ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 8(18), DOI: 10.17485/ijst/2015/v8i18/55109, August 2015* ISSN (Online) : 0974-5645

A Improved Incremental and Interactive Frequent Pattern Mining Techniques for Market Basket Analysis and Fraud Detection in Distributed and Parallel Systems

K. K. Sherly[1*] and R. Nedunchezhian[2]

[1]Department of Information Technology, Toc H Institute of Science and Technology, Ernakulam - 682313, Kerala, India; sherly.shilu@gmail.com
[2]Department of Computer Science and Engineering, Sri Ranganathar Institute of Engineering and Technology, Coimbatore - 641110, Tamilnadu, India; rajuchezhian@gmail.com

Abstract

Objectives: To develop a memory efficient, incremental and interactive distributed FPM having less communication and synchronization overhead with good load balancing capability, to analyze the dynamic transactional data in a distributed database. **Methods/Analysis:** This technique adopts prefix based equivalence class partitioning scheme to generate frequent item sets without generating local frequent sets with low memory consumption. This approach uses a range of support values to update the frequent patterns with less time complexity. This paper proposes distributed FPM techniques with both count distributed and compressed data distributed parallel approaches. The performance of the algorithms are tested and compared with popular distributed FPM algorithms using standard datasets. **Findings:** To deal with the massive dynamic data stored in distributed databases, this approach develops three distributed frequent set generation algorithms, which update frequent patterns by reusing the previously stored pattern information with no complex calculations or data structures. The proposed approaches also provide the user with the facility to interactively adjust the minimum support value as per their own conveniences by keeping the nearly frequent itemsets with the help of two minimum support thresholds (low, high). Measures have been taken to reduce the additional itemset storage and computations as well as to achieve good load balancing with low communication and synchronization overhead. Since the proposed algorithms adopt prefix based equivalent class partitioning technique at each n-itemset level and undergo four levels of itemset filtering to remove infrequent items from each class before calculating the individual item count, the inter node communication required is less in this approach. To eliminate the drawbacks of both count and data distribution approaches one of the algorithms proposed adopts a hybrid approach which distributes the compressed data only once, hence communication overhead is less compared with other DD algorithms. **Conclusion/Application:** The proposed distributed techniques reduce memory utilization and itemset comparisons compared to the existing approaches. The performances are tested and evaluated for market analysis and online credit card fraud detection applications.

Keywords: Credit Card Fraud Detection System, Incremental Distributed Frequent Pattern Mining, Interactive Parallel Mining Techniques, Market Basket Analysis, Prefix Based Equivalence Class Partitioning Approach

1. Introduction

Association rule mining is one of the data mining tasks which have been applied for market analysis[1-3]. Frequent Pattern Mining (FPM) plays a key role to obtain associations and correlations among items in a large transactional dataset. A large number of algorithms have been proposed for frequent pattern generation. Almost all these algorithms are used for offline analytical task. With the increase in the demand of various real time business applications, online analysis is on demand. Fraudsters are coming up with new methods every day and between year

* *Author for correspondence*

2012 and 2013, there has been nearly 15 percent increase of card frauds reported by cardwatch. Thus an incremental parallel frequent pattern mining techniques are essential to analyze the dynamically growing databases[4,5]. This study proposes three distributed frequent pattern mining algorithms to analyze the transactional data in the distributed database and to detect the online fraudulent transactions.

The popular algorithm Apriori[1] forms the foundation for static frequent pattern mining. It generates candidate itemsets iteratively, which makes the computational cost very high. Instead of using generate and test paradigm of Apriori, FP-tree approaches[2,6] encode the dataset using a compact tree structure and directly extracts the frequent itemsets from this structure. But it has to generate conditional pattern bases and sub-conditional pattern tree recursively. An interactive mining algorithm CARMA[5] provides a lower and upper bound for its support for each set and generates frequent patterns in two database scans. Thus the user can interactively adjust the support and confidence at any time. A dynamic algorithm CanTree[6] facilitates incremental mining as well as interactive mining with one database scan. It keeps the entire transactions in the CanTree for preparing frequent itemsets; thus it requires more memory. An incremental binary tree algorithm IMBT is presented by Yang C et al.[7] in which each node of the tree represents one of all the possible combinations of items in the entire dataset. It also provides incremental and interactive mining with less processing and I/O time but requires more memory to keep all combinations of items in the database. Sherly K, et al.[8] proposes IAPI Quad-Filter (Interactive and Adaptive Partitioned Incremental FPM) algorithm for incremental frequent pattern mining in large databases to solve the space and computational complexity. But it requires more than two database scan (equivalent to the number of frequent items), thus the data fetching time is fairly high.

A potential solution for improving the performance and scalability in frequent pattern mining from dynamically growing database is to parallelize the mining algorithms. An algorithm PDM[9] is proposed for parallel mining which is an adaptation of the DHP algorithm [3] in the distributed environment. In PDM each node computes the globally large ite msets by exchanging the support counts of the candidate sets, thus $\mathbf{O}(n^2)$ messages are required for support count exchange among n nodes for each candidate set. A tree-partition algorithm for parallel mining of frequent patterns on shared-memory structures is presented in[10]. It builds one FP-Tree of the entire database, then partitions it into several independent parts and distributes them to different threads. This approach uses a Master/Slave Model. The parallel implementation of Apriori algorithm based on MapReduce framework[11] is suggested for processing huge datasets using a large number of computers. Iko P, et al.[12] proposed a parallel FP growth algorithm on distributed environment. It also introduced a novel notion of path depth to break down the granularity parallel processing of conditional pattern bases. But these parallel algorithms are not suitable for incremental database. A parallel IMBT[13] structure is proposed in distributed system to enumerate the support count of each itemset in an efficient way after the new transactions are added or deleted.

Several researchers have shown interest on credit card fraud detection with special emphasis on data mining. Jianyun et al.[14] have presented a framework for detecting fraudulent transactions in an online system using frequent pattern mining technique. This paper describes an FP tree based method to dynamically create user profile. FP tree performs well for high support but the size of the tree increases as minimum support reduces. Ghosh and Reilly have proposed credit card fraud detection with a three layer, Feed-forward Neural network[15]. Neural network requires long training time. For improving the speed in credit card fraud detection, Syeda et al.[16] have used Parallel Granular Neural Networks (PGNNs). It suffers load imbalance problem when more number of processors are used. Aleskerov et al. introduces CARDWATCH algorithm[17] for fraud detection. It provides an interface to a variety of commercial databases, but it requires one network per customer. Chiu and Tsai[18] have proposed a web service based collaborative scheme for fraud detection in the banking industry. In this model participant banks share the knowledge about fraud patterns prepared using Apriori algorithm in a heterogeneous and distributed environment. It undergoes multiple database scan to generate fraud pattern. Stolfo et al.[19] present a Fraud Detection System (FDS) using metalearning techniques i.e. by combining and integrating a number of separately built classifiers. This technique doesn't consider short term behavioral changes of card holders.

Fan et al.[20] suggest the application of distributed data mining in credit card fraud detection. But it requires more computational resources and has the incompatibility schema problem. To address the skewness of data problem in credit card transaction Phua et al.[21] suggest the use of Meta classifier similar to[19]. They consider naïve Bayesian,

C4.5, and Back Propagation neural networks as the base classifiers. Bayesian networks are more accurate and faster to train but are slower when applied to new instances. C4.5 can output accurate predictions, but scalability and efficiency problem occurs when applied to large data sets. Amlan Kundu et al.[22] suggest a model BLAST-SSAHA Hybridization technique for fraud detection using a two stage sequence alignment method by combining anomaly and misuse detection techniques. Abhinav Srivastava et al.[23] have proposed a Hidden Markov model for detecting fraudulent transactions. Time series and Markov model have time complexity problem with large data sets. Renugadevi et al.[24] presents a behavioral pattern mining technique which prepares both personalized and aggregate model to detect the online fraud by implementing the Classifiers Naïve Bayesian and Random Forest.

Most of the above mentioned approaches are supervised methods which require labeled data to train the classifiers for both genuine, as well as fraudulent transactions. Hence these methods are able to detect the known attacks only. This paper proposes an unsupervised fraud detection technique, which detects unusual behaviors using clustering and association rule mining approaches. It prepares adaptive frequent patterns from both legal and fraudulent transaction history. Thus unknown types of fraud can also be detected.

2. Basic Terminologies

Let D be a database with N number of variable length transactions T. Let I be the item domain, $\{I_1,I_2,.....I_m\}$ and transaction T is a set of items such that $T \subseteq I$. Let X and Y are sets of items. An itemset with k elements is called k-itemsets. Database D is a multiset of subsets of I, $T \in D$. Association rule is a relationship between two or more items of the form $X \rightarrow Y$ where $X,Y \subseteq I$ and $X \cap Y = \emptyset$. Such a rule reveals that the transactions in the database containing items in X tend to contain in Y. Support of an itemset X in a transaction sequence is the fraction of all transactions containing the itemset i.e. the frequency occurrence of X in D. Support S of an association rule $X \rightarrow Y$ in the transaction set D is the percentage of transactions in D that contain items both X and Y. An itemset X is called a frequent itemset, if the support of X in D is greater than the minimum support threshold set by the user. The rule $X \rightarrow Y$ holds in D with confidence C where

$$C = \text{support } D(X \cup Y)/\text{support } D\ (X) \quad (1)$$

A partition on D is

$$\pi(D)= \{Di \mid i \in I, Di \subseteq D\} \text{ where } Di \neq \emptyset \quad (2)$$

$$\text{If } Di \cap Dj = \emptyset \text{ for each pair i, j} \in I, i \neq j, \text{ and}$$

$$\cup_{i \in I} Di = D \quad (3)$$

2.1 Problem Definition

The problem is to identify all interesting frequent patterns in an interactive and incremental manner to support the market analysis and to prevent the online financial fraud. Initially the database D is logically partitioned into r partitions of size Z. To assist interactive and incremental mining two minimum support values used here are: Sl, Sh namely, lower minimum support value and upper minimum support value. It creates two category itemsets: Frequent ($Fset$), Nearly Frequent ($NFset$). Itemset X is Frequent if support (X) $\geq Sh$ and Nearly frequent if $Sl \leq$ support(X) $\leq Sh$. Pn represents a partition number at which a $NFset$ has been last updated. Let f be the frequent item domain, $\{f_1, f_2,.....f_n\}$ in the ascending order of occurrence count. Each frequent item is associated with a co-occurring itemset list Cf_i, refers to the subset of frequent items, whereas Cf_1 be the co-occurring item list of frequent item f_1, $Cf_1 = \{f_2, f_3,.....f_n\}$, $Cf_2 = \{f_3, f_4,.....f_n\}$ be the co-itemset list of f_2. This indicates that as the frequency of occurrence is more the number of co-occurring items considered for frequent itemset mining gets reduced. Let $Cf = \{Cf_1, Cf_2,...Cf_{n-1}\}$ is a nested family of Cfi sets where $Cf_{i+1} \subseteq Cf_i$ for all i = 1, 2,…..n-1.

3. Proposed Distributed IAPI Algorithms

Distributed incremental parallel mining approaches are required where the databases are distributed and growing dynamically. Fraudsters consistently develop new methods of stealing funds and identities and consequently consumers are increasingly losing confidence in their bank's ability to protect them from fraud. Banks and web merchants face enormous challenges in preventing illegal transactions. The major challenge is that millions of transactions are processed daily and the data are highly skewed, i.e., many more legitimate transactions occur than fraudulent ones. For an individual bank, the ratio of fraud transactions to normal transactions is extremely low. Online fraud can happen from anywhere in the world. If banks can share their individual fraud transactions to a central center, the integrated data will extract more update fraud patterns to help banks enhancing their fraud detection. The two types of distributed approaches used

in this study are compressed data distributed and count distributed parallel mining approaches. Compressed Data Distributed approach (CDD-IAPI) is suitable for huge business organizations that are distributed into different geographical locations. In fraud detection, participating banks may not be interested in revealing the fraud details to other banks; thus count distributed approach with a client server model (CD-IAPI) may be best suited for hiding the fraud information from the participating banks.

3.1 Count Distributed IAPI

CD algorithms have less communication overhead compared to DD algorithms. In CD-IAPI algorithm, first every node computes the local support of each item and sends to the server. Further server computes their global support and identifies the global frequent 1-itemsets. Also prepares a co-occurring item list, ***Cf*** for each frequent item and broadcast them to all the nodes. Second each node collects all frequent item transaction groups separately with their respective co-occurring itemsets and finds the local count of each item in the selected group. Then each node sends the local count of items from each group to the sever and identifies the global frequent 2-itemsets. Then proceed to find the higher itemsets by sub grouping each 1-itemset transaction groups into separate 2-itemset transaction groups according to the global frequent itemsets obtained in the previous pass. Follow the same procedure to obtain the higher frequent itemsets. Functional details of CD-IAPI algorithm is shown in Figure 1.

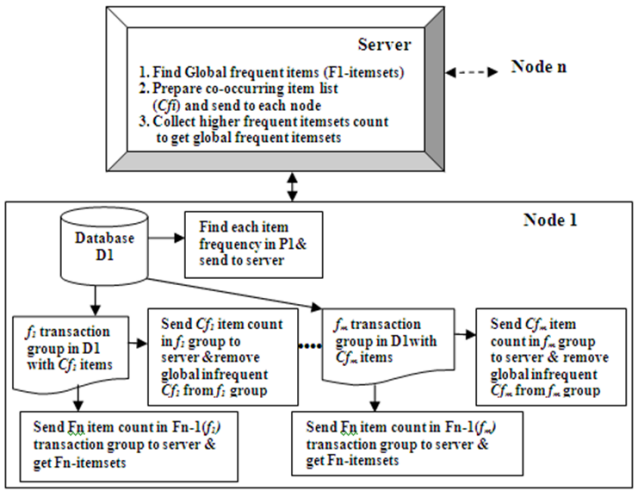

Figure 1. Functional block diagram of CD-IAPI.

3.1.1 CD-IAPI Algorithm

Input:

D: Transaction database contain ***N*** transactions (T_1, T_2, ... T_N), horizontally partition

D into ***n*** non-overlapping partitions (P_1, P_2, ... P_n) and sort the items of each transaction in the order of item code.

Sl: low minimum support value

Sh: user selected minimum support (***Sh*** > ***Sl***)

Output:

Complete set of frequent item sets

1. For each node do
   - Read local partition and find local frequency *flocal*(i) for each item i;
   - Send *flocal*(i) of each item i to the server
2. In server
   - Ftotal(i) = Σ *flocal*(i) for each i
   - $F_{1\text{-}itemset}$ ={i | Ftotal(i) ≥ ***Sh*** for each item i} and send to each node
3. For each node do
   - Prepare co-occurring itemset list ***Cf*** = {Cf_1, Cf_2, ...Cf_{m-1}}| $Cf_1 \supset Cf_2 \supset$.....$\supset Cf_{n-1}$
   - where Cf_i = {{f_{i+1},f_{i+2}...f_m}| frequency (f_{i+1},f_{i+2},....f_m) ≥ frequency (f_i)}
   - Assign ***m*** buffers to store ***m*** frequent items transaction groups separately with the corresponding ***Cfi*** items.
   - Read each transaction and store in to each f_i buffer if it contain the assigned f_i , remove items that are not in the Cf_i list from each transaction.
   - Find the frequency of each Cf_i item *flocal*(Cf_i)in each buffer and send to the server and repeat the above steps to find the higher frequent itemsets of each f_i
4. In server
   - For each ***fi***
   - Ftotal (Cf_i) = Σ *flocal*(Cf_i) for each Cf_i from each node
   - $F_{n\text{-}itemset}$= {Cf_i }| Ftotal(Cf_i) ≥ ***Sl*** for each Cf_i item and send to all other nodes
   - ***Fset*** = $F_{n\text{-}itemset}$ (f_i) if support($F_{n\text{-}itemset}$) ≥ ***Sh***) for each fi where n=2, 3,……l
   - Else ***NFset*** = $F_{n\text{-}itemset}$ (f_i) with last counted partition number ***z.***

3.2 Count and Compressed Data Distributed IAPI

Unlike other DD approaches, to eliminate the drawbacks of both count and data distribution approaches this algorithm adopts a hybrid approach. This algorithm finds the frequent 1-itemsets and 2-itemsets with count distribution approach. Then to find the higher itemsets, algorithm assigns separate node for each frequent

1-itemset and send the transactions which include only the assigned frequent item after removing both infrequent 1-itemsets and 2-itemsets to the respective nodes. The data is distributed only once, hence communication overhead is less compared with other DD algorithms. The functionality details are shown in figure 2. To reduce the communication overhead and to improve the load balance first ***n*** frequent items are assigned to ***n*** nodes according to the count of item in these nodes, (node which has item count more is assigned with that item). Each node calculates the count of items in each transaction group and sends to the assigned nodes to get the global frequent 2-itemsets of each group. Then after removing the infrequent 2-itemsets from each group, the compressed transaction groups are sent to the assigned nodes. Each node proceeds with the higher frequent itemset generation process and whichever node finishes the itemset generation, sends request to the server for next frequent itemset generation. Server sends request to all other nodes to send the count of items in the frequent item transaction group from which the higher frequent set is to be generated, to the requested node. If more than one node requested for next frequent itemset generation assigns the transaction groups on first come first serve basis. If the requests arrived at the same instant then assign the transaction groups based on the count of next frequent item in the requested nodes (nodes which have more count is assigned first). Same procedure is repeated for the remaining frequent items. Finally all frequent itemsets generated are sent to the server and global frequent and nearly frequent itemsets are stored separately.

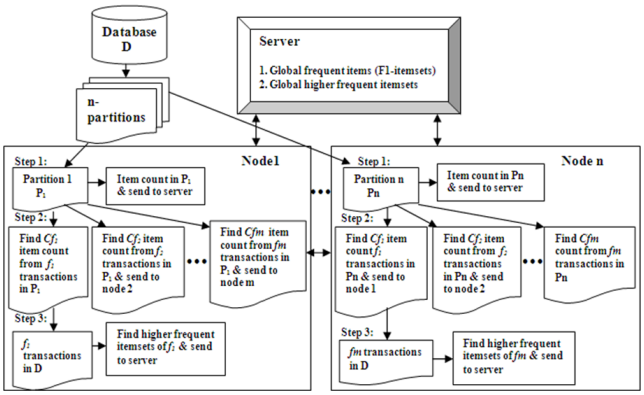

Figure 2. Functional block diagram of CDD-IAPI.

3.2.1 Working of CDD-IAPI FPM with Example

The working of CDD-IAPI can be illustrated using a sample dataset having 10 transactions, as given in Figure 3. Consider that the dataset is equally distributed among two nodes, Node1 and Node 2. During the first database scan Node1 and Node 2, calculate the individual item count concurrently and send to server to find global frequent items. Then server assigns Node 1 to find the higher itemsets of item ***f*** and node 2 with item ***e***. Then server generates co-occurring item list of each item and sends to both the nodes to find the higher itemsets of frequent items {a, b, c, d, e, f}. Both nodes scan the database second time and store each frequent item transaction groups with their respective co-occurring itemsets to separate buffers as shown in second step in the nodes (Figure 3). Then local counts of each item in the selected groups are calculated and send to the assigned nodes to identify frequent 2-itemsets. Then higher frequent itemsets of items ***f*** and ***e*** are generated in parallel by node 1 and node 2 respectively. After the completion of frequent itemset generation, generated itemsets are sent to the server and it assigns the node for next frequent itemset generation (item ***d***). According to this example, Node 1 is assigned with items ***b*** and ***f*** and Node 2 with items ***e*** and ***d.*** Item ***a*** has no co-items and item ***c*** has only one co-item whose count is obtained at the second scan.

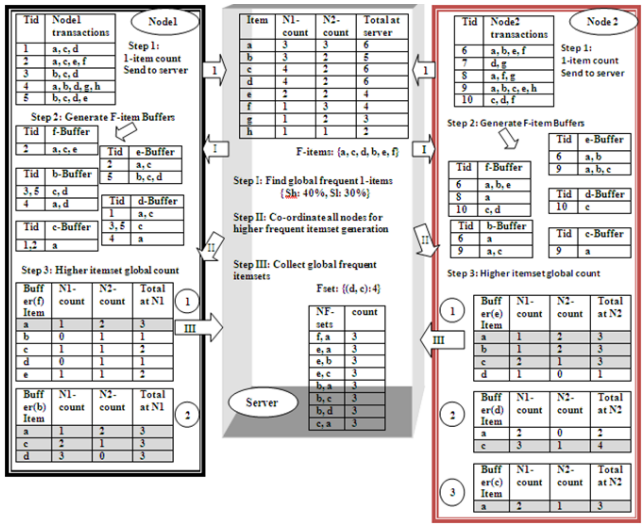

Figure 3. Working of CDD-IAPI with example.

In incremental mining new partitions are added at both nodes (Phase 2) and the count of each item in

the newly added partition is calculated at each node and send to the server. Server adds the count of items obtained from each node; then the sum gets added with the previous count to identify the present frequent items. Frequent item buffers in the newly added partition are generated and frequency of the higher frequent itemsets is obtained by both the nodes. Higher frequent itemsets count in the new partition is added with the previous count by the server to update the frequency of the existing ***Fset*** and ***NFset***. New patterns may get generated on adding new transactions made by the new customers as well due to the change in purchase behavior. Thus to reflect the pattern changes (Phase 3) old transactions may be removed and update the frequent patterns. Count of individual items in the removing partition is calculated by the respective nodes and gets deducted from the total count by the server. Further the frequency of the existing frequent itemsets in the removed partition is obtained by the corresponding nodes and is deducted from the previous count by the server. Higher frequent itemsets of newly created frequent items ($\mathbf{F1}_{new}$) is obtained by rescanning the remaining partitions by respective nodes in coordination with the server and update the ***Fset*** and ***NFset.***

3.2.2 CDD-IAPI Algorithm

Input:

D: Transaction database contain ***N*** transactions (T_1, T_2, ... T_N), horizontally partition

D into ***n*** non-overlapping partitions (P_1, P_2, ... P_n) and sort the items of each transaction in the order of item code.

Sl: low minimum support value

Sh: user selected minimum support (***Sh*** > ***Sl***)

Output:

Complete set of frequent itemsets

1. For each node do
- Read local partition and find local frequency *flocal*(i) for each item i;
- Send *flocal*(i) of each item i to the server
2. In server
- Ftotal(i) = ∑ *flocal*(i) for each i
- $F_{1\text{-}itemset}$ ={i | Ftotal(i) ≥ ***Sh*** for each item i} and send to each node
- Assign n nodes to find the higher frequent itemsets of first n frequent items
3. For each node do
- Prepare co-occurring itemset list ***Cf*** = {Cf_1, Cf_2, ...Cf_{m-1}}| $Cf_1 \supset Cf_2 \supset$.....$\supset Cf_{n-1}$
- where Cf_i = {{f_{i+1},f_{i+2}...f_m}|frequency(f_{i+1},f_{i+2},....f_m) ≥ frequency(f_i)}
- Assign ***m*** buffers to store ***m*** frequent items transaction groups separately with the corresponding Cf_i items.
- Read each transaction and store in to each f_i buffer if it contain the assigned f_i and remove items that are not in the Cf_i list from each transaction.
- Find the frequency of each Cf_i item *flocal*(Cf_i)in each buffer and send to the assigned nodes.
- Ftotal(Cf_i) = ∑ *flocal*(Cf_i) for each Cf_i at each node
- $F_{n\text{-}itemset}$= {Cf_i }| Ftotal(Cf_i) ≥ ***Sl*** for each Cf_i item and send to all nodes
- Remove infrequent items from each transaction in the corresponding buffers and send each buffer contents to the assigned nodes.
- Follow the same steps to obtain higher frequent itemsets of f_i ; further there is no need of sending item counts to other nodes.
- Each node proceeds with the higher frequent itemset generation procedures and whichever node finishes the itemset generation send request to the server for next frequent itemset generation.
- Server send request to all other nodes to send the count of items in the (n+1)th transaction group to the requested node.
- If more than one node requested for next frequent itemset generation assigns the transaction groups on first come first serve basis. If the requests arrived at the same instant then assign the transaction groups based on the count of next frequent item (nodes which have more count is assigned first).
- Repeat the above steps till there is no higher frequent itemset generation.
- Send all frequent itemsets of assigned ***fi*** with count to the server
4. In server
- ***Fset*** = $F_{n\text{-}itemset}$ (f_i) if support($F_{n\text{-}itemset}$) ≥ ***Sh***) for each fi where n = 2, 3,……l
- Else ***NFset*** = $F_{n\text{-}itemset}$ (f_i) with last counted partition number ***z***

Procedure Higher-frequentItemset-Generate (f_i-transactions(Buffer$_1$), ***Fn***)

//Fn_p : p^{th} item of $F_{n\text{-}itemset}$

1. Collect ***fi***-transactions contain selected $F_{n\text{-}itemset}$ i.e Fn_p to a new temporary buffer$_n$ and remove items having count ≤ Fn_p from each transaction in the buffer$_n$, ***p*** initialized to 0
2. Find frequency of each item in the selected Fn_p transaction group
3. $F_{(n+1)\text{-}itemset}$ = {$Fn_{(p+k)}$ | frequency($Fn_{(p+k)}$) ≥ ***Sl*** for each $Fn_{(p+k)}$ item where ***k***=1 to (***m-p***)
4. else remove $Fn_{(p+k)}$ from the Fn_p transaction group

5. Sort $F_{(n+1)\text{-itemset}}$ in ascending order
6. To obtain higher frequent itemsets of f_i do
7. if ($F_{(n+1)\text{-itemset}} \neq \Phi$) then
8. n = n+1 & Repeat above steps
9. else if p < size($F_{n\text{-itemset}}$) then
10. p = p+1 & remove $buffer_n$ content
11. else n = n-1 & remove $buffer_n$ content
12. Repeat above steps if n ≥ 2
13. else return

3.2.3 Incremental Mining

Rather than fixing single minimum support value IAPI uses a range of support values (***Sl, Sh***) for making the dynamic and the interactive mining faster. The incremental mining procedure of CDD-IAPI algorithm is given below.

1. For each node do
- Read newly added local partition and find frequency $flocal_{new}(i)$ for each item i;
- Send $flocal_{new}(i)$ of each item i to the server
2. In server
- $Ftotal_{new}(i) = \sum flocal_{new}(i)$ for each i
- Updated item count $UFtotal_{new}(i) = Ftotal(i) + Ftotal_{new}(i)$
- $UpdatedF_{1\text{-itemset}} = \{i \mid UFtotal(i) \geq Sh$ for each item i$\}$& inform each node to find higher frequent itemsets
- If new $F_{1\text{-itemset}}$ then set $Cf_{new} \supset Cf_1$ and update existing ***Cf*** and find its higher frequent itemset by collecting the transactions containing the new frequent item from the entire old partitions. Then include it in the ***Fset*** list.
3. For each node do
- Read newly added local partition and find the higher frequent itemsets of the assigned frequent item ***fi*** using the same ***Cfi*** sets.
- Send all FrequentItemsets of assigned ***fi*** with count to the server
4. In server
- Collect FrequentItemsets of each frequent item ***fi*** from newly added partitions of each node
- Update the count of existing ***Fset*** and ***NFset*** with the frequent itemsets of new partition.
- If any of the existing ***Fset*** are not updated, collect its frequency from the corresponding node buffer and update it.
- If any existing ***Fset*** become infrequent shift it to ***NFset*** list, similarly any existing ***NFset***s become frequent do vice versa.
- If any new ***Fset*** obtained, conduct a possibility test and if possible to be frequent find its global count by rescanning the entire old partitions.

3.3 CDD-Parallel IAPI

To reduce the computational cost as well as I/O overhead while finding the frequent itemsets, IAPI algorithm collects each frequent item transaction group into separate buffers and processes them separately. The frequent set generation time can be further improved by processing these buffers in parallel using multiple processors at each node. Thus this approach proposes CDD-Parallel IAPI (CDD-PIAPI) algorithm for a faster frequent itemset generation. Functional block diagram of CDD-PIAPI algorithm is shown in Figure 4.

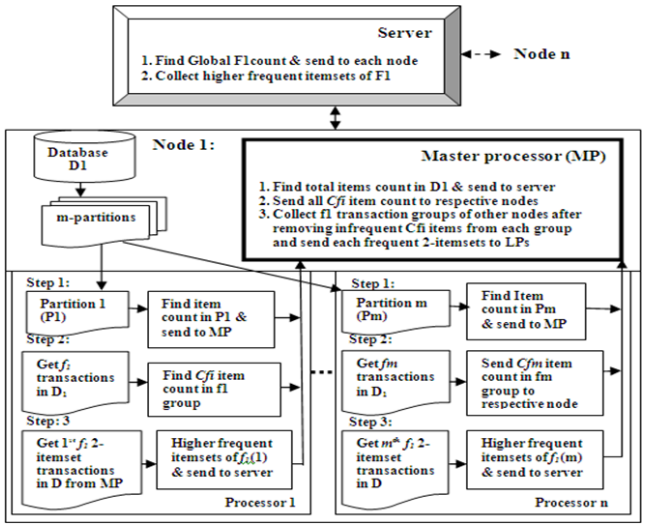

Figure 4. Functional block diagram of PCDD-IAPI.

Similar to CDD-IAPI this algorithm finds the frequent 1-itemsets and 2-itemsets with count distribution approach using parallel processors at each node. In this approach local database of each node is partitioned in to n non-overlapped horizontal partitions. The count of distinct items from each partition is obtained by ***n*** Local Processors (LP) simultaneously and sends them to the Master Processor (MP) to calculate their total count. Then MP of each node sends the local count of each item to the server to identify the global frequent 1-itemsets and server assigns separate node for each frequent item to find their higher length itemsets. Master processors of each node collects the assigned frequent item transactions groups from all other nodes after removing the infrequent 2-itemsets and finds their higher length itemsets with IAPI approach in parallel manner using their LPs. Algorithm steps are given below.

3.3.1 CDD-PIAPI Algorithm

1. Partition the database at each node into n horizontal partitions
2. For each local processor at each node do
   Read local partition and find local frequency flocal(i) for each item i;
3. In master processor at each node do
   $F_{node}(i) = \Sigma$ flocal(i) and send to server
4. In server do
- $F_{1\text{-itemset}} = \{i \mid \Sigma F_{node}(i) \geq Sh$ for each i$\}$and assign node(i) to find higher large itemsets
- Prepare co-occurring itemset list $Cf_i = \{f_{i+1}, f_{i+2} \ldots f_m\}$ |count $(f_{i+1}, f_{i+2}, \ldots f_m) \geq$ count $(f_i)\}$ for each f_i & send to Master Processor (i)
5. At each node
- Read each transaction and send to LP
- For each local processor do
- Collect transactions contain frequent item f_i in to buffer$_i$ and remove items that are not in the Cf_i list from each transaction.
- Find the local frequency of each Cf_i item in the selected f_i-transaction group and send to the master processors of the assigned nodes.
6. Master processor calculates the global frequency of each Cf_i i tem in the assigned group and sends the frequent item list to all nodes.
7. At each node LP removes the infrequent items in the assigned transaction group and sends the compressed transactions to the respective nodes.
8. Each node proceeds with the higher frequent itemset generation procedures by assigning each LP to each frequent n-itemsets transaction groups and whichever node finishes the itemset generation send request to the server for next frequent itemset generation.
9. Server sends request to all other nodes to send the count of items in the (n+1)th transaction group to the requested node.
10. If more than one node is requested for next frequent itemset generation, assign the transaction groups on first come first serve basis. If the requests arrived at the same instant, then assign the transaction groups based on the count of next frequent item (nodes which have more count are assigned first).
11. Repeat the above steps till there is no higher frequent itemset generation.

4. Proposed Fraud Detection System

An intelligent Fraud Detection System (FDS) monitors card transactions, collects data from the current and previous transactions and processes this data to compute a transaction score for the current transaction. In every credit card transaction there are three stages of verifications. At the first stage of verification the merchant sends the card details and the amount of purchase to the card issuing agency for the identity verification. After passing the authenticity check, the merchant verifies the transaction data to make sure that card owner is doing the transaction. As a security measure the third stage verification is done by the Fraud Detection System (FDS) at the Bank. Transaction data stored in the bank database contains the attributes such as credit card number, account number, transaction amount, transaction date, merchant and city. In this paper we consider six transaction attributes such as transaction amount, time, IP address of the machine from which the order is placed, Card holder's address, item purchased and delivery/ shipping address, which are relevant for identifying the user spending behavior. Every credit card has a credit limit, the transaction within the credit limit is considered as valid transaction. The transaction time, IP address of the order placing machine and the delivery address are not restricted. The transaction time depends on the card holder's convenience, surroundings and lifestyle. The spending behavior such as amount, item, city and frequency of purchase may change over time.

4.1 Data Discretization

A credit cardholder makes different kinds of purchases of different amounts over a period of time. Therefore these raw data need to be classified into specific groups for generating frequent patterns and association rules. In this approach first phase converts these continuous parameters into categorical parameters using efficient clustering and classification techniques. In the second phase frequent patterns and association rules are generated using Parallel and distributed IAPI algorithms.

The purchase time is categorized into two slots, namely, morning (*MR*) and evening (*EV*).The second attribute the transaction amount be quantized into three different levels - Low (*CL*), Medium (*CM*) and High (*CH*) for convenience, using K-means clustering algorithm. Each individual transaction amount usually depends on the corresponding type of item purchased. Based on the spending habit of individual card holders purchase amount ranges can be determined dynamically using K-Means clustering algorithm on their past transactions.

The third attribute IP address of order placing machine is also an important parameter which can assist fraud detection easily. Classify them into two category local/small group (*SG*) and dynamic/large group (*LG*). The fourth attribute shipping address is categorized into two groups, user address (*UA*) and other address (*OA*). The fifth attribute item purchased can be categorized into five groups such as Groceries (Gr), Electronic items (EI), Gold (Gl), Medical (MD) and Miscellaneous (Mi) purchases. Some merchant may sell variety items, the item purchased from these merchants may consider as miscellaneous for convenience. Categorization of the items purchased can be done by referring the merchant profile database and applying If-THEN classification technique.

4.2 Two Stage IAPI Enabled Fraud Detection System

Figure 5 shows the structure of the Fraud Detection System (FDS) proposed to implement at the banking sector. This has two sections; first part is to generate frequent transaction patterns of each card holder from their legal transaction history using parallel IAPI algorithm also to generate frequent fraud patterns from the fraudulent transaction history of all participating bank using CD-IAPI algorithm. Second part verifies the newly arrived transactions with the frequent legal and fraud patterns and distinguishes fraudulent transaction from legal transactions. Second section has two stages, the first stage identifies the anomalous behavior and the second stage confirms the misuse.

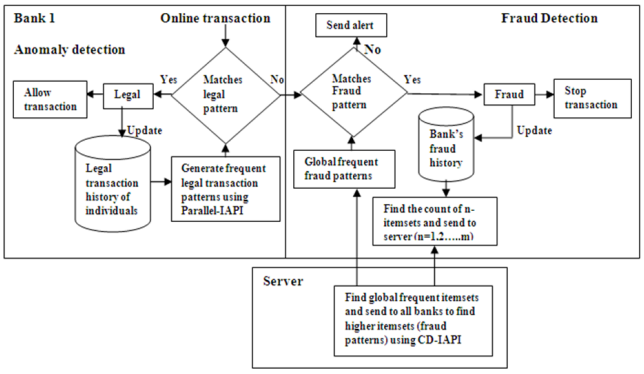

Figure 5. Structure of proposed FDS.

This model checks the similarity of the incoming transactions with the profiled legal frequent transaction patterns of the card holder and high score transactions are considered as genuine transactions. Low scored transactions have suspected anomalies and the second stage confirms whether the detected anomalies were due to fraudulent transaction or short term behavioral changes by comparing with the fraud patterns generated. If the similarity score with fraud patterns is high, the transaction is highly suspicious; then alarm is generated and locks the transaction. Low scored transaction may be a new type of attack; to reduce the detection cost alarm will be generated only when the transaction amount is higher than a certain threshold value and an alert message is sent to the user for low amount transaction. After getting the confirmation from the user the transactions are recorded in the respective history database and further protection should be performed for the detected fraud.

5. Experimental Setup and Performance Analysis

Functionalities and effectiveness of the proposed IAPI algorithms were tested with market basket datasets T10I4D100K prepared by IBM Almaden Quest research group and a Synthetic dataset. This algorithm is developed and tested using programming language Java and MySQL Server 4.1 on Xeon 6 core processor with GPU as server and 10 to 30 nodes with Intel dual core processor systems having Debian 7.0 OS. Execution time and memory utilization are compared for different number of nodes with various support threshold values and different sized partitions as well as with different number of partitions in both datasets.

Experimental results show that execution time is directly proportional to the size of the dataset when the minimum support value remains constant (Figure 6). CDD-IAPI algorithm designed in this research work has less communication overhead compared to CD-IAPI approach. Thus time required to generate frequent pattern in CDD-IAPI is less compared with CD-IAPI. The speed of pattern generation of CDD-IAPI is further improved by using CDD-Parallel IAPI algorithm. Updating of the frequent sets on addition of new data and deletion of old data requires less time compared with the initial pattern creation time. From the test results it is observed that updating time is related with the heterogeneity of the data, i.e. if new frequent 1-itemset generated, then it requires entire database scan, else previous information can be used and require less time (5%-30% of the initial pattern creation time) depends on the number of frequent itemsets (Figure 8). It is also observed that, when the

support threshold reduces, the number of frequent items increases, thus execution time required is more. Due to heterogeneity of dataset there are chances of reducing the number of frequent items, even though the dataset size increases. The test results illustrate that the execution time and the memory requirement of all IAPIs directly depend on the number of frequent items in the dataset. Figure 7. shows that the execution time gets reduced when the number of nodes increases.

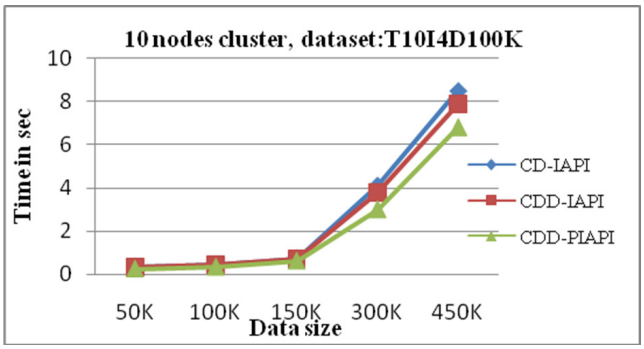

Figure 6. Execution time comparison of the proposed algorithms.

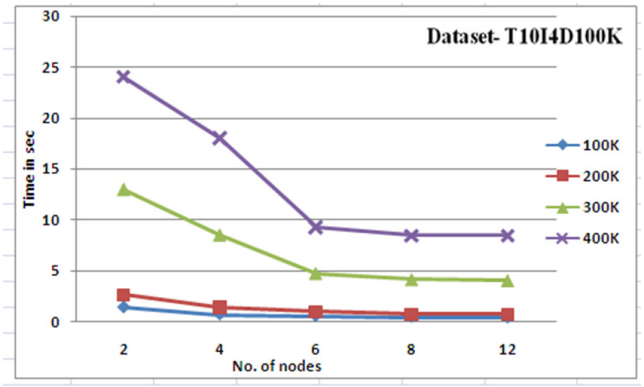

Figure 7. Execution time comparison with variable number of nodes.

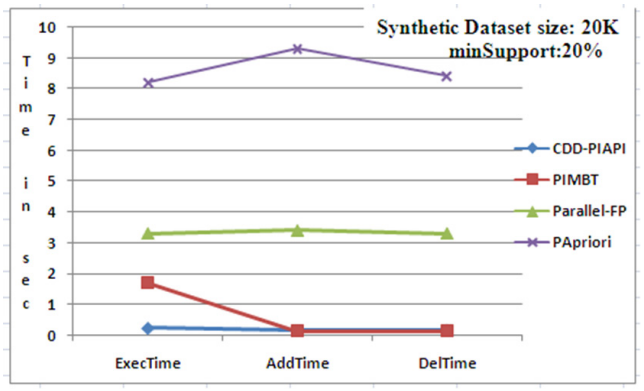

Figure 8. Pattern update time comparison.

5.1 Performance Comparison

Performance of CD-IAPI Algorithm, CDD-IAPI Algorithm and Parallel CD-IAPI Algorithm are compared with three parallel algorithms Parallel FP- tree, Parallel-IMBT and parallel-Apriori using T10I4D100K dataset and a synthetic dataset. Experimental results shown in Figure 9. illustrates that IAPI algorithms generate frequent itemsets in less time and requires less memory compared with Parallel-IMBT and Parallel-Apriori. IAPI algorithms requires only two database scan for frequent itemset generation where as Apriori has to read the entire database in every pass. FP-growth shows an outstanding improvement over Apriori; but it has to generate conditional pattern bases and sub- conditional pattern tree recursively. Thus initial pattern creation time of IAPI is less compared with Apriori and FP tree. It is observed that IMBT tree requires more time to create and more memory to store the entire tree. Thus it may not be suitable for datasets having more number of distinct items. In FP-Tree addition of new transactions may require reconstruction of the tree. Apriori also doesn't support incremental mining. Since IAPI algorithms use a range of values for minimum support: lower minimum support ***Sl*** and upper minimum support ***Sh,*** addition and deletion of dataset doesn't introduces much time to update the frequent patterns. IBT approach doesn't need to predetermine the minimum support threshold and scans the database only once. Thus IMBT requires less time to update the frequent sets on addition and deletion of data than other algorithms.

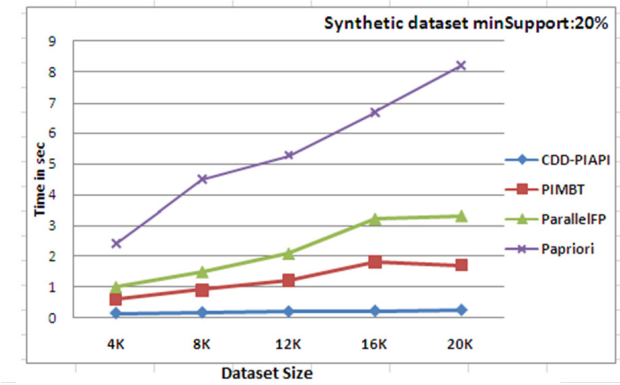

Figure 9. Performance comparison of popular FPM.

Transaction history of customers is required to analyze the spending behavior. No credit card companies are ready to share their data for testing the efficiency of the system. Thus experiments are conducted on synthetic dataset and past few years' real world card transaction data of few

credit card holders, who belong to three spending groups: low, medium and high. The relevant attributes are selected and categorized into specific category. Preprocessing of the transaction amount is done using K-Means clustering algorithm and frequent transaction sets are identified using Parallel IAPI and CD-IAPI algorithms.

The performance of the proposed model is tested and compared with the model developed by Jianyun et al. [15] and Chiu and Tsai [19] using synthetically generated data of different category spending behavior and minimum support, with the help of a transaction simulator developed by us. Frequent itemsets generated with low minimum support value can profile a user's behavior more accurately. The execution time of proposed IAPI algorithms directly depend on the number of frequent 1-itemsets generated whereas FP growth algorithm used by Jianyun et al. requires more time to construct FP tree for small support values. Apriori algorithm used to generate fraud pattern in Chiu and Tsai model also requires more time due to multiple database scan.

The accuracy of the model is tested with a test set of both real-life and synthetic data that were not used to generate association rule with different support values and calculated the percentage of test set tuples that are correctly detected by the model. Standard performance metrics: True Positive (TP) and False Positive (FP) are used to analyze the effectiveness of the system. Percentage of fraudulent transaction identified as fraud is TP and percentage of genuine transactions identified as fraud is FP. It is found that the accuracy of the system is dependent on the minimum support percentage. TP reduces with low support values and FP increases with high support percentages. In our approach we are able to get 90% and above TP and up to 10% FP with moderate support values. Variation in TP & FP with different minimum support values observed with the proposed method is shown in Figure 10.

By comparing the results reported in the literature survey we found that Maes et al. [14] have achieved 70% TP, 15% FP by applying neural network and 74% TP, 15% FP by applying Bayesian belief network. Abhinav Srivastava et al. [24] have achieved an overall accuracy of 80% even under large input condition variations which is much higher than the method proposed by Stolfo et al. [20]. Aleskerov et al. [18] obtained a fraud detection rate of 85%. Depending on the mode of implementation and the selected operating point for fraud detection Ghosh and Reilly [16] brought about a fraud loss reduction from 20% to 40%. Amlan Kundu et al. [23] have achieved about 85% TP and up to 5% FP with moderate profile size. But it is not much effective for very large profile size.

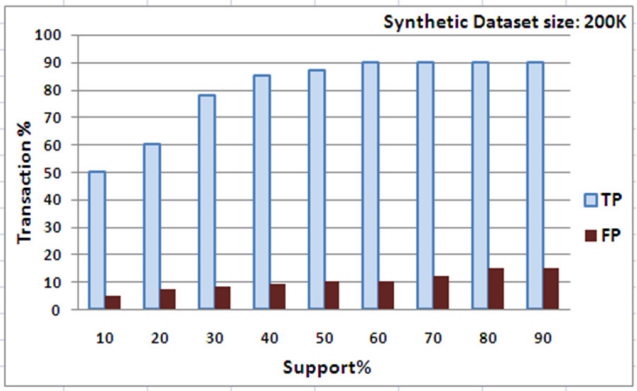

Figure10. Variation of TP-FP with different Support.

6. Conclusion

The hybrid of the anomaly and misuse detection models can improve the efficiency of fraud detection systems. Credit card transaction database of bank customers is utilized to conduct experiments. Most of the frequent pattern mining algorithms have complex structures and requires more time to generate frequent patterns. This approach uses three distributed frequent set generation algorithms, which incrementally generate frequent patterns and identifies the anomalies with less time and space complexity resulting in speedy decisions irrespective of the data set size with no complex calculations or data structures. The most attractive feature of IAPI algorithms is that the user can interactively adjust the support values with less time to update the frequent patterns. To eliminate the drawbacks of both count and data distribution approaches CDD-IAPI algorithm adopts a hybrid approach which distributes the compressed data only once; hence communication overhead is less compared with other DD algorithms. Thus the proposed method can prepare more accurate user spending profile with low minimum support in short time period. Experimental results show that proposed algorithms are capable to efficiently generate frequent patterns from very large sized dynamically growing distributed database with less communication overhead and good load balancing feature.

7. References

1. Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules. Proceedings of International Conference Very Large Data bases. 1994; 487–99.
2. Han J, Pei J, Yin Y. Mining frequent patterns without canidate generation. Proceedings of ACM SIGMOD International Conference on Management of Data. 2000; 1–12.
3. Park J, Chen M, Yu P. An effective hash-based algorithm for mining association rules. PYOC ACM-SIGMOD International Conference Management of Data. 1995; 175–86.
4. Cheung D, Ng T, Fu A, Fu Y. Efficient mining of association rules in distributed databases. IEEE Transactions on Knowledge and Data Engineering. 1996; 8(6):911–22.
5. Hidber C. Online association rule mining. Proceedings of the ACM SIGMOD International Conference on Management of Data. 1999; 145–56.
6. Leung C, Khan Q, Quamrul I, Li Z, Hoque T. CanTree: A canonical-order tree for incremental frequent-pattern mining. Knowledge and Information Systems. 2007; 11(3):287–311.
7. Yang C, Yang D. IMBT-a binary tree for efficient support counting of incremental data mining. International Conference on Computational Science and Engineering; IEEE Computer Society. 2009; 324–9.
8. Sherly K, Nedunchezhian R, Rajalakshmi M. IAPI Quad-Filter: An interactive and adaptive partitioned approach for incremental frequent pattern mining. Journal of Theoretical and Applied Information Technology. 2014; 63(1):147–57.
9. Park J, Chen M, Yu P. Efficient parallel data mining for association rules. Proceedings of International Conference Information and Knowledge Management. 1995.
10. Chen D, Lai C, Hu W, Chen W, Zhang W, Zhen W. Tree partition based parallel frequent pattern mining on shared memory systems. Proceedings of 20th International conference on Parallel and Distributed Processing Symposium. 2006.
11. Li N, Zeng L, He Q, Shi Z. Parallel implementation of apriori algorithm based on mapreduce. International Journal of Networked and Distributed Computing. 2013; 1(2):89–96.
12. Pramudiono I, Kitsuregawa M. Parallel FP-growth on PC cluster. Proceedings of the 7th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining. 2003; 467–73.
13. Bhadane C, Shah K, Vispute P. An efficient parallel approach for frequent itemset mining of incremental data. International Journal of Scientific and Engineering Research. 2012; 3(2):1–5.
14. Xu J, Sung AH, Liu O. Behavior mining for fraud detection. Journal of Research and Practice in Information Technology. 2007; 39(1).
15. Ghosh S, Reilly DL. Credit card fraud detection with a Neural-Network. Proceedings of International Conference on System Science. 1994; 621–30.
16. Syeda M, Zhang YQ, Pan Y. Parallel granular neural networks for fast credit card fraud detection. Proceedings of IEEE International Conference on Fuzzy Systems. 2002; 572 –7.
17. Aleskerov, Freisleben B, Rao B. CARDWATCH: A Neural Network based database mining system for credit card fraud detection. Proceedings of IEEE/IAFE Conference on Computational Intelligence for Financial Engineering (CIFEr). 1997; 220–6.
18. Chiu C, Tsai C. A web services based collaborative scheme for credit card fraud detection. Proceedings of IEEE International Conference on e-Technology, e-Commerce and e-Service. 2004; 177–81.
19. Stolfo SJ, Fan DW, Lee W, Prodronidis AL, Chan PK. Credit card fraud detection using meta-learning: issues and initial results. Proceedings of AAAI Workshop AI Methods in Fraud and Risk Management. 1997; 83–90.
20. Fan W, Wang H, Philip S. YuSalvatore J. Stolfo. A fully distributed framework for cost-sensitive data mining. Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02). 2002.
21. Chun Wei Clifton Phua. Investigative data mining in fraud detection. A thesis submitted. 2003; 1–126.
22. Kundu A, Sural S. BLAST-SSAHA hybridization for credit card fraud detection. IEEE Transactions on Dependable and secure Computing. 2009; 6(4):309–15.
23. Srivastava A, Majumdar AK. Credit card fraud detection using Hidden Markov Model. IEEE Transactions on Dependable and Secure Computing. 2008; 5(1):37–48.
24. Renuga Devi T, Rabiyathul Basariya A, Kamaladevi M. Fraud detection in card not present transactions based on behavioral pattern. Journal of Theoretical and Applied Information Technology. 2014; 61(3):447–55.