed model which computes the frequent items locally at participating sites using GA. The block diagram for the frequent pattern mining at the local parties is shown in Fig. 2. GA consists of the following stages for computing the frequent patterns:

i. Representation of dataset
ii. Chromosome representation
iii. Crossover
iv. Mutation
v. Selection
vi. Fitness functions for frequent pattern mining

i. *Representation of dataset* Each transaction is represented as a constant length transaction with the presence and absence of an item correspondingly represented by ‘1’ and ‘0’. Each transaction in Table 1 has the same length as of chromosome, that is, 25, but has different number of items. This is necessary, because in GA the chromosomes must be of same length. So, indeed our transactions are of variable length, that is, each has different number of items.

For example, let us consider transactions t_1 to t_5, whose itemsets are {‘ADJLNQTW’, ‘DPQY’, ‘AINQTU’, ‘DLPQ’, ‘DELNPQR’}. The representation is shown in Table 1.

ii. *Chromosome representation* The bit ‘1’ and ‘0’ correspondingly represents the presence and absence of item in the chromosomes. Table 2 represents the corresponding entries for transactions $t1$ to $t5$.

iii. *Crossover* Crossover selects genes from parent chromosomes and creates a new offspring. The Uniform Multi Point crossover is performed by randomly choosing some crossover points. An offspring’s chromosome is made by copying the first parent’s content before the crossover point and second parent’s content after the crossover point. Crossover point is denoted by the symbol ‘|’, and it is well depicted in Table 3. Crossover rate is the percentage of chromosomes used for generating offspring from the given population. Crossover rate for our proposed approach is 0.75.

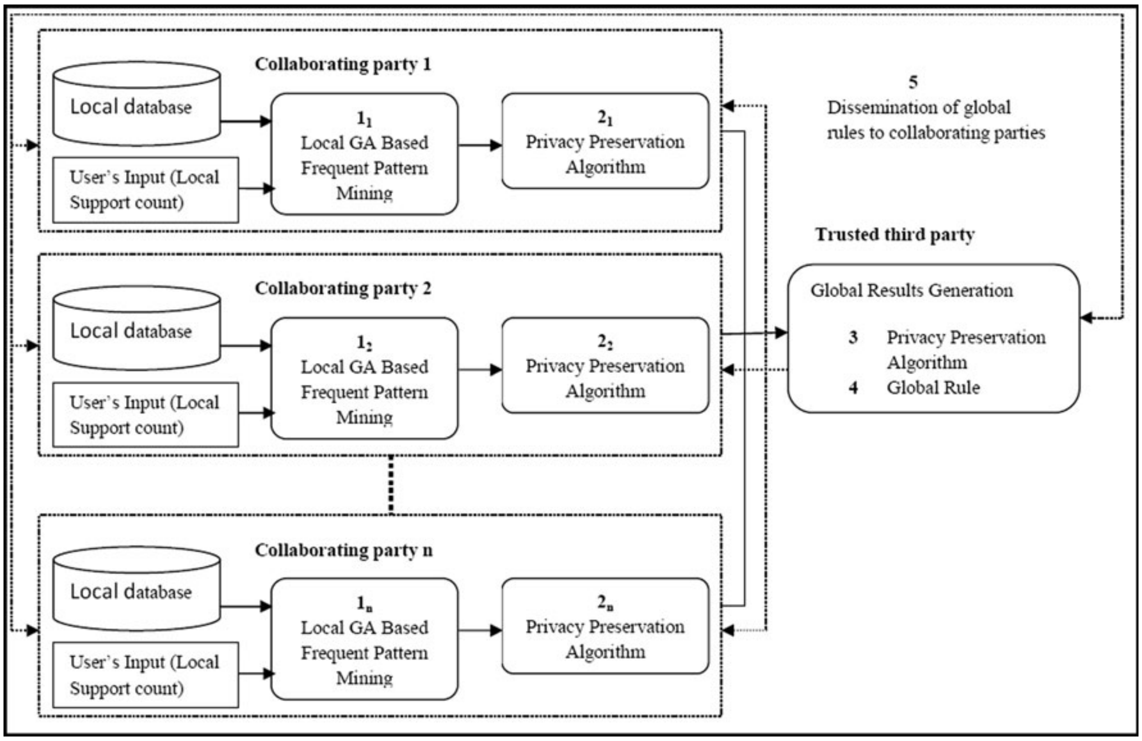
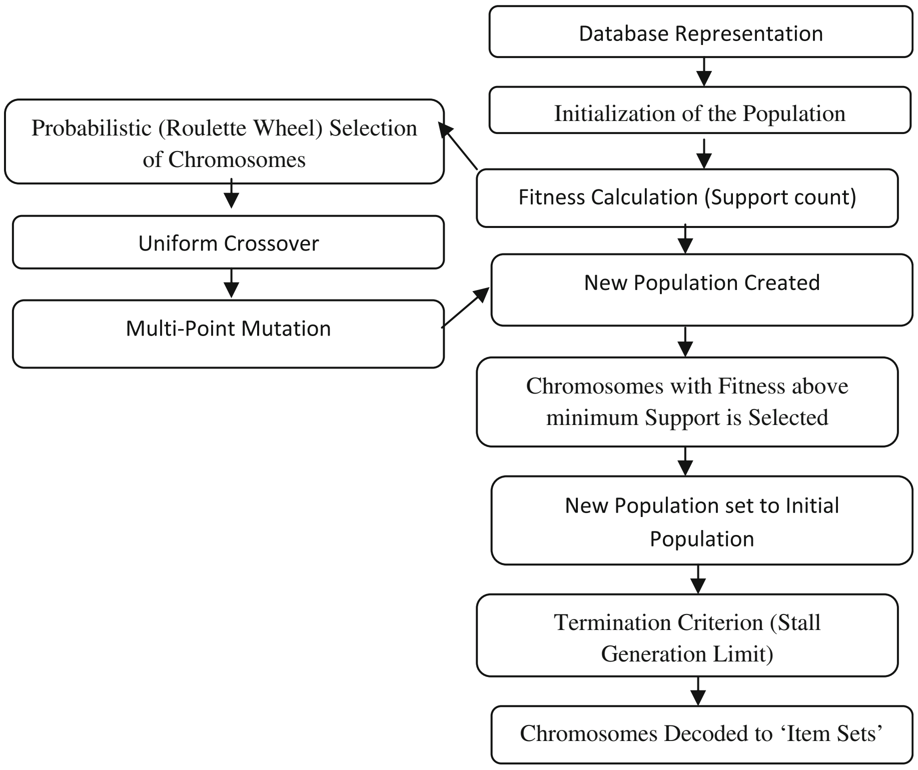

Collaborating party 1
Local database
User's Input (Local Support count)
1₁ Local GA Based Frequent Pattern Mining
2₁ Privacy Preservation Algorithm
Collaborating party 2
Local database
User's Input (Local Support count)
1₂ Local GA Based Frequent Pattern Mining
2₂ Privacy Preservation Algorithm
Collaborating party n
Local database
User's Input (Local Support count)
1ₙ Local GA Based Frequent Pattern Mining
2ₙ Privacy Preservation Algorithm
5 Dissemination of global rules to collaborating parties
Trusted third party
Global Results Generation
3 Privacy Preservation Algorithm
4 Global Rule

Fig. 1 Proposed scheme for privacy preservation in distributed databases using genetic algorithm

Fig. 2 Frequent pattern mining using genetic algorithm


Database Representation
Initialization of the Population
Fitness Calculation (Support count)
Probabilistic (Roulette Wheel) Selection of Chromosomes
Uniform Crossover
Multi-Point Mutation
New Population Created
Chromosomes with Fitness above minimum Support is Selected
New Population set to Initial Population
Termination Criterion (Stall Generation Limit)
Chromosomes Decoded to 'Item Sets'

iv. *Mutation* The mutation process helps in preventing the solutions from falling to the same category through local optima. Mutation modifies randomly selected offsprings generated from crossover process. In the mutation process, the mutation bits are randomly chosen and are inverted in place. For

Table 1 Data representation

Transaction	Item	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
T1	ADJLNQT	1	0	0	1	0	0	0	0	0	1	0	1	0	1	0	0	1	0	0	1	0	0	0	0	0
T2	DPQY	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1
T3	AINQTU	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	1	1	0	0	0	0
T4	DLPQ	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0
T5	DELNPQ	0	0	0	1	1	0	0	0	0	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0

Table 2 Chromosome representation

Itemset	Chromosome
ADJLNQT	1001000001010100100100000
DPQY	0001000000000001100000001
AINQTU	1000000010000100100110000
DLPQ	0001000000010001100000000
DELNPQ	0001100000010101100000000

Table 3 Crossover computation

Chromosome 1	110110000 ǀ 0010011011000000	ABDE ǀ LOPRS
Chromosome 2	110010000 ǀ 1100001111000000	ABE ǀ JKPQRS
Offspring 1	110110000 ǀ 1100001111000000	ABDE ǀ JKPQRS
Offspring 2	110010000 ǀ 0010011011000000	ABE ǀ LOPRS

chromosomes with fitness value below threshold, few randomly selected bits with the value ‘1’ are inverted. Similarly for chromosomes with fitness value greater than threshold, a few randomly selected bits with value ‘0’ are inverted. Hence, this process results in increase or decrease in the itemset size of the population.

The proposed approach uses ‘multipoint nonuniform mutation’. Mutation rate for the proposed approach is as follows:

(a) For flipping bit value from 0 to 1 is 0.025.
(b) For flipping bit value from 1 to 0 is 0.095.

The difference in the case (a) and (b) of mutation is because initialization of the population was done with a probability of 0.25 for bit 1 and 0.75 for bit 0. With further progress of GA, the average number of bits with value 1 will decrease and with value 0 will increase.

For example, if the chromosome 1100000010000010100000100 has fitness value greater than the threshold, then the randomly selected bit(s) with value ‘0’ is inverted to ‘1’, thus modifying the above chromosome to 1100000010010010100000100. Similarly, if the chromosome 1001000001000100000000000 has fitness value less than the threshold, then the randomly selected bit(s) with value ‘1’ is inverted to ‘0’, so the above chromosome becomes 1000000001000100000000000.

v. Selection Roulette Wheel selection is performed. In the Roulette Wheel selection, individuals are given a probability of being selected that is directly proportionate to their fitness. Two individuals are then chosen randomly based on these probabilities and produce offspring.

```
Procedure:
For all members of population
   sum += fitness of this individual
For all members of population
   probability = sum of probabilities + (fitness / sum)
   sum of probabilities += probability
Loop Until new population is full                    //Loop executes twice
Do
   number = Random between 0 and 1
   For all members of population
          If (number > probability but less than next probability)
               Then you have been selected
    create offspring
```

vi. *Fitness functions for frequent pattern mining* Fitness function for rule discovery is the key to the convergence of the GA. The basic parameters to measure the fitness function are support and confidence.

At local party The fitness function used is the support factor of the itemset represented by the chromosomes. The frequent pattern discovered should have a higher value of support compared to the threshold (local support count).

2.3.2 *Privacy preservation of local data at collaborating parties side and global data at trusted third-party side (TTP)*

i. *Horizontal partition* In horizontal partition, each party receives offset from the TTP to preserve the privacy of personal information. Each collaborating party will then perturb their data using this offset (offset + personal data) and then send the results to the logical neighbor specified by TTP. Each collaborator will not know whether his logical neighbor is a TTP or other collaborator. This will continue till the last party, where the last party will send the result to TTP without knowing his logical neighbor is the TTP. Once TTP receives the data from the last party, it decrypts the actual frequent patterns by subtracting the offset. The following secure sum computation with two offset values and trusted third-party procedure is used to preserve the privacy.

Procedure: Secure sum computation with two offset values and trusted third party

Assumption Let us assume that there are n distributed data sources $DS_0, DS_1, \ldots, DS_{n-1}$ such that each DS_i has a private data item d_i, $i = 0, 1, \ldots n-1$, and the parties want to compute $\sum_{i=0}^{n-1} d_i$ privately, without revealing their private data d_i to each other. It is also assumed that $\sum_{i=0}^{n-1} d_i$ is in the range $[0, m-1]$ and *TTP* is the protocol initiator.

At Trusted Third Party (TTP)

1. The collaborators willing to get global result are connecting to the TTP.
2. TTP generates a uniform random number r to select one of the n parties as an initiator, that is, DS_i.
3. TTP generates offset values and sends these offset values to each collaborator to perturb their personal patterns:
   a. If it is an initial collaborator, then it receives only *newoffset* to perturb his patterns.
   b. If a collaborator has new patterns in comparison with the existing list of patterns, received from his logical predecessor, then it uses two offsets called as *oldoffset* and *newoffset*, where *oldoffset* is used to perturb existing patterns and *newoffset* is used to perturb patterns initiated by him.
4. Finally, when TTP receives values from the data source DS_{n-1} (*mod n*), it will be equal to the total sum (*offset values* + $\sum_{i=0}^{n-1} d_i$). Since *offset values* are only known to TTP, it can find the sum $\sum_{i=0}^{n-1} d_i$ and distribute this global sum to collaborating data sources.

At Collaborating Party (CP)

CP receives the offset, adds its value to the random value received from TTP, and sends the resulting values to the logical neighbor specified by TTP.

1. If it is an initial collaborator, then add its own data value D_i to *newoffset* value received from TTP.
2. Based on the kind of patterns at each DS_i, $i = 1, \ldots n-1$, following actions will be done at each collaborator:

(a) If there are no new patterns in comparison with the pattern list obtained from its logical predecessor, then *Do sum* (d_i + *oldoffset*).

(b) If there are new patterns in comparison with the pattern list obtained from its logical predecessor, then *Do sum* (d_i + *newoffset*) for existing patterns and *Do sum* (d_i + *oldoffset*) for newly initiated patterns.

(c) Finally, send the list of patterns to the logical successor specified by TTP.

For example, let us consider three collaborating parties party1, party2, and party3, respectively, hold the dataset {20AB, 50DEF, 5G}, {10AB, 25DEF, 20 K, 33XYZ}, and {10AB, 15DEF}, who are willing to share their data for global pattern mining. Party1 uses *newoffset* value 20 to perturb its patterns. Party 2 uses *oldoffset* value 15 *and newoffset* value 35 to perturb its patterns, since it has both repeated patterns and newly initiated patterns when compared to the list of patterns obtained from its logical predecessor. Party 3 uses *oldoffset* value 17 to perturb its data, since it does not has any newly initiated patterns. Party1, party2, and party3 correspondingly send their resulting patterns {40AB, 70DEF, 25G}, {65AB, 80DEF, 55 K, 40G, 68XYZ}, and {92AB, 112DEF, 72 K, 57G, 85XYZ} to their logical successor specified by TTP. Finally, when TTP receives the resulting patterns from the last collaborator, it subtracts global offset value 52 from the obtained pattern values to obtain actual patterns {40AB, 60DEF, 20 K, 5G, 33 XYZ}. This is well depicted in Fig. 3.

ii. *Vertical partition* In vertical partition, each party receives offset from the TTP to preserve the privacy of personal information. Each party then perturbs its data and sends the result (offset + personal data) to the TTP.

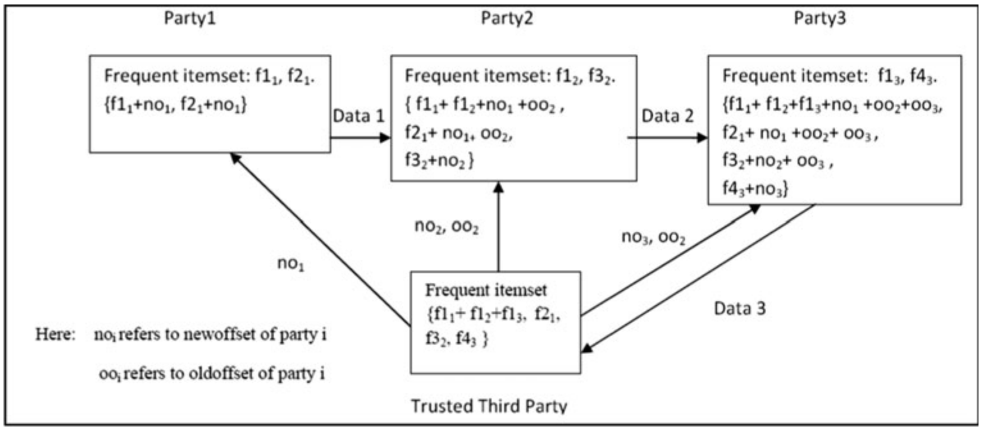

Party1
Party2
Party3
Frequent itemset: f1₁, f2₁. {f1₁+no₁, f2₁+no₁}
Data 1
Frequent itemset: f1₂, f3₂. { f1₁+ f1₂+no₁ +oo₂ , f2₁+ no₁, oo₂, f3₂+no₂}
Data 2
Frequent itemset: f1₃, f4₃. {f1₁+ f1₂+f1₃+no₁ +oo₂+oo₃, f2₁+ no₁ +oo₂+ oo₃ , f3₂+no₂+ oo₃ , f4₃+no₃}
no₂, oo₂
no₃, oo₂
no₁
Data 3
Frequent itemset {f1₁+ f1₂+f1₃, f2₁, f3₂, f4₃ }
Here: noᵢ refers to newoffset of party i
ooᵢ refers to oldoffset of party i
Trusted Third Party

Fig. 3 Privacy preservation for horizontal partition

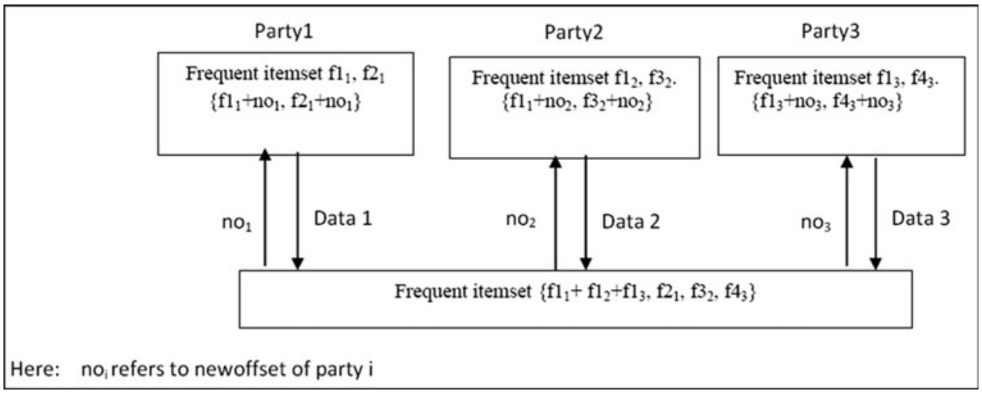

Party1
Party2
Party3
Frequent itemset f1₁, f2₁ {f1₁+no₁, f2₁+no₁}
Frequent itemset f1₂, f3₂. {f1₁+no₂, f3₂+no₂}
Frequent itemset f1₃, f4₃. {f1₃+no₃, f4₃+no₃}
no₁
Data 1
no₂
Data 2
no₃
Data 3
Frequent itemset {f1₁+ f1₂+f1₃, f2₁, f3₂, f4₃}
Here: noᵢ refers to newoffset of party i

Fig. 4 Privacy preservation for vertical partition

For example, let us consider three collaborating parties party1, party2, and party3, respectively, hold the dataset {40AB, 60DEF}, {20 K, 5G}, {33XYZ}, who are willing to share their data for global pattern mining. Party1 uses 25 as offset value, party2 uses 10 as offset value, and party3 uses 15 as offset value, to perturb their patterns. Unlike in horizontal partition, here each collaborator receives only one offset value. Party1, party2, and party3, respectively, send their perturbed patterns {65AB, 85DEF}, {30 K, 15G}, and {48XYZ} to TTP. After receiving patterns from all collaborators, TTP subtracts offset values provided to each CP, from the perturbed patterns to obtain original aggregate patterns. Thus, TTP obtains complete frequent patterns of all the collaborators which are {65AB, 85DEF, 30 K, 15G, 48XYZ}. This is well depicted in Fig. 4.

iii. *Arbitrary partition* It is the combination of horizontal and the vertical partition. Here, the parties are predetermined about their distribution, and if the parties are distributed horizontally, then procedure mentioned in Sect. 2.3.2 (i) is used to aggregate the frequent patterns, else the procedure mentioned in Sect. 2.3.2 (ii) is used to aggregate the frequent patterns at TTP. Then, find the total frequent items by adding the frequent items of horizontal and vertical partitions.

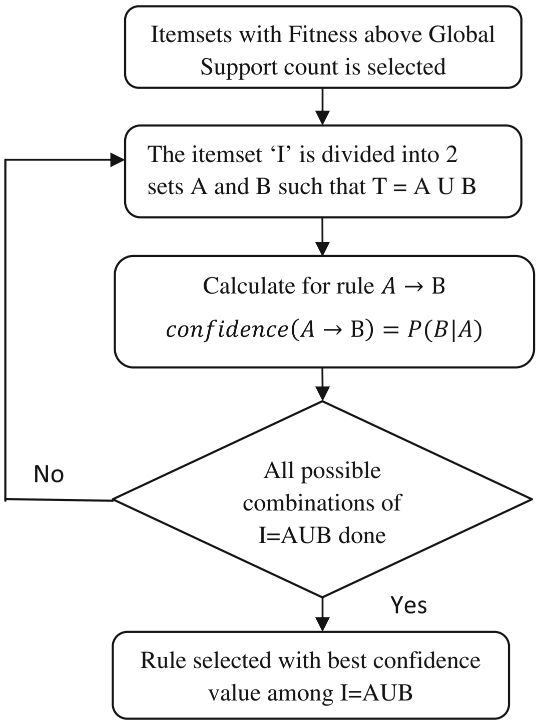

Itemsets with Fitness above Global Support count is selected
The itemset 'I' is divided into 2 sets A and B such that T = A U B
Calculate for rule $A \rightarrow B$
$confidence(A \rightarrow B) = P(B|A)$
All possible combinations of I=AUB done
No
Yes
Rule selected with best confidence value among I=AUB

Fig. 5 Association rule formation for distributed database mining

2.3.3 *Formulation of association rules*

Formulation of association rules is done at TTP side. The rule discovered should have high support as well as high confidence, but there is always a trade-off between the two, as they are inversely related. So, a good rule will be one with high confidence and a significant amount of support value. The condition of significant amount of support is met by keeping two support values, local support count (at CP end) and global support count (at TTP end). The condition of high confidence (at third party) value for a rule like *A*–*B* is met using Eq. (2), where *A* and *B* are set of items.

2.3.4 *Dissemination of global association rules to participating parties*

Finally, the mined association rules from the above step will be sent to the collaborating parties by TTP. The collaborating parties can use these results in making their critical decisions such as customer retention, efficient stock updating, open new franchise, etc.

3 Discussion and results

Section 3.1 gives the details about the procedure used to generate the dataset. Section 3.2 gives the comparison of results of frequent item generation of Apriori versus genetic algorithm, and Sect. 3.3 details the distributed mining results. All experiments have been conducted on Intel Core 2 Duo CPU @ 2.10 GHZ machines supporting RAM 8 GB with 64 bit OS (Fig. 5).

3.1 Dataset generator

We have modified the synthetic data generator of Rakesh Agarwal by adding priority to an item and frequency to an itemset, assuming the number of customers to be five, which can be easily extended for other set of values [22]. Parameters and its values assumed in algorithm are mentioned in Table 4. The algorithm is given as follows:

```
Algorithm GenDB (Parameters shown in Table IV)
For (Number of maximal potentially large itemsets)
     {
     Determine the size of next itemset from Poisson distribution with mean = |I|;
     Items in first itemset are chosen randomly;
     Some fraction of items in subsequent itemset is chosen from previous itemset
     generated according to random exponential distribution with mean equal to
     correlation level r;
     Remaining items are chosen randomly;
     Weight is assigned according to exponential distribution with unit mean;
     The weights are normalized so that sum of weights for all itemsets in T is 1;
     }     // end of for loop
Weights to items are assigned according to exponential distribution with unit
mean;
Normalize the weights of items so that sum of weights for all items is 1;
For (Number of transactions)
    {   //beginning of the outer for loop
        Randomly select which Sequences have transaction at this timestamp;
        For (Sequences selected in previous step)
                {        //beginning of inner for loop
                Determine the size of next transaction from Poisson distribution
                with mean= |T|;
                Do {
                    The next itemset to be put in the transaction is chosen from T
                    by tossing a |L|-sided weighted coin;
                    Randomly assign support for each item added;
                    If (the large itemset on hand does not fit in the transaction)
                       {
                       Itemset is put in the transaction anyway in half the cases,
                       (and the itemset is moved to the next transaction the rest of
                       the cases);
                       }        //end of if condition
                    } While (Transaction is incomplete)
                  } //end of inner for loop
          }  //end of outer for loop
    End
```

Table 4 Parameters

Abbreviation	Description	Values
$\|D\|$	Number of transactions	100
$\|T\|$	Average size of the transactions	6
$\|I\|$	Average size of the maximal potentially large itemsets	4
$\|L\|$	Number of maximal potentially large itemsets	25
N	Number of items	20
n	Number of sequences (Number of Clients)	20
r	Correlation level (0, 1)	0.5

3.2 Frequent items generation using genetic algorithm versus Apriori

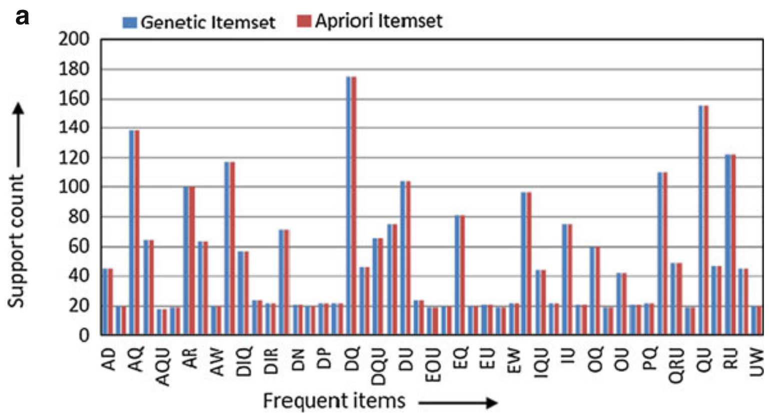
In our study, we have conducted a test for frequent items generation. We compared the genetic algorithm with Apriori algorithm using different support counts. The number of items generated by genetic algorithm is quite same as the number of frequent items generated by Apriori. Figure 6a, b gives the comparison of Apriori algorithm versus genetic algorithm. From the results, we can conclude that the genetic algorithm is an appropriate choice for frequent pattern analysis of market basket data.

3.3 Distributed scenarios

Here, results obtained for the different distributed fragmentations such as horizontal, vertical, and hybrid described with necessary graphs.

3.3.1 Horizontal partition

Here, we have considered five collaborating parties for dataset mentioned in Sect. 3.1. This can be easily extended for more number of collaborating parties. Frequent items are generated locally at collaborator side and globally at TTP. The number of frequent items generated at different collaborators and TTP, for different threshold values of support, is shown in Fig. 7a–d. The graphs are plotted for different frequent items against its support count. We varied the support threshold from 0.25 to 0.4, with incremental value of 0.05, and studied its effect on frequent item generation. From the graph, it is clear that as the threshold value for support increases, the number of frequent items decreases.



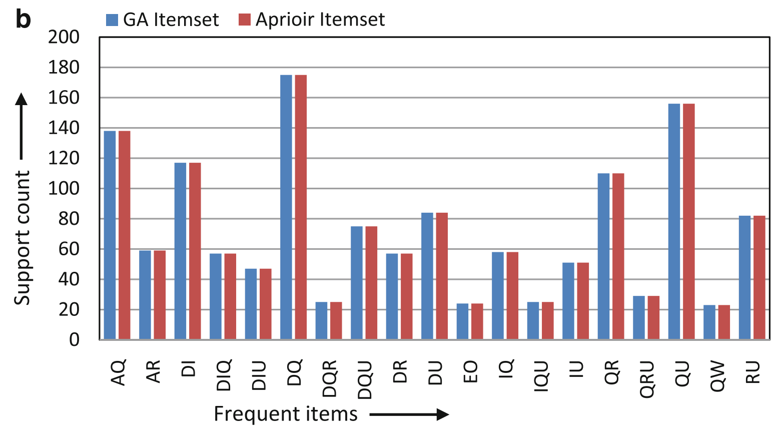


Fig. 6 Apriori versus genetic algorithm for frequent pattern generation **a** at support = 0.3, **b** at support = 0.35

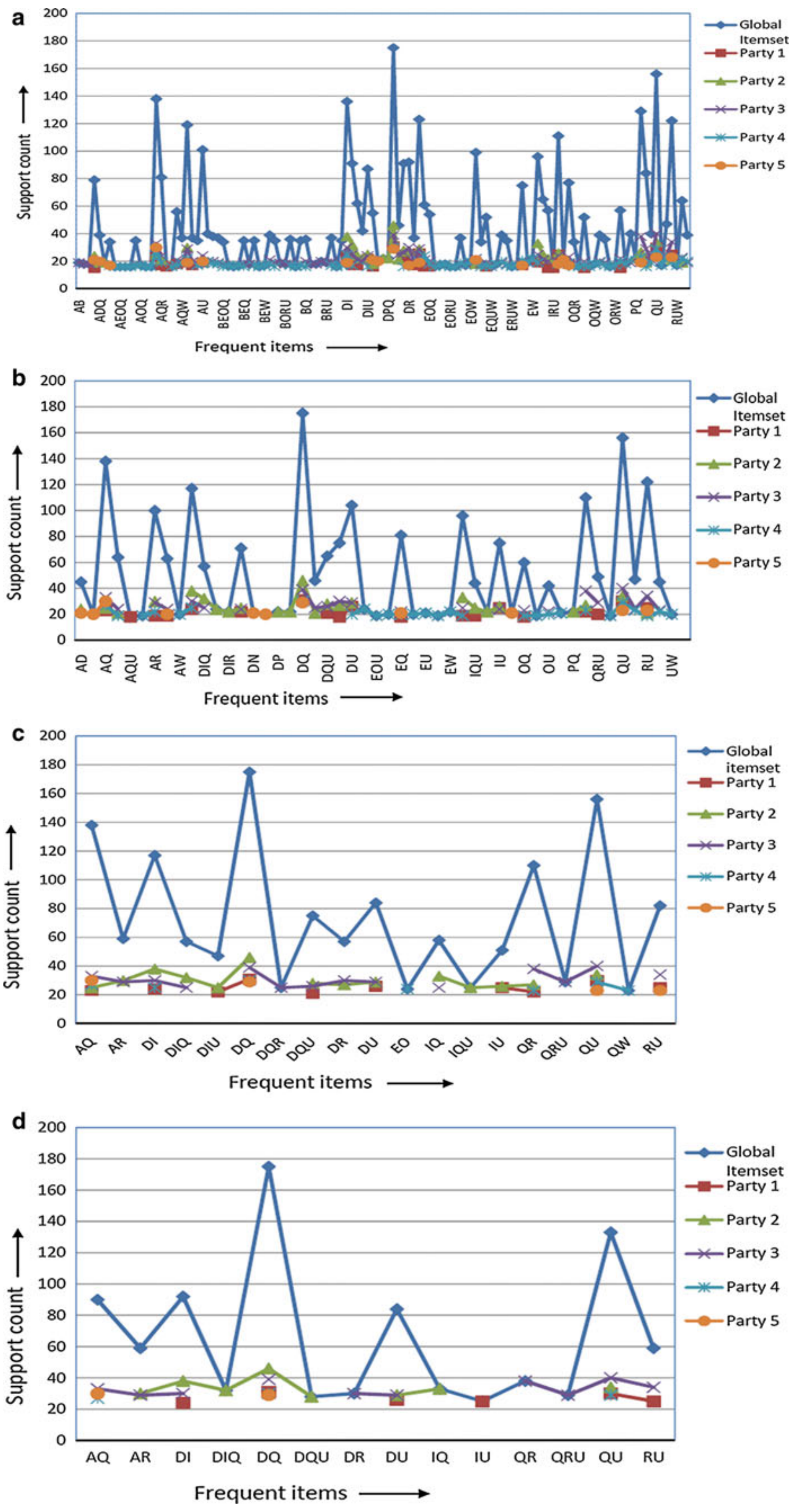


Fig. 7 Frequent items generated for horizontally partitioned collaborators **a** when support = 0.25, **b** when support = 0.3, **c** when support = 0.35, and **d** when support = 0.4

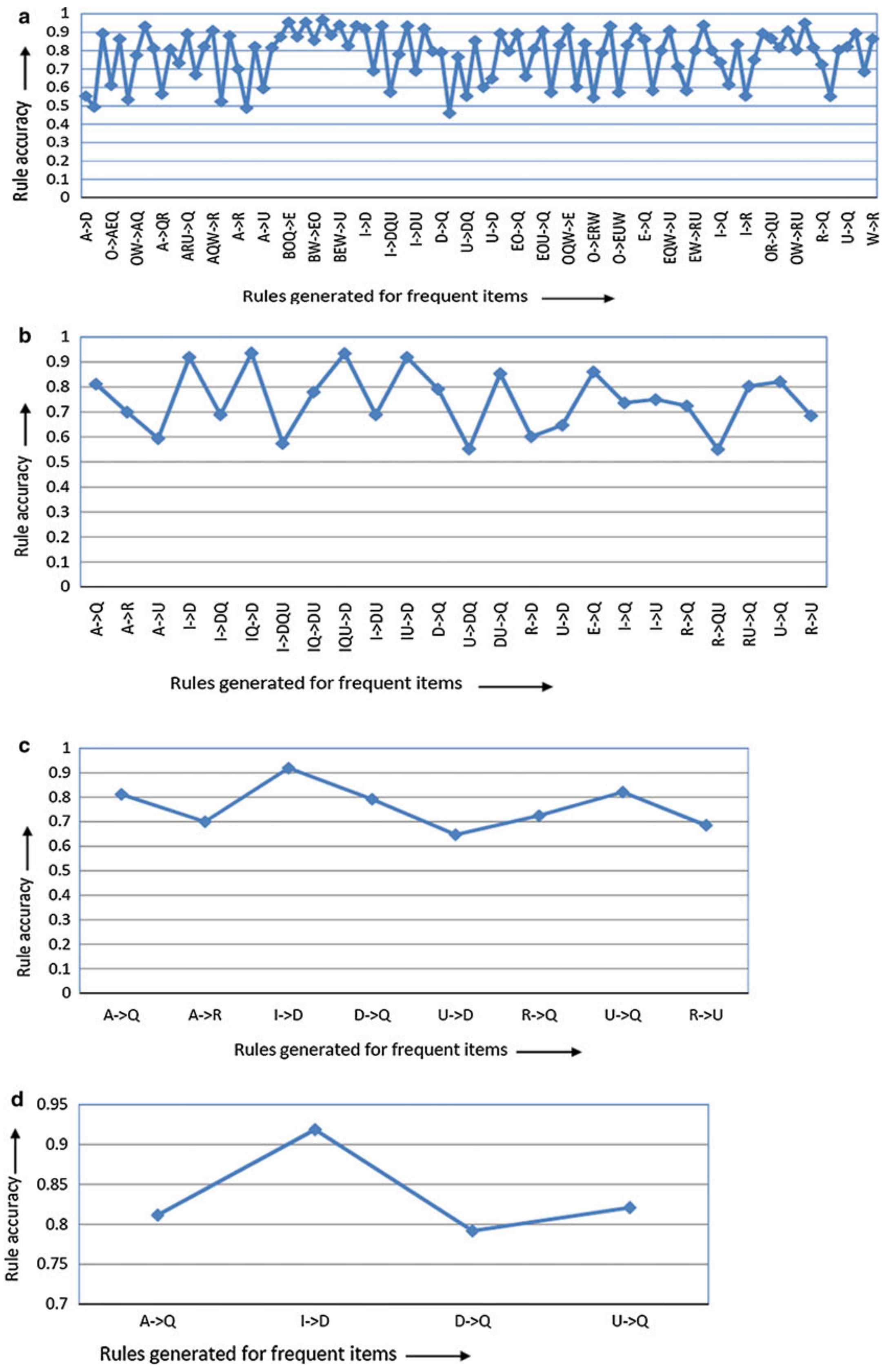


Fig. 8 Association rule generation for global frequent items for horizontal partition **a** when support = 0.25, **b** when support = 0.3, **c** when support = 0.35, and when support = 0.4

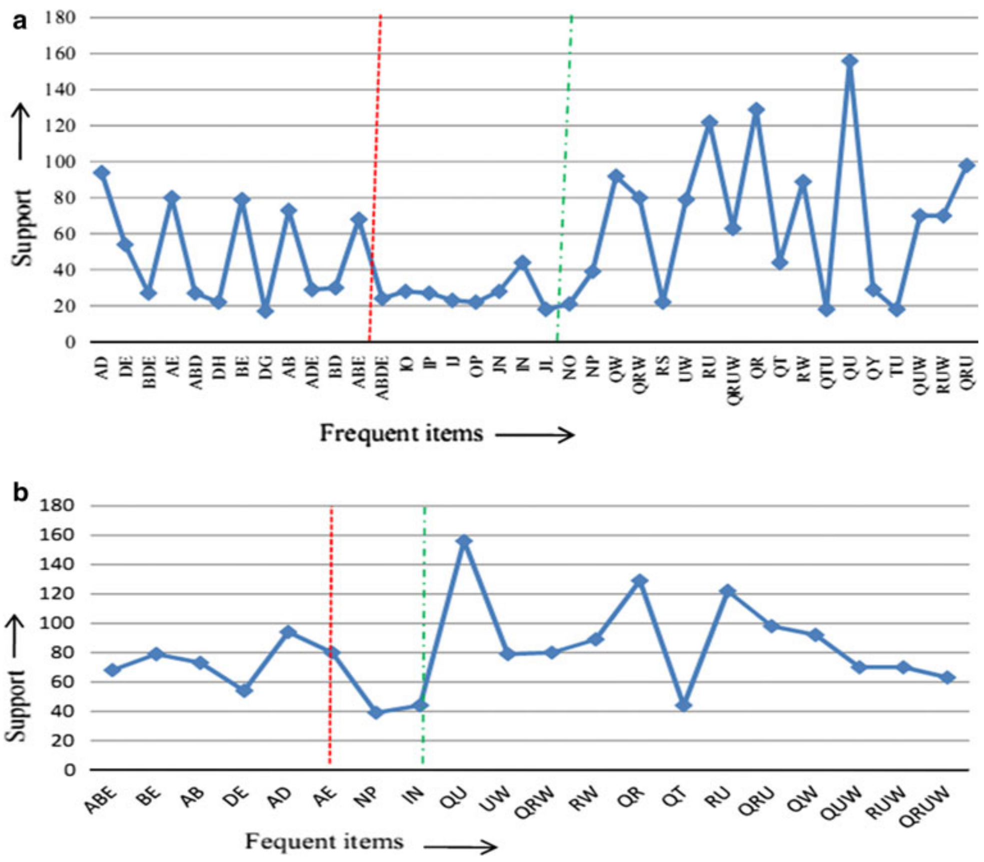


Fig. 9 Frequent items generated for vertically partitioned collaborators **a** when support = 0.05, **b** when support = 0.1

The association rules are generated from aggregate frequent items at TTP, for same threshold values of support as mentioned above. The graph is plotted based on the rules generated against its rule accuracy. It is clear from the graph (Fig. 8a–d) that as the threshold increases, the number of rules generated will be less.

3.3.2 Vertical partition

In vertical partition, we have considered three collaborating parties for our dataset mentioned in Sect. 3.1, which can also be easily extended for higher number of collaborators. Figure 9a, b shows the items generated at different collaborators against their support, for the threshold values 0.05 and 0.1. Items to the left of red line (dotted line) were generated by the first party, items in between the red (dotted) and green (dash and dotted line) were from the second party, and items to the right of green line (dash and dotted line) were from third party. From the graphs, it is clear that as threshold value of support increases, the number of frequent items decreases.

The association rules are generated from aggregate frequent items at TTP, for same threshold values of support as mentioned above. The graph is plotted based on the rules generated against its rule accuracy. It is clear from the graph (Fig. 10a, b) that as the threshold increases, the number of rules generated will be less.

3.3.3 Hybrid partition

Here, collaborators distributed either horizontally or vertically. Frequent items of collaborators are aggregated at trusted party end and compute the association rules based on confidence. It is quite similar to the vertical partition.

4 Conclusions

In comparison with traditional frequent pattern mining algorithm, the proposed approach of GA has two potential advantages. In frequent pattern mining, the population is formed only once, whereas in GA method, the population is formed for each generation maximizing the sample set. Further, in the frequent pattern mining technique for kth iteration, previous $k-1$ itemsets need to be stored which is not required in GA. However, one major drawback of GA approach is that GA could result in duplicate formation in its successive generations.

The key goal for privacy preservation data mining over distributed database is to allow computation of aggregate statistics over an entire dataset without compromising the

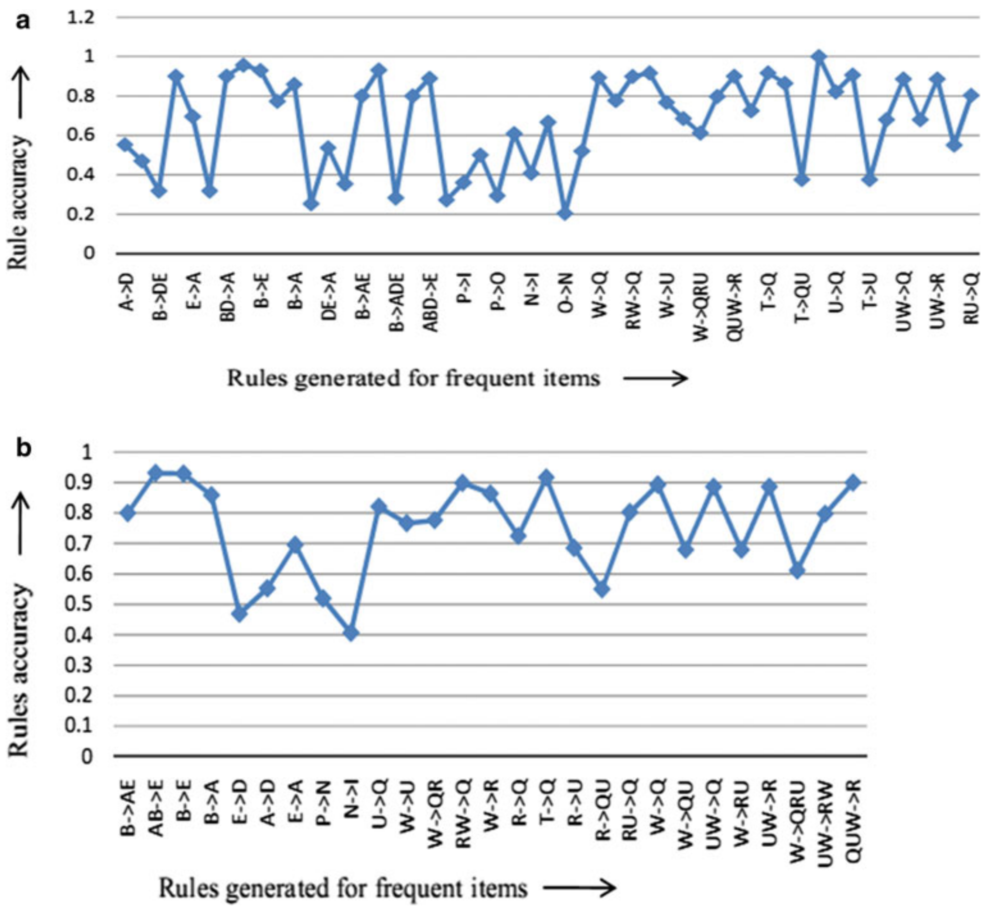


Fig. 10 Association rule generation for global frequent items for vertical partition **a** when support = 0.05, **b** when support = 0.1

privacy of private data of the participating data sources. So, algorithms for privacy preservation can further be improved based upon the trade-offs between reconstruction accuracy and privacy. The fitness function of GA plays an important role, and the convergence of search space is directly proposal to the effectiveness of fitness function; in other words, if fitness function is good, then better the convergence of GA for a given problem. Also, genetic operator refinement for a problem plays a vital role for the convergence of search space to an optimal solution. The proposed approach can further be improved by improving the chromosome representation and fitness function parameters.

Acknowledgments This research work was partially funded by IBM Centre for Advanced Studies Projects (CAS), currently going on at IIT Roorkee.

References

1. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the international conference on very large data bases, Santiago, pp 487–499
2. Qiong G, Xiaohui C (2009) A privacy-preserving distributed method for mining association rules. Artificial intelligence and computational intelligence, AICI '09, Shanghai, pp 294–297
3. Cao W, Hu X, Dong C (2011) Research of the mining algorithm based on distributed database. In: Proceedings of the 8th international conference on fuzzy systems and knowledge discovery (FSKD), pp 1252–1256
4. Zhou L, Li S, Xu M (2010) Research on algorithm of association rules in distributed database system. In: Proceedings of the 2nd IEEE international Asia conference on informatics in control, automation and robotics, pp 216–219
5. Kantarcioglu M, Clifton C (2004) Privacy preserving distributed mining of association rules on horizontally partitioned data. IEEE Trans Knowl Data Eng 16:6
6. Wang H, Hu C, Liu J (2010) Distributed mining of association rules based on privacy-preserved method. In: Proceedings of the 3rd IEEE international symposium on information science and engineering, pp 494–497
7. Nguyen XC, Le HB, Cao TA (2012) An enhanced scheme for privacy-preserving association rules mining on horizontally distributed databases. In: Proceedings of the IEEE international conference on computing and communication technologies, research, innovation, and vision for the future (RIVF), Vietnam, pp 1–4
8. Rozenberg B, Gudes E (2006) Association rules mining in vertically partitioned databases. IEEE Trans Knowl Data Eng 59:376–396
9. Ge X, Yan L, Zhu J, Shi W (2010) Privacy-preserving distributed association rule mining based on the secret sharing technique. In:

Proceedings of the 2nd international conference on software engineering and data mining (SEDM), pp 345–350
10. Yu-quan Z, Yang T, Geng C (2011) A privacy preserving algorithm for mining distributed association rules. In: Proceedings IEEE of the international conference on computer and management (CAMAN), Wuhan, pp 1–4
11. Yao AC (1982) Protocol for secure sum computations. In: Proceedings of the IEEE foundations of computer science, pp 160–164
12. Zhong S (2007) Privacy preserving algorithms for distributed mining of frequent itemsets. Inf Sci 177(2):490–503
13. Qodmanan HR, Nasiri M, Bidgoli BM (2011) Multi objective association rule mining with genetic algorithm without specifying minimum support and minimum confidence. Expert Syst Appl 38:288–298
14. Jin Y, Sendhoff B (2008) Pareto-based multi objective machine learning: an overview and case studies. IEEE Trans Syst Man Cybern Part C Appl Rev 38(3)
15. Bhowan U, Johnston M, Zhang M (2011) Evolving ensembles in multi-objective genetic programming for classification with unbalanced data. In: Proceedings of the 13th annual genetic and evolutionary computation conference, Ireland, pp 1331–1338
16. Yang G, Mabu S, Shimada K, Hirasawa K (2011) A novel evolutionary method to search interesting association rules by key words. J Expert Syst Appl 38(10):13378–13385
17. Yan X, Zhang C, Zhang S (2005) ARMGA: Identifying interesting association rules with genetic algorithms. Appl Artif Intell 19(6):677–689
18. Sindhya K, Sinha A, Deb K, Miettinen K (2009) Local search based evolutionary multi-objective optimization algorithm for constrained and unconstrained problems. In: Proceedings of the 11th IEEE conference on congress on evolutionary computation, Piscataway, pp 2919–2926
19. Jabeen H, Baig AR (2010) Review of classification using genetic programming. Int J Eng Sci Technol 2(2):94–103
20. Shi X-J, Lei H (2008) A genetic algorithm-based approach for classification rule discovery. In: Proceedings of the IEEE international conference on information management, innovation management and industrial engineering, Taipei, pp 175–178
21. Oltean M, Diosan L (2009) An autonomous GP-based system for regression and classification problems. Appl Soft Comput 9:49–60
22. Keshavamurthy BN, Toshniwal D, Eshwar BK (2012) Hiding co-occurring prioritized sensitive patterns over distributed progressive sequential data streams. J Netw Comput Appl 35:1116–1129
23. Patrick TO (1999) Principles of distributed database systems. Prentice-Hall, NJ