Behaviour Mining for Fraud Detection

Jianyun Xu

Microsoft Corporation; One Microsoft Way; Redmond, WA 98052; USA
dennisxu@microsoft.com

Andrew H. Sung and Qingzhong Liu

Department of Computer Science; New Mexico Tech; Socorro, NM 87801; USA
Institute for Complex Additive Systems Analysis; New Mexico Tech; Socorro, NM 87801; USA
{sung, liu}@cs.nmt.edu

Despite significant efforts by merchants, card issuers and law enforcement to curb fraud, online fraud continues to plague electronic commerce web sites. More advanced solutions are desired to protect merchants from the constantly evolving problem caused by fraud. The supervised machine learning technique for the most well known fraud detection algorithms makes them inadequate for an online system, which usually contains a mammoth size of non-stationery data. This paper describes a method to dynamically create user profile for the purpose of fraud detection. We use a data mining algorithm to adaptively profile legitimate customer behaviour in a form of association rule set from a transaction database. Then the incoming transactions are compared against the user profile to indicate the anomalies. A novel pattern match approach is proposed to evaluate how unusual the new transactions are. An empirical evaluation shows that we can accurately differentiate the anomaly behaviour from profiled user behaviour.

Keywords: Fraud detection, adaptive profiling, behaviour mining, anomaly detection

ACM Classification: K.6.5 (Security and Protection); H.2.8 Database Applications (Data Mining)

1. INTRODUCTION

It was shown by Kerr (2002) that internet transaction fraud is 12 times higher than in-store fraud. And online fraud rates have held steady since 2000, despite significant efforts by merchants, card issuers and law enforcement to curb fraud. All web merchants are liable to any of the various types of online transaction fraud, regardless of size, transaction volume or internet expertise. Hackers and fraudsters are becoming more sophisticated and skillful at manipulating internet protocol, web languages and tools to discover any weakness that they can exploit. Thousands of web merchants experience suspicious transaction activities and other types of account abuse each day. Therefore, an efficient online transact fraud detection system is badly needed by merchants to protect the legitimate customer, to lower the maintenance costs and to increase public confidence and trust.

Most previous work for fraud detection or anomaly detection was classification based. Some well known algorithms utilized for fraud detection are NB (naïve Bayesian), proposed by Elkan *et al* (1997), C4.5, by Quinlan (1998) and BP (Back-propagation). The researches of Domingos (1995), Elkan (1997; 2000) and Witten (1999) show that NB algorithm is very effective in many

Manuscript received: 4 October 2005
Communicating Editor: Rei Safavi-Naini

real world data sets and is extremely efficient in that it learns in a linear fashion. However when attributes are redundant and not normally distributed, the predictive accuracy is reduced. C4.5 can output not only accurate predictions but also explain the patterns, decision tree and rule set, in it. However, scalability and efficiency problems, such as the substantial decrease in performance, can occur when C4.5 is applied to large data sets. Back-propagation neural networks can process a very large number of instances, and have a high tolerance to noisy data. However, Agrawal (1995) showed that the BP algorithm requires long training times and extensive testing and retraining of parameters. The common disadvantage of all these algorithms is that they rely on supervised training, which requires human involvement to prepare training cases, and test cases to optimize parameters. Since most online systems contain non-stationery data due to the changing of the individual users' behaviours. A fraud detection system must be able to adjust its detector to keep up with the change of the user behaviour in order to indicate fraud transactions from legal transactions more accurately. The supervised learning techniques described above are inadequate to detect fraudulent transactions in an online system.

To solve this problem, a novel fraud detection framework is proposed in this paper. Individual user's behaviour pattern is dynamically profiled from the transactions by using a set of association rules. The association rule is first introduced by Agrawal *et al* (1993). The incoming transactions for that user are then compared against the profile in order to discover the anomalies, based on which the corresponding warnings are outputted. Our algorithm is evaluated on both synthetic data and real data. Experimental results show that our algorithm can accurately differentiate the anomaly behaviour from profiled user behaviour.

The paper is organized as follows. The next section presents the basic idea of our fraud detection framework. In Section 3 we introduce the architecture and the sub-modules of our framework and the corresponding algorithms in detail, including definitions and theoretical background of our approach. An empirical evaluation is shown in Section 4. The performance of our system is compared among different algorithms by different measuring techniques. And finally, in Section 5 we present our conclusions and propose our idea for future work.

2. BASIC IDEA

In this section, we will describe the basic idea of our fraud detection algorithm. Before doing so, we will first give some definitions.

Definition 2.1. A set of ***attribute-value pairs*** or ***items*** $\Sigma = \{a_i\ (v_j)\}$, where $i \in \{1, \ldots, n\}$, n is the number of all possible attributes in the database we want to keep record, $j \in \{1, \ldots, m(i)\}$, $m(i)$ is the possible values of attribute a_i, $m(i)$ is depended on the granularity specified along attribute a_i. We also use I_i to represent an attribute-value pair or item $a_i\ (v_j)$ for simplicity. An example of attribute-value pair is Price("*\$1-\$10*"), where Price is an attribute, "*\$1-\$10*" is a value of this attribute. The possible values are depended on the granularity or interval of the attribute Price. The interval is \$10 in this example. Another example of an attribute-value pair is Time("*Evening*"), where Time is an attribute, "*Evening*" is a possible value. By different granularity the attribute-value pair could be Time("*9pm*"). The different granularity could cause large differences on the performance of behaviour profiling.

Definition 2.2. The ***transactions*** are records of the form $T(t)$ where t is a value of the time variable D. Each transaction consists of a set of certain attribute-value pairs from Σ recorded in a period of t. A ***transaction database*** contains all the transactions.

The most recent transactions for an individual user in a transaction database are analyzed in order to profile the current behaviour or habit for that customer. The word 'recent' is spelled by a slide window, which could be a time window or a transaction count window. For example, recent transactions could be all the transactions in the past two months, or the recent 500 transactions. The customer's profile is utilized to monitor a new transaction of this user to indicate how unusual the new transaction is. At the same time, the customer's old profile is automatically updated by accumulating the occurrences of the new attributes, which represents the user's new behaviour. It is quite reasonable to assume that a normal user should have a behaviour pattern which indicates his or hers consuming interests or habits, since a totally randomly consuming behaviour is very uncommon.

We use a set of ***association rules*** to profile a user's recent behaviour.

Definition 2.3. ***Association rules*** are the implication of the form $X \rightarrow Y$, where

1. $X \subseteq \sum, Y \subseteq \sum$.
2. $X \bigcap Y = \varnothing$
3. $\exists I_i \in X, 1 \leq i \leq n$
4. $\exists I_j \in Y, 1 \leq j \leq n$

The association rule $X \rightarrow Y$ is interpreted as data set that satisfies the conditions in X are also likely to satisfy the conditions in Y. An example of an association rule is:

day("*Saturday*") ∧ time("*8pm-10pm*")→play("*Xbox contest*") [***support***=15%, ***confidence***=77%]. The rule indicates that for all transactions of a customer recorded in a time window, 15% (***support***) transactions are playing "*Xbox contest*" in "*Saturday*", "*8pm-10pm*". There is a 77% probability (***confidence***) that if a transaction happens in "*Saturday*", "*8pm-10pm*" it would be an Xbox contest.

Two important measures for association rules, ***support*** and ***confidence***, are defined as follows.

Definition 2.4. The ***support***, s, of an association rule is the ratio (in percent) of the transactions containing $X \cup Y$ to the total number of transactions analyzed, $|R(t)|$. If the ***support*** of an association rule is 15% then it means that 15% of the analyzed transactions contain $X \cup Y$. ***Support*** is the statistic significance of an association rule. The association rules have the ***supports*** less than 5% would be considered not very important to profile a user's behaviour. While a high ***support*** is often desirable for association rules.

Definition 2.5. For a given number of transactions, ***confidence***, c, is the ratio (in percent) of the number of transactions that contain $X \cup Y$ to the number of transactions that contain X. Thus if we say an association rule has a confidence of 77%, it means that 77% of the transactions containing X also contain Y. The ***confidence*** of a rule indicates the degree of correlation in the dataset between X and Y. It is used as a measure of a rule's strength. Often a large ***confidence*** is required for association rules.

An FP-tree (frequent pattern tree) structure and FP-tree growth algorithm, proposed by Han (2000) are utilized to uncover these hidden association rules from the recent transactions for this user. We improve the FP-tree growth algorithm to allow it to mine both ***intra-transaction associations*** and the ***inter-transaction association***. It will be discussed in Section 3 and Section 4.

Any new transaction of a user is compared against his or hers FP-tree to indicate the anomaly, which means how unusual the transaction is. A novel FP-tree based similarity measure algorithm is utilized to calculate the anomaly. We also use an accumulating algorithm to accumulate the low suspicious level warning to generate a high-confidence alarm. By comparing the alarm against a set of thresholds, corresponding fraud resolution could be performed.

3. NMT FRAUD DETECTION FRAMEWORK

We will describe the architecture of our experimental system, NMT (New Mexico Tech) fraud detection framework in this section. Our system consists of three major modules: Data engine, rule engine and rule monitor. They are shown inside a dashed rectangle in Figure 1. The objects surrounding the FDS (fraud detection system) make a typical online transaction system, which includes several web applications and web services to provide OLTP (OnLine Transaction Process), a database storing transaction data, and a database replication in order to provide minimum performance degradation on OLTP by backend data process or analysis.

FDS is a backend process, whose impact on the front end of the online system is minimized, since it only talks to the replication database. Data engine serves as an interface between the replication database and the FDS. It collects and pre-formats the recent transactions of all individual customers in the online system. The recent transactions depend on the detection sensitivity, which is specified by the customer through the web applications in OLTP. The rule engine module mines the recent transactions to generate a profile, an association rule set stored in an FP-tree, for each user. The FP-tree is updated adaptively after the new transactions of the user are added in replication database. The rule monitor module monitors the new transaction for every user. Any new transaction of a particular user is compared against the FP-tree for that user to indicate the anomaly. The anomaly is then mapped into a corresponding suspicious level, which is sent back to OLTP. The corresponding resolution is performed based on the suspicious level of a new transaction. In the following sub-sections of this paper, we will describe the implementation of rule engine and rule monitor in detail.

3.1 Adaptive Association Rule Mining

The major responsibility of rule engine is to adaptively generate association rule sets to profile user behaviours. There are a large amount of techniques to mine association rules from a transaction

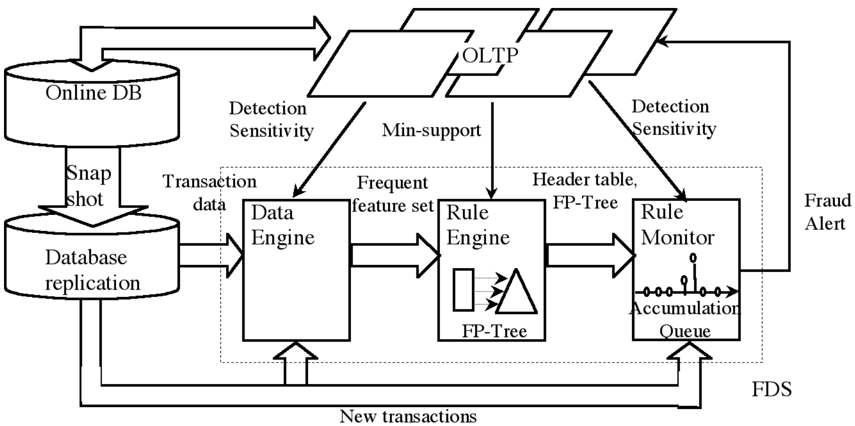


Figure 1: Architecture of NMT fraud detection system. The dashed rectangle is the fraud detection system. The objects surrounding it mask a typical online transaction system.

database. Agrawal *et al* (1994) introduced an Apriori-like mining method to mining association rules. An extension version of the Apriori algorithm also by Agrawal (1995) is able to mine generalized, multilevel, or quantitative association rules. Moe *et al* (1996) introduced association rule mining query languages. Ng *et al* (1998) introduced a constraint-based rule mining technique. Cheung *et al* (1996) presented an incremental updating technique to discover the association rules in a large scale database. Agrawal (1996) proposed another technique to perform parallel and distributed mining. Brin *et al* (1997) introduced a dynamic itemset counting technique to reduce the number of database scans. Ozden *et al* (1998) presented cyclic and interesting association rule mining. The most efficient association rule mining algorithm so far is the FP-tree growth algorithm proposed by Han *et al* (2000). It is able to mine the frequent patterns without candidate generation. Our fraud detection approach is largely based on this algorithm. The benefits of this method are highly condensed yet complete for frequent pattern mining, avoiding costly database scanning. More importantly, it allows us to adaptively generate user profile without preparing labeled data. Some popular rule generators, RL proposed by Clearwater (1993), C4.5 by Quinlan (1993) for example, are based on supervised learning, which is not capable for adaptive profiling and is unaffordable for online transaction detection, since training cases preparation requires human investigation and is time consuming.

TID	Transaction data	Frequent Itemsets
1	ET, ST, EV, 129.138, L50	ST, 129.138, EV, ET
2	ET, ST, MR, 202.55, L10	ST, ET, L10
3	ET, SU, MR, 129.138, L50	129.138, ET
4	BK, ST, EV, 129.138, L10	ST, 129.138, EV, L10
5	CL, ST, EV, 129.138, L10	ST, 129.138, EV, L10

Each transaction includes the attributes of purchased product, time (in weekday), time (in day part), grouped IP address and the grouped purchased amount. The abbreviations: ET-entertainment, BK-book, CL-clothes, ST-Saturday, SU-Sunday, MO-Monday, EV-evening, MR-morning, L#-less than # of dollars.

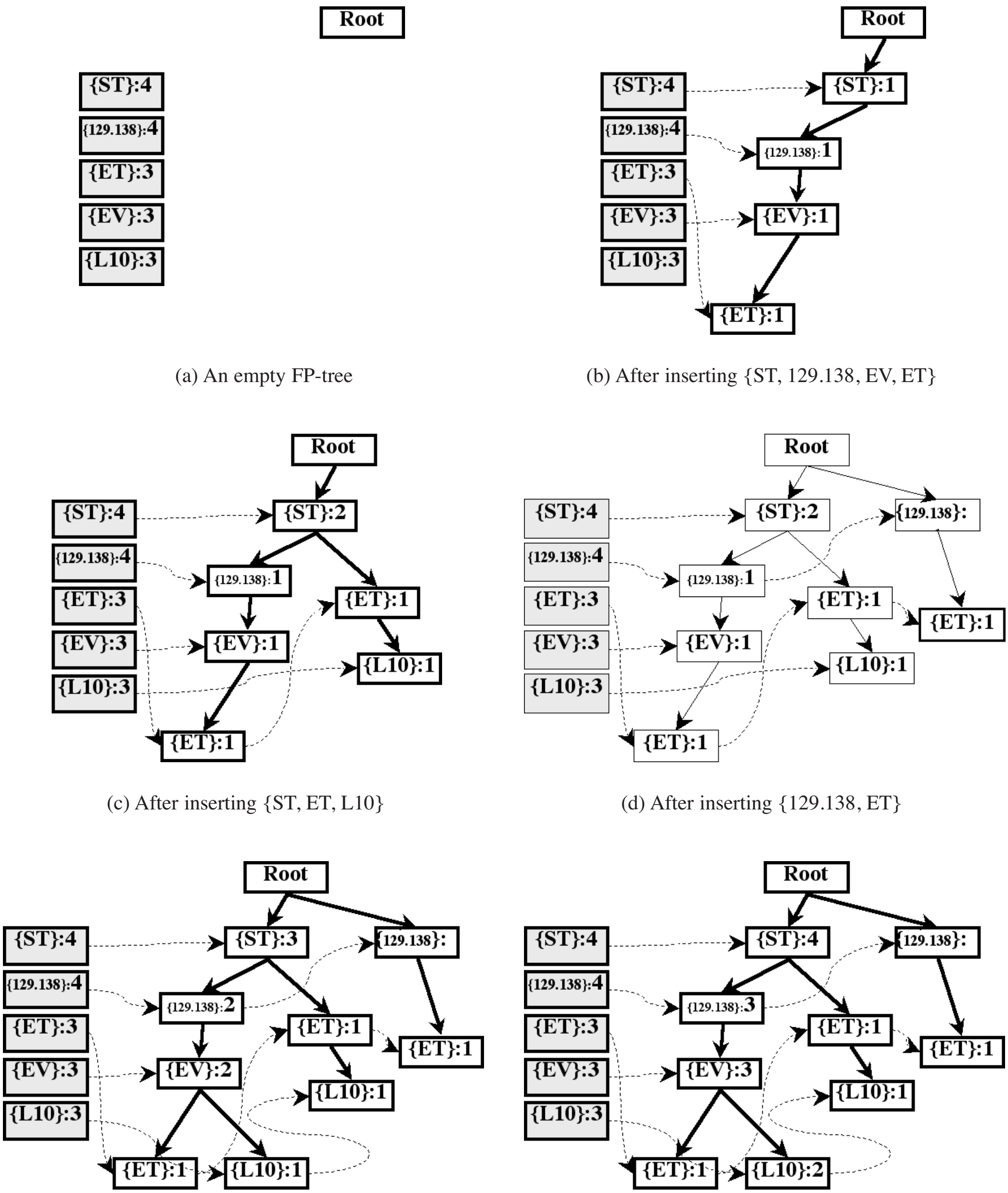
Table 1: An example of recent transactions for a customer. This table shows five transactions for a typical online transaction system. The right column lists the corresponding frequent itemsets. Minimum support value in this example is 60%. Minimum support is a user specified threshold for mining association rule from a database.

Table 1 shows an example of five transactions. Each transaction is an itemset, which contains five items or attribute-value pairs.

Definition 3.1. A set of items or attribute-value pairs is referred to as an ***itemset***. The occurrence ***frequency*** of an itemset is the number of transactions that contain the itemset. This is also known as ***support count***. The frequency of itemset {ST} is 4 in this example.

Definition 3.2. If the occurrence frequency of an itemset is greater than or equal to the product of *min_sup* and the total number of transactions, then it is a ***frequent itemsets***. The value of *min_sup* is called ***minimum support*** value. In the example shown in Table 1, the corresponding frequent itemset for transaction 2 is {ST, ET, L10}, since items MR, 202.55 are less than the minimum support value, 3 in this example (*min_sup* = 60%). Since the rarely occurred items would be filtered out when we generate the frequent itemsets, the rarely occurred behaviour such as fraudulent actions will be filtered out. That is the reason we could profile a user's behaviour without the preparation of the labeled data.

To profile the behaviour or a user, an FP-tree structure, introduced by Han (2000), is used to store compressed, crucial information about frequent pattern, from which association rules are generated. FP-tree is a combination of a general prefix tree and a linked list table. Figure 2 shows an example of creating a small FP-tree structure from the transactions shown in Table 1. The FP-



(a) An empty FP-tree

(b) After inserting {ST, 129.138, EV, ET}

(c) After inserting {ST, ET, L10}

(d) After inserting {129.138, ET}

(e) After inserting 1st {ST, 129.138, EV, L10}

(f) Insert 2nd {ST, 129.138, EV, L10}

Figure 2: An example of an FP-tree construction

tree stores quantitative information about frequent pattern. The tree nodes are arranged in such a way that more frequently occurring nodes will have a better chance of sharing nodes than lesser ones. From every tree node to the root, the path is a frequent pattern, from which association rules could be generated.

FP-tree is constructed by accumulating the occurrences of the attributes in recent transactions of an individual customer. By mining an FP-tree, we can find all the conditional rules that correlate the presence of one set of features with that of another set of features. However, we are not interested in extracting these association rules. We use a pattern matching algorithm to compare a new transaction directly with the FP-tree to indicate the new transaction's anomaly. It will significantly improve the overall system performance. The most attractive feature of the FP-tree is that it could be updated by accumulating the occurrences of the attributes without any human involvement. It is an ideal data structure to profile and monitor a non-stationary data set.

To create an FP-tree, recent transactions should be scanned twice. The first scan of the transactions derives the set of frequent items. For example, the frequent itemset of the last transaction is {CL, ST, EV, 129.138, L10}. The frequent items are then sorted in the order of a descending support count. The resulting set or list is denoted L, which is {ST, 129.138, EV, L10} in this example L is utilized to construct a header table shown in the left side of Figure 2(a).

An FP-tree is then constructed from an empty root, shown in Figure 2(a). Branches of the FP-tree are then inserted into the tree by scanning the transactions a second time. The items in each transaction are processed in L order, and a branch is created for each transaction. For example, the scan of the first transaction, "ET, ST, EV, 129.138, L50", which contains four items {ST, 129.138, EV, ET} in L order, shown in the first data row of Table 1, leads to the construction of the first branch of the tree with four tree nodes: {ST}:1,{129.138}:1, {EV}:1 and{ET}:1, where {ST}:1 is linked as a child of the root, {129.138}:1 is linked to {ST}:1…, shown in Figure 2(b). The number after the semicolon in the tree node shows the current occurrence of an item. To make tree traversal easy, an item header table is built so that each item pointer points to its occurrences in the tree via a chain of node links. The items in the header table are absolutely the same as the element in L.

The second transaction is processed in the same way as the first one. T_2, "ET, ST, MR, 202.55, L10" contains {ST}, {ET} and {L10} in L order. These items would result in a branch where {ST}:1 is linked to the root and {ET}:1 is linked to {ST}:1, {L10}:1 is linked to {ET}:1. However this branch would share a common prefix, {ST}, with the existing path for T_1. Therefore we should increase the count of the {ST} node by 1, and create two new tree nodes, {ET}:1 and {L10}:1, which are linked as a child tree of {ST}:2, shown in Figure 2(c). In general, when considering the branch to be added for a transaction, the count of each node along a common prefix is increased by 1, and nodes for the items following the prefix are created and linked accordingly. The repeated insertion for $T_3 – T_5$ are shown in Figures 2(d)–(f).

The size of an FP-tree is largely dependent on the minimum support, which decides the size of the frequent itemset for a transaction after filtering. The FP-tree is constructed from these itemsets. Therefore, the smaller minimum support value leads to a larger FP-tree, which profiles a user's behaviour more accurately, since it stores some associations having relatively small support counts. However it requires more time to construct. The tradeoff between accurate profiling and time for tree construction is shown in Figure 3.

We can get all association rules under the condition of a particular frequency item by following its node link chain in header table. For example, to get all rules for {L10} we should firstly search the header table to find the node link chain of {L10}. The first node in the link chain is {L10}:2, whose prefix path is ({ST}:4, {129.138}:3, {EV}:3). The corresponding association is

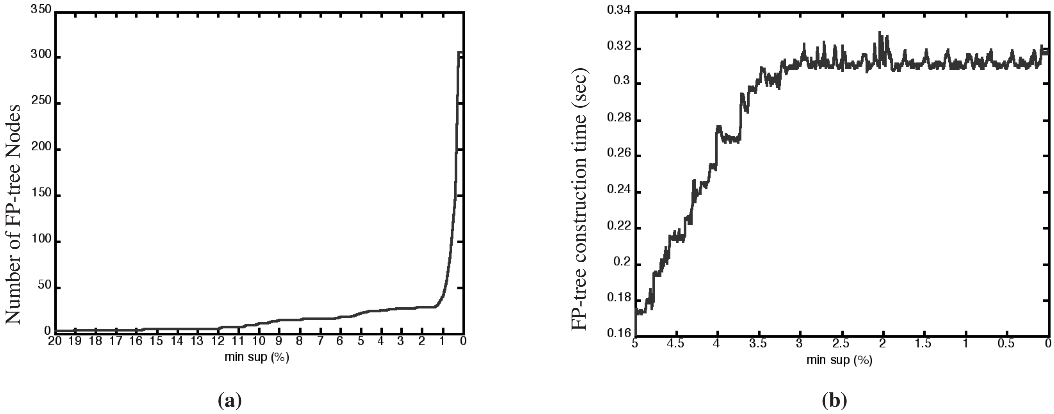


Figure 3: Effect of decreasing minimum support value on a FP-tree. (a) FP-tree nodes increase when minimum support value decreases. (b) FP-tree construction time increases when minimum support value decreases. The experiment is performed on the synthetic data. 10K transactions are used in this experiment.

({L10})→({ST}, {129.138}, {EV}) [40%, 67%], where 40% is the support, 67% is the confidence. The support value is calculated by dividing current nodes' occurrence count, 2 in this example, by total transaction count, 5 in this example. It indicates how important this rule is to profile the user behaviour. We calculate the confidence by dividing current frequency node's occurrence count, 2 in this example, by the node's total occurrences, 3 in this example. It shows how strong this rule is. By following current node's link chain, we can get the second tree node {L10}:1. The corresponding association is {L10}→({ST},{ET}) [20%,33%], which is generated in the same way. This process continues until the null link chain pointer is met.

3.2 FP-Tree Based Pattern Matching Algorithm

Two techniques are utilized to build the rule monitor. To indicate the anomaly of a new transaction, we designed a novel FP-tree based pattern matching algorithm. And an alert accumulating algorithm is used to lower the false alarm and to detect a set of fraudulent transactions with low suspicious values.

For each frequent item t_i in an FP-Tree, we should find all the prefix paths of t_i, which is the sub patterns base under the condition of t_i's existence. These prefix paths do not include all the possible patterns containing t_i. Since, some patterns are filtered by minimum support. An incoming transaction having more than one item in the header table means that this transaction matches the single frequent pattern in some level. To discover in what extent the new transaction matches the association rules, we need to walk through the FP-tree by following the header table link chains to compare that transaction to every sub pattern base.

double SimMatch(T) {
  sim = 0.0;
  for each item t_i in T {
    if (found *headtablelink*, in the head table) {
      sim_credit = 0.0;
        for each tree node in N_i^j in *headtablelink*,

```
        if (P^j(t_i) ⊆ T )
            sim_credit += G(N_i^j .s, N_i^j .c) × weight(t_i);
      sim += sim_credit;
    }
  }
  return sim;
}
```

The above pseudo code shows the pattern matching algorithm for calculating the similarity between a new transaction and the user's FP-Tree. Suppose $T = \{t_1, t_2, \ldots, t_n\}$ is an incoming transaction. For each frequent item t we calculate a similarity credit $sim\_credit(t_i)$.by the following steps: Search t_i in header table. If not find, $sim\_credit(t_i) = 0$. Otherwise, follow the link chain to the first tree node N_i^1 containing t_i. A conditional frequent pattern (set of frequent items) can be obtained by following N_i^1's parent link until it reaches the root. Let's denote the attributes set in this pattern as $P^1(t_i)$. If $P^1(t_i) \subseteq T$, $sim\_credit(t_i)$ is increased by a credit function, $G(s, c)$, based on the support and the confidence of N_i^1. In our implementation, we used an entropy like function: $G(s, c) = -s \times \log_2(1+\varepsilon-c)$, where s is the support, c is the confidence, $s \leq c \leq 1.0$, ε is a real number used to specify the upper boundary of function G. The function is chosen by the intuition that if a new transaction matches with a rule having larger probability it is more likely to match the user's behaviour, and the confidence value is emphasized. Discovering the best $G(s, c)$ itself would be a good research topic.

Since different kinds of frequent items are of different importance to profile a user or a system. For example, a matched IP pattern could be more important than a matched time pattern. A weight function, weight(t_i), was used to give various stress to the different item types. So we increase $sim\_credit(t_i)$ by $G(s, c) \times \text{weight}(t_i)$ instead of $G(s, c)$. The weight function could be a fixed look up table, which maps the different item types to different weights. It is also possible to use a neural network to train real data to get the optimized look up table.

After N_i^1 is processed, we follow the link chain of N_i^1 to reach the second tree node, N_i^2, containing t_i. For that tree node we do the same thing as to N_i^1.The process is stopped when the N_i^j's link chain pointer is null. The similarity value for T is computed as: $sim(T) = \sum_{i=1}^{n} sim\_credit(t_i)$, where $sim\_credit(t_i) = weight(t_i) \times \sum_j G(N_i^j.s, N_i^j.c)$, j is the index of the node containing t_i found by following the head table link of t_i.

The $sim(T)$ would be mapped into a corresponding suspicious value. Similarity value represents the extent that a new transaction is comparable to the customer behaviour patterns. The minimum similarity value is zero, which means not a single rule is matched between the new transaction and the user pattern. It indicates the highest suspicious level. By contrast, the larger similarity value means a smaller suspicious level.

3.3 Alert Accumulating Algorithm

We can setup a set of thresholds to fire corresponding fraud warnings. Technically, it is possible to mis-classify some legal transactions, which do not follow the customer's normal behaviour. Since the frequent items are filtered by min support before creating the FP-tree. Therefore the tree is not completed, the very unusual patterns are not collected in the tree. Moreover, a customer could also suddenly change his or her behaviour. Another important issue is that in order to minimize the fraud detection cost, the purchased amount is a factor of firing alarm. For a very small purchase amount, for example .50$, even it is highly suspicious, fire an alarm is not economical. Since the objective

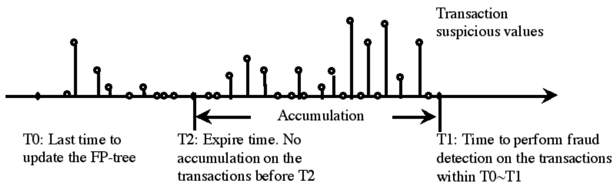


Figure 4: Suspicious values within accumulation window.
The height of a point indicates the suspicious value of this transaction. T0 is the moment that the FP-tree is most recently updated. T1 is the current time. T1 and T2 is the boundary of the accumulation window.

of fraud detection action is to minimize the total cost. The cost model section explains this issue in detail. By using a suspicious threshold for a single transaction, a sequence of fraud transactions with low purchasing amounts could be missed.

To solve these problems, we use a novel technique to accumulate the warnings from a set of new transactions.

We calculate the alert values for a set of transactions by accumulating their suspicious values. The transactions to be processed could include all the new transactions after last FP-tree updating, or we can use an expiring time window to specify the transactions we would like to accumulate. Then by comparing the alert sum instead of the single alert against a set of thresholds, a corresponding fraud alarm would be fired. The threshold set is decided by the detection sensitivity specified by the user.

We accumulate all transactions within the specified time window (T2~T1), as shown in Figure 4, by adding them together with the same emphasis. A simple step function is the most straightforward expiring function. Some examples of the expiring functions are shown in Figure 5. For a step function, if time of a transaction, t, is larger than T2, $f(t)$ equals to 1, else $f(t)$ equals to 0. We also can emphasize the transactions that happen more recently by using either the nature logarithm function or the polynomial functions.

The fraud alert value is calculated by: $AlertValue = \sum_{i=1}^{n} (s(T_i) \times f(T_i) \times amount(T_i))$, where T_i is a transaction in the accumulation window, $s(T_i)$ is the suspicious value of T_i, $f(T_i)$ is the expiring weight of T_i specified by expiring function, $amount(T_i)$ is the consumed money of transaction T_i. It is reasonable that several highly suspicious transactions having very low consumed points will not fire an alert, by contrast, only one highly suspicious transaction having very high consumed points could fire an alert.

3.4 Fraud resolutions

The output of the fraud detection system is an alert value indicating the suspicious level of analyzed transactions. This value is sent back to a web application in the OLTP. By comparing the alert value

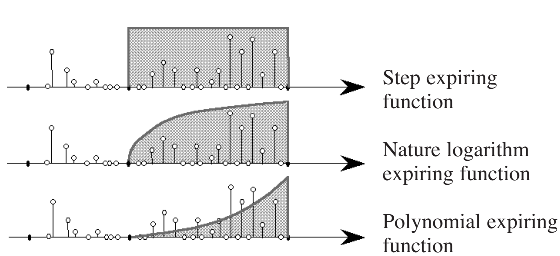


Figure 5: Some examples of expiring functions

to a set of thresholds, we are able to know the corresponding reaction to be performed. For example, if the alert value for a transaction or a set of transactions is below .50, no warning is given. If the alert value is in a range of .50 to .85, an email is sent to the user in order to give a fraud warning. If the alert value is higher than .85, we might temporally lock the account to prevent further fraud or loss. And the user should also be informed by an email or a message to verify the transaction and unlock his or her account. If the user confirms this is a fraud transaction, further protection should be performed, for example changing the password. If the user confirms this is a legal transaction, we should unlock his account and the transaction would be added to the recent transaction set, from which the customer's FP-tree would be updated. Therefore the user profile, FP-tree, could be adaptively changed by keeping up with the changing of the user's behaviour.

4. PERFORMANCE STUDY

4.1 Data preparation

The real commercial database for performance evaluation is far more than possible. The only available real time data for our experiments is a set of American Express Credit Card transactions of an individual user within three and a half years. It is inadequate to measure the system performance since the real time behaviours could be various from user to user. To evaluate the system's effectiveness on different kinds of user behaviours, we created a tool to simulate different kinds of user behaviours. We then tested and evaluated NMT-FDS and other well known algorithms (Naïve Bayes, C4.5, BP, SVM) on the same data sets.

Diversity real time user behaviour requires a powerful and flexible simulator. To generate the various user patterns, profile driven is an effective solution, since the profile could be creatively designed in order to represent complex patterns. Our simulator is similar to the one proposed by Chung (1995), whose functions are parsing the user profile and modeling the user behaviour by following the rules defined in the profile file.

The simulated data has the very similar formation as the American Express data. Table 1 is an example of the simplified record coming from the simulated data sets. Two different types of user profiles are used in our experiments to simulate the relatively regular behaviours and the relatively irregular behaviours. An example of regular behaviour profile used in experiments is: most of the transactions come from a small IP group, most transactions take place at weekend, transactions are purchasing a group of products, the transactions likely occur in the evening. An example of irregular behaviour is: transactions come from dynamic IPs, most transactions take place at weekend and transactions are purchasing a group of products.

For both of the profiles, we create 3000 legal transactions and 50 fraudulent transactions as a labeled training set to train the supervised classification algorithms. Our approach takes all 3050 transactions as an unlabeled data set, that is to say, we don't know the fraudulent and legitimate when we create the user profile. The trick is that the fraudulent behaviour with very low occurrence ratio will be filtered out when we create the FP-tree. We also create 3000 legal transactions and 20 fraudulent transactions as testing data. The fraudulent transactions are the transactions representing totally random behaviours.

4.2 Experiment results

Figure 6 on the following page shows the performance comparison among BP, NB, SVM and NMT-FDS on both real time and simulated data. We use the ROC (Receiver Operating Characteristics) curve to evaluate these classification algorithms. The ROC of a classifier shows its performance as a trade off between selectivity and sensitivity. The area under the ROC is a convenient way of

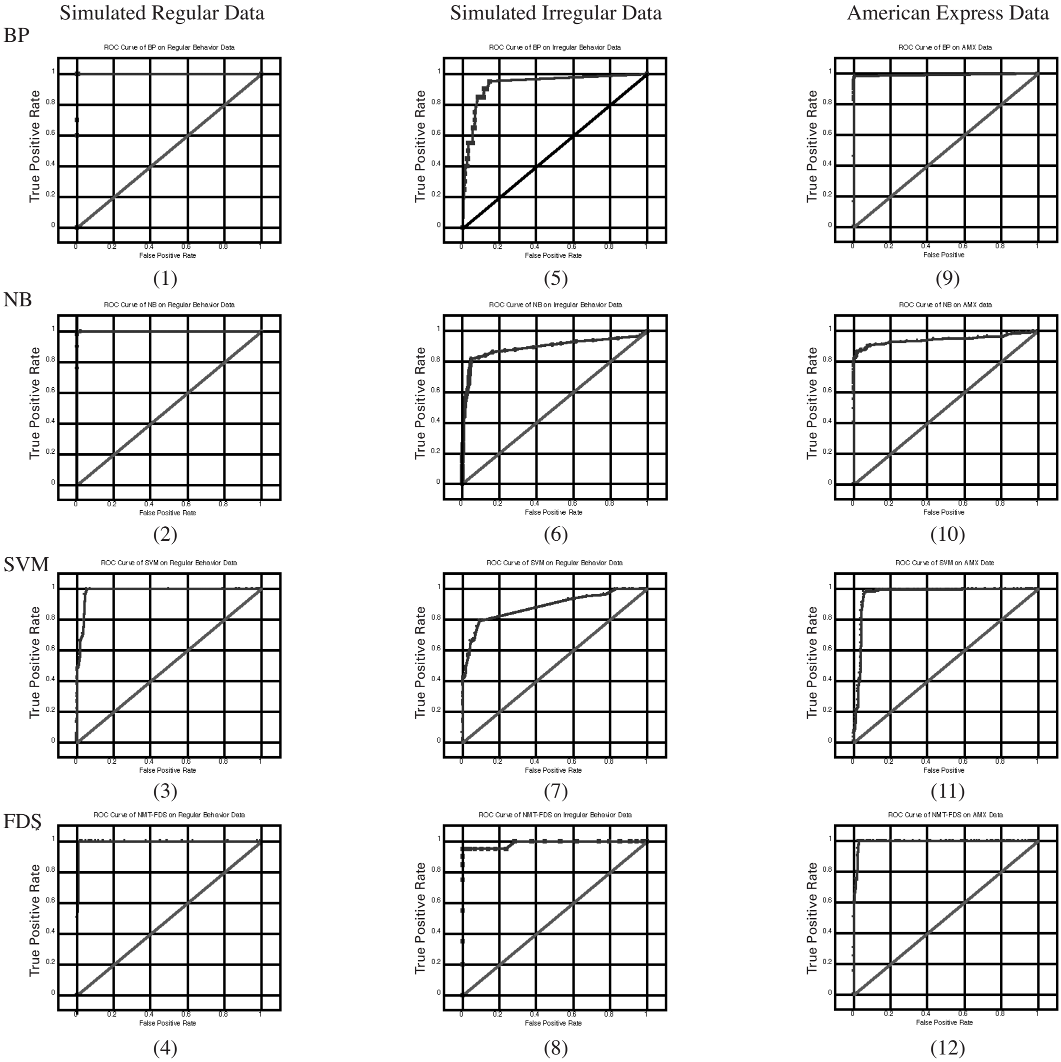


Figure 6: ROC curve comparison among the different algorithms.
(1)–(4) show the ROC curves of different four algorithms on relatively regular data. (5)–(8) show the ROC curves on relatively irregular data. (9)–(12) show the ROC curves on a data set of a single American Express user.

comparing classifiers. A random classifier has an area of 0.5, while an ideal one has an area of 1. Figures 6(1)–6(4) show the ROC curves of four algorithms on relatively regular data. The areas under the ROC are very close to 1, which means they work well to differentiate fraudulent data against regular behaviour data. Figures 6(5)–6(8) show the ROC curves on a test data set of the irregular behaviours. Figure 6(9)–6(12) show the ROC curves on a data set of an American Express user. Experimental results show that our algorithms, NMT-FDS works much better on irregular data than other algorithms. The reason is that dynamic IPs in irregular behaviour data set introduces noises which largely hamper accurate learning of NB, SVM and BP. Since noises or rarely attributes

Prediction	Fraud	Legal
Alert	*Hit*	*False Alarm*
No alert	*Miss*	*Normal*

Table 2: Outcome matrix for fraud detection

such as dynamic IPs would be automatically filtered by the min-support, FSD is capable to work on the irregular data. A tree based classification algorithm, C4.5, has also been studied. It has a very similar performance to NMT-FDS. However the supervised learning makes C4.5 inadequate for adaptively profile updating.

ROC curve is a very convenient way of comparing the performance of classifiers, however it cannot show the misclassification cost. To measure the benefit of detecting fraud, we used a Cost Model introduced by Chan *et al* (1998), a more realistic model, to accompany the different classification outcomes. A cost model formulates the total expected cost of fraud detection. It considers the trade off among all relevant cost factors and provides the basis for making appropriate cost sensitive detections. The detection outcome is one of the following: *hit, false alarms, miss*, and *normal*. They are outlined in Table 2.

Table 3 shows the cost for corresponding outcome. We use a simplified cost model to evaluate different algorithms. In our measurements, we assume fraud handling cost, *fh*, detection cost, *dt* and false alarm cost, *fp* are fixed numbers. And *tr* is the amount of money related to a transaction lost due to the mis-detection. The real time cost model could be much more sophisticated.

The evaluation for the predictive models to find the optimum cost savings are:

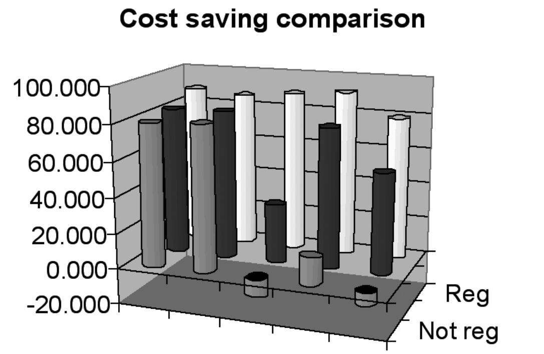
Model Cost Savings = *No Action* – [*Hits Cost* + *False Alarms Cost* + *Misses Cost* + *Normal Cost*], where *No Action* is the total lost caused by fraud of a system without fraud detection, the costs of the different outcomes are listed in Table 3.

Percentage Savings = *Model Cost Savings* / *Best Case Scenario Cost Savings* × 100, where the *Best Case Scenario Cost Savings* is the total cost of an ideal fraud detection system, whose *Miss Cost* and *False Alarm Cost* equal to zero.

Figure 7 shows the cost saving comparison among NMT-FDS, C4.5, NB, BP and SVM. All algorithms are effective (*percentage saving*≈80%) to detect fraud against simulated regular

Outcome	Cost
Hits	$\lvert hits\rvert \times Cost(dt) + \sum_{i\in hits} Cost(fh_i)$
False Alarms	$\lvert falsealarms\rvert \times Cost(dt) + \sum_{i\in falsealarms} [Cost(fh_i) + Cost(fp_i)]$
Misses	$\lvert misses\rvert \times Cost(dt) + \sum_{i\in misses} Cost(tr_i)$
Normal	$\lvert normal\rvert \times Cost(dt)$

Table 3: Cost for corresponding outcome.
***fh*: fraud handling and investigation cost, *dt*: detection cost, *fp*: false penalty including the inconvenience brought to user. *tr*: the amount of money of a transaction lost due to miss detection. It is easy to find the false alarm has the largest penalty. An ideal fraud detection system should reduce the false alarms and the misses as much as possible.**



	NMT-FDS	C4.5	NB	BP	SVM
■ Not reg	80.203	81.726	-9.645	15.736	-7.1066
■ AMX	82.016	82.768	32.869	78.130	55.983
□ Reg	88.832	86.802	89.340	91.878	79.1878

Figure 7: Percentage savings comparison among algorithms

behaviour data. However, NB, BP and SVM are not good to detect the American Express data set and simulated irregular behaviour data. In the latter case, by using NB or SVM, we even got a negative saving, which means the system lost even more money by performing fraud detection with these algorithms.

5. CONCLUSIONS AND FUTURE WORK

We have presented a framework for detecting fraudulent transactions in an online system. We describe the major modules of the framework and the related algorithms in detail. A prototype of the fraud detection system has been built to evaluate our algorithm. A profile driven simulator is designed to generate transaction data representing various behaviour patterns in order to evaluate the performance of our algorithm. Comparisons are performed among NMT-FDS, C4.5, NB, BP and SVM. Table 4 shows a summary of the qualitative comparison. Our system generates fraud alarm accurately and efficiently on both the simulated and real data. Unsupervised training and self adjustment to changing user behaviour make our system effective for monitoring online transaction systems and provide fraud detection and protection.

In the future, we will extend our system to detect system level fraud by utilizing an approximate weighted tree matching algorithm. Inter-transaction behaviour mining is also planned to enhance the performance of individual user profiling. Our future work also includes optimizing pattern matching algorithm, optimizing weight selection for different types of rules by real data training and expanding our algorithm to real-time fraud prevention.

Algorithms	Effectiveness	Scalability	Speed	Training data	Adaptability
NB	Various	Good	Good	Labeled	Poor
C4.5	Good	Poor	Fine	Labeled	Poor
BP	Fine	Good	Poor	Labeled	Poor
SVM	Fine	Good	Poor	Labeled	Poor
NMT-FDS	Good	Good	Fine	Unlabeled	Good

Table 4: Summary of qualitative comparison of algorithms *Effectiveness* highlights the overall predictive accuracy and performance of each algorithm. Scalability refers to the capability to construct a model effectively given large data sets. Speed indicates the efficiency in model construction. Training data shows the training data in model construction. Adaptability refers the capability and efficiency to adjust the model to follow the changes of the user behaviours.

ACKNOWLEDGEMENTS

Partial support for this research was received from an NSF SFS grant, a DoD IASP grant, and ICASA (Institute for Complex Additive Systems Analysis, a division of New Mexico Tech) is gratefully acknowledged.

REFERENCES

AGRAWAL, R. and SRIKANT, R. (1993): Fast algorithms for mining association rules. In *Proc. of the 20th Intl. Conf. on Very Large Data Bases*, 478–499. Santiago, Chile.

AGRAWAL, R. and SRIKANT, R. (1995): Mining sequential patterns. In *Proc. of the International Conference on Data Engineering*, 3–14. Taipei, Taiwan.

AGRAWAL, R. and SHAFER, J. (1996): Parallel mining of association rules. *IEEE Transactions on Knowledge and Data Engineering* 8 (6): 962–969.

BRIN, S., MOTWANI, R. and SILVERSTEIN, C. (1997): Beyond market basket: generalizing association rules to correlations. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, 265–276. Tucson, Arizona, USA.

CHAN, P. and STOLFO, S. (1998): Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. *Proc. of the Fourth International Conference on Knowledge Discovery and Data Mining*, 164–168.

CHEUNG, D., HAN, J., NG, V. and WANG, C. (1996): Maintenance of discovered association rules in large databases: an incremental updating technique. In *Proc. of the Intl. Conf. on Data Engineering*, 106–114. New Orleans, Louisiana, USA.

CLEARWATER, S. and PROVOST, F. (1993): RL4: A tool for knowledge-based induction. In *Proc. of the Second International IEEE Conference on Tools for Artificial Intelligence*, 24–30.

CHUNG, M., PUKETZA, N.J., OLSSON, R.A. and MUKHERJEE, B. (1995): Simulating concurrent intrusions for testing intrusion detection systems: parallelizing intrusions. *Proc. of the 1995 National Information Systems Security Conference*, 173–183.

DOMINGOS, P. and PAZZANI, M. (1996): Beyond independence: conditions for the optimality of the simple Bayesian classifer, in *Proc. of the 13th Conference on Machine Learning*, 105–112, Bari, Italy.

ELKAN, C. (2000): Magical thinking in data mining: lessons from CoIL challenge 2000, Department of Computer Science and Engineering, University of California, San Diego, USA.

ELKAN, C. (1997): Naïve Bayesian Learning. *Technical Report CS97–557*, Department of Computer Science and Engineering, University of California, San Diego, USA.

HAN, J. and KAMBER, M. (2000): *Data Mining: Concepts and Technique*, Morgan Kaufmann; 1st edition.

HAN, J., PEI, J. and YIN, Y. (2000): Mining frequent patterns without candidate generation. SIGMOD'00, 1–12, Dallas, TX, USA.

KERR, K. and LITAN, A. (2002): Online transaction fraud and prevention get more sophisticated, Gartner. http://www.gartnerg2.com/research/rpt-0102-0013.asp

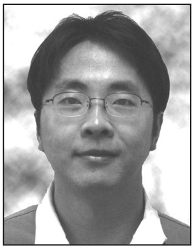
MEO, R., PSAILA, G. and CERI, S. (1996): A new SQL-like operator for mining association rules. In *Proc. of the 22nd Intl. Conf. on Very Large Data Bases*, 122–133. Mumbai, India.

NG, R., LAKSHMANAN, L., HAN, J. and PANG, A. (1998): Exploratory mining and pruning optimizations of constrained association rules. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, 13–24. Seattle, Washington, USA.

OZDEN, B., RAMASWAMY, A. and SILBERCHATZ, A. (1998): Cyclic association rules. In *Proc. of the Intl. Conf. on Data Engineering*, 412–421.

QUINLAN, J. R. (1993): C4.5: *Program for machine learning*. Morgan Kaufmann, San Mateo, CA, USA.
WITTEN, I. and FRANK, E. (1999): *Data Mining: Practical Machine Learning Tools and Techniques with Java*, Morgan Kauffman Publishers, California, USA.

BIOGRAPHICAL NOTES


Jianyun Xu

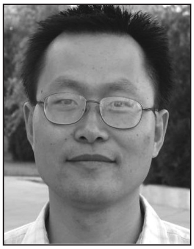
Jianyun Xu received his B.Sc. in computer science and engineering in 1999 from Zhejiang University, Hangzhou, P.R.C. and M.Sc. and Ph.D. in computer science in 2005 from New Mexico Tech, Socorro, U.S.A.

In January 2006, he joined the Identity Service Group, Microsoft Corporation, Redmond, USA, where he is currently a software design engineer working on network security.

His research interests include fraud detection, mal-ware detection, network security, steganography, steganalysis, watermarking, imaging processing and computer graphics.


Andrew H. Sung

Andrew H. Sung is a professor and the chairman of the Computer Science Department of New Mexico Tech. He received his Ph.D. in 1984 from the State University of New York at Stony Brook. His current research interests include information security, application algorithms, and soft computing and its engineering applications.


Qingzhong Liu

Qingzhong Liu received his B.E. in 1993 from Northwestern Polytechnical University and M.Sc. in 1997 from Sichuan University of China. Currently he is pursuing his Ph.D. in the Computer Science Department of New Mexico Tech of USA. His research interests include data mining and computational intelligence for information security, bioinformatics, and image analysis.