An Improved Algorithm for Fuzzy Data Mining for Intrusion Detection

German Florez, Susan M. Bridges, and Rayford B. Vaughn

***Abstract*— We have been using fuzzy data mining techniques to extract patterns that represent normal behavior for intrusion detection. In this paper we describe a variety of modifications that we have made to the data mining algorithms in order to improve accuracy and efficiency. We use sets of fuzzy association rules that are mined from network audit data as models of "normal behavior." To detect anomalous behavior, we generate fuzzy association rules from new audit data and compute the similarity with sets mined from "normal" data. If the similarity values are below a threshold value, an alarm is issued. In this paper we describe an algorithm for computing fuzzy association rules based on Borgelt's prefix trees, modifications to the computation of support and confidence of fuzzy rules, a new method for computing the similarity of two fuzzy rule sets, and feature selection and optimization with genetic algorithms. Experimental results demonstrate that we can achieve better running time and accuracy with these modifications.**

***Index Terms*—Fuzzy logic, intrusion detection, data mining**

I. Introduction

An intrusion detection system (IDS) is a component of the computer and information security framework. Its main goal is to differentiate between normal activities of the system and behavior that can be classified as suspicious or intrusive [1]. IDS's are needed because of the large number of incidents reported increases every year and the attack techniques are always improving.

IDS approaches can be divided into two main categories: misuse or anomaly detection [1]. The misuse detection approach assumes that an intrusion can be detected by matching the current activity with a set of intrusive patterns (generally defined by experts or "underground" web sites). Examples of misuse detection include expert systems, keystroke monitoring, and state transition analysis. Anomaly detection systems assume that an intrusion should deviate the system behavior from its normal pattern. This approach can be implemented using statistical methods, neural networks, predictive pattern generation and association rules among others techniques

In previous work [4,10,11] we used the standard Apriori algorithm described by Agrawal and Srikant [2] for mining association rules as modified by Kuok, Fu, and Wong [7] for fuzzy association rules. Fuzzy logic is appropriate for the intrusion detection problem because quantitative features such as the number of different connections or messages are often used for anomaly detection, and because security itself involves fuzziness [4]. In order to detect anomalous behavior, we mine sets of fuzzy association from new audit data and compute the similarity with sets mined from "normal" data. If the similarity values are below a threshold value, an alarm is issued.

Originally we used Kuok, Fu, and Wang's [7] method for computing confidence and support in which an "occurrence" of an itemset is computed by multiplying the memberships of each item. This method of computation has some properties that seem counter-intuitive when applied to intrusion detection. Suppose that an itemset consists of attributes each of which has an associated membership value of 0.9. If the itemset contains 2 attributes, its occurrence would be 0.81. If it contains 5 attributes, its occurrence would be 0.59. The result is that small itemsets tend to have much higher support and confidence than large itemsets. However, we have found that, for our application, larger itemsets tend to be more informative.

Christian Borgelt [3] has developed a much faster algorithm for association rule mining that uses *prefix-trees* to store itemsets with sufficient support and confidence (among other constraints). His algorithm is included in the commercial datamining tool *Clementine* available from *SPSS*. We have adapted this algorithm for mining fuzzy association rules. In standard association rule mining, itemsets are sets of attribute values that have occurred in a transaction. Fuzzy association rule mining is more complicated because the elements of the itemsets may (and usually do) occur to some degree. Instead of just counting items to compute the confidence and support for a rule, we must compute fuzzy occurrences of the items in the itemsets. In order to adapt Borgelt's algorithm to mine fuzzy association rules, we modified the prefix tree structure to store the fuzzy frequencies of each itemset and modified the algorithm to propagate these values through the tree as it is updated.

This paper is organized as follows. Section II presents literature review, Section III describes data preprocessing, section IV shows improvements to the Apriori algorithm, section V describes genetic algorithms for feature selection and optimization, Section VI discusses some experiments

This work was supported in part by the National Science Foundation Grant# CCR-9988524 and the Army Research Laboratory Grant# DAAD17-01-C-0011

with a fuzzy prefix-tree implementation of the Apriori algorithm, and Section VII presents our conclusions.

II. LITERATURE REVIEW

A. Fuzzy Association Rules

Agrawal and Srikant [2] defined an association rule using the following notation: n transactions, $\{T_1, T_2, \ldots, T_n\}$ of m items, $\{ i_1, i_2, \ldots, i_m\}$ in the set of all items, I. Each of the transactions T_j ($1 \le j \le n$) in the database D represents an association of items ($T_j \subseteq I$). An itemset is defined as a non-empty subset of I. An association rule can be represented as: $X \rightarrow Y$, c, s, where $X \subseteq I$, $Y \subset I$, and $X \cap Y = \varnothing$. In this association rule, *s* is called support and *c* is confidence of the association rule. The support is the percentage of the transactions in which both X and Y appear in the same transaction and the confidence is the ratio of the number of transactions that contain both X and Y to the number of transactions that contain only X.

Originally, Agrawal and Srikant [2] implemented the Apriori algorithm to mine single-dimensional Boolean association rules from transactional databases [5]. However, in the intrusion detection field we need to mine quantitative attributes. Kuok, Fu, and Wong [7] integrated fuzzy logic with association rules to handle numerical values and to obtain rules that are more understandable to human beings. Using fuzzy logic, a quantitative feature can be mapped to different fuzzy sets.

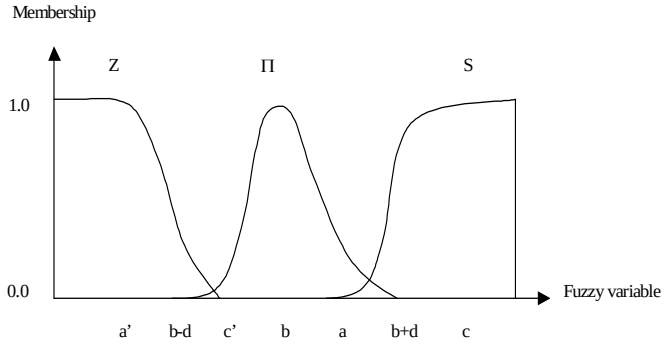

Figure 1. Representation of the standard membership functions: S, Π, and Z

Figure 1 (taken from Shi [11]) shows the standard membership functions Z (low), Π (medium) and S (high) that we are using to express membership values (Each function is controlled by two parameters) .

B. The Apriori Algorithm

The basic Apriori algorithm [2] finds frequent itemsets for Boolean association rules, receiving as input a database T of transactions and the minimum support for the rules. It uses the *Apriori* property: if an itemset I is not frequent, the itemset $I \cup A$ (A is any other item) is also not frequent; i.e. "all nonempty subsets of a frequent itemset must also be frequent" [5, pp. 231].

The Apriori algorithm builds a set C_k (candidate itemsets of size k) and L_k (frequent itemsets of size k) to create frequent itemsets of size k+1:

```
L1={frequent items}
k=2
While(Lk != ∅) {
        Ck = GenerateCandidates(Lk);
        for each transaction t in the database
          Increment the count of all candidates in Ck that
          belong to T
        Lk+1 = candidates in Ck with enough support
        K++;
}
return (L=L1 ∪ L2 ∪ ….);
```

GenerateCandidates() returns a subset of the join operation of L_k and L_k, pruning itemsets that do not satisfy the *Apriori* property. Computing the support and confidence of all nonempty susbsets of each frequent itemset generates the set of association rules

III. DATA PREPROCESSING

We are using the DARPA-MIT dataset for the 1999 intrusion detection systems evaluation [9]. This dataset contains 3 weeks of traffic with normal patterns (attack free) and some old (well-know) attacks. For testing, the fourth and fifth weeks of data contain 201 instances of about 56 types of attacks distributed throughout these two weeks. Specifically, we are interested in network anomaly-detection, so we are using the *tcpdump* binary files that contain the information of packet headers of the outside traffic in the Air Force simulated network.

The first step in the data preprocessing consists of the extraction of an ASCII file with selected fields of the TCP/IP header (features) by using a modified version of *sanitize-tcp* program (downloaded from http://ita.ee.lbl.gov/html/software.html on February 2001). The features extracted are timestamp, source host (renumbered), destination host (renumbered), source port, destination port, flag, and size of each packet.

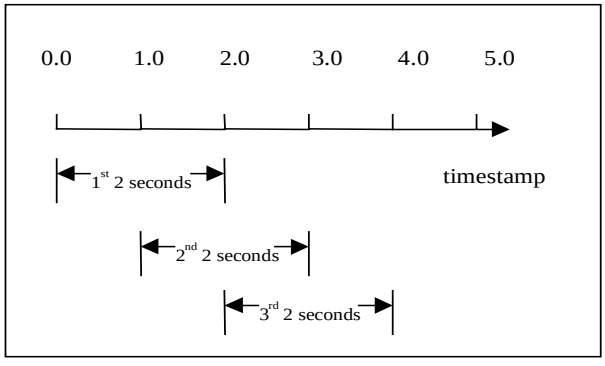

Figure 2. Specification of temporal statistical measurements for the experiments

We have implemented a program that joins those records using the timestamp to compute the frequency of the attributes for consecutive fixed-intervals of time. Such a program extracts 11 features: number of different source IP addresses, number of different destination IP addresses, number of different source ports, number of different destination ports, number of SYN flags, number of FYN flags, number of RST flags, number of packets of normal data (i.e. with no special flag associate with it), number of PSH flags, total size of the packets, and total number of packets. These features not only reflect the header information used by Luo [10] or Shi [11] but also give information about the data transfer itself. We are using a fixed-interval of 2 seconds, as is shown in Figure 2 (taken from Luo [10]).

IV. IMPROVING THE APRIORI ALGORITHM WITH FUZZY VALUES

We have modified the fuzzy association rules algorithm implemented by Lee, Stolfo and Mok [8] and Luo [10], to define a symmetric similarity function of two rules. Given the association rules: R1: X → Y, c, s, R2: X' → Y', c', s', where X, Y, X', and Y' are fuzzy itemsets, the similarity of these rules is given in (1).

$$similarity(R_1, R_2) = \begin{cases} 0 & if\ (X \neq Y) or (X' \neq Y') \\ \max\left(0, 1 - \max\left(\frac{|c - c'|}{\max(c, c')}, \frac{|s - s'|}{\max(s, s')}\right)\right) & if\ (X = Y) and (X' = Y') \end{cases} \quad (1)$$

Given two rule sets S1 and S2, we can define *s* as (2).

$$s = \sum_{\substack{\forall R_1 \in S_1 \\ \forall R_2 \in S_2}} similarity\ (R_1, R_2) \quad (2)$$

The total similarity is given in (3).

$$similarity\ (S_1, S_2) = \frac{s}{|S_1|} * \frac{s}{|S_2|} \quad (3)$$

Figure. 3 shows the similarity between 6 hours of normal data (from week one) with respect to attack free data (from week 3) and test data (week 4). We can see, for instance, that for Monday, the similarity with attack free data is about 0.5, but the similarity with test data (with many attack instances) is almost 0.

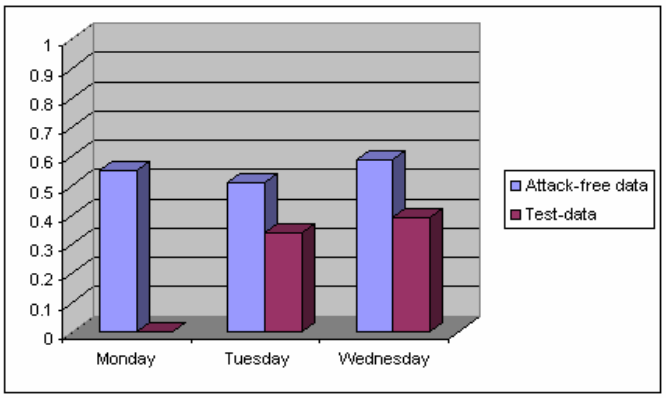

Figure 3. Similarity using 6 hours of data

For the experiments reported in this paper, we are using a minimum confidence of 0.85 and a minimum support of 0.3. We are analyzing 11 features of TCP header packets. This large number of features improves the detection capability of the IDS, but increases the execution time of the algorithm. In order to reduce the running time of the Apriori algorithm, we are penalizing rules with few itemsets (since we have found that large itemsets tend to be more informative for our application and Luo's [10] implementation generates too many uninteresting rules). The basic idea is to modify the minimum support and confidence constraints of a rule taking into account the number of items (features) in the left-hand side (LHS) and right-hand side (RHS). A user threshold *K* (between 0 and 1) is used to control this pruning step. If *K* increases, the number of rules generated decreases. Equations (4) and (5) are used to compute the new support and confidence. It is important to observe that the support or confidence of the rule does not change; only the constraints of rule selection in the candidate generation step of the Apriori algorithm change dynamically.

$$NewMinSup = (1 - \min\_\sup) \frac{(NumberFeatures - CurrentFeatures)}{NumberFeatures} K + \min\_\sup \quad (4)$$

$$NewMinConf = (1 - \min\_conf) \frac{(NumberFeatures - CurrentFeatures)}{NumberFeatures} K + \min\_conf \quad (5)$$

NumberFeatures is the total number of features available, *CurrentFeatures* is the number of itemsets in the LHS and RHS of the rule, *K* is the user threshold, *min_sup* is the minimum support and *min_conf* is the minimum confidence. If *K* is equal to 0, all the rules will have the same confidence and support (i.e. the basic Apriori algorithm). If *K* is 1, then rules with few itemsets in its LHS and RHS must have much more confidence and support to be selected by the algorithm. As an example of this pruning step, consider the following rule (with 3 itemsets) being generated:

(nrf=low) (npf=high) → (tsize=medium)
support = 0.346,confidence= 0.82

If *min_sup* = 0.2 and *min_conf* = 0.8, then with the basic Apriori algorithm (i.e. with *K=0*) this rule would be selected. However, by using *K=0.5*, this rule does not satisfy the new Apriori constraints:

$$NewMinSup = (1 - 0.2) \frac{(11 - 3)}{11} (0.5) + 0.2 = 0.49$$

$$NewMinConf = (1 - 0.8) \frac{(11 - 3)}{11} (0.5) + 0.8 = 0.872$$

Experimental results demonstrate that large values for *K* (near 1) result in a similarity function that is not accurate, because the number of rules generated by the algorithm is too low.

The algorithms were tested with a well know network intrusion, the mailbomb attack, using different amounts of network traffic information (300, 600, 1800 and 3600

seconds). Mailbomb "is an attack in which the attacker sends many messages to a server, overflowing that server's mail queue and possible causing system failure." [6,pp. 44].

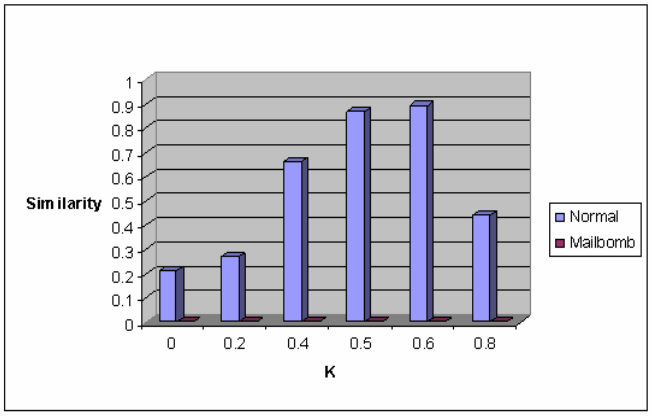

Figure 4. Impact of *K* for the mailbomb attack detection (300 seconds)

Figure 4 shows the similarity between an attack-free set, with normal traffic and mailbomb sets, using an interval of time of 300 seconds. The x-axis corresponds to *K*, and y-axis is the similarity. We can see how *K* values near 0 or 1 cause the similarities to drop (with 0, there are too many rules; with 1, there are too few). A *K* value near 0.5 gives the best accuracy. Figure 5 shows the results of an experiment with 600 seconds of network traffic data and Figure 6 with 3600 seconds (with *K=0* we obtained a false positive because of the large amount of uninteresting rules).

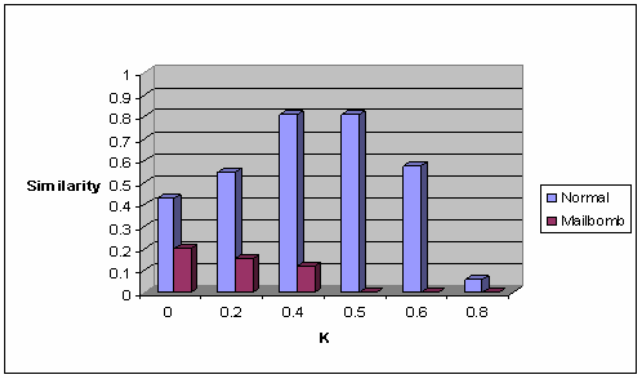

Figure 5. Impact of *K* for the mailbomb attack detection (600 seconds)

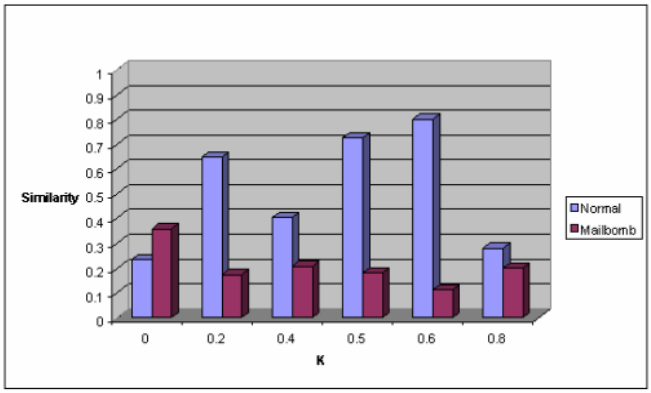

Figure 6. Impact of *K* for the mailbomb attack detection (3600 seconds)

Figure 7 shows the impact of *K* in the number of rules produced by the fuzzy association rules algorithm.

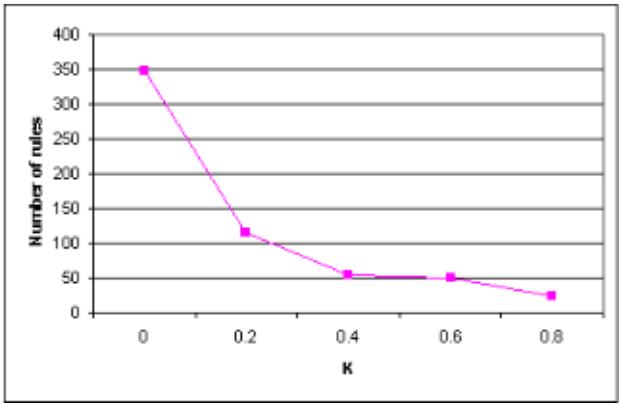

Figure 7. Impact of *K* in the number of rules generated (1800 seconds)

V. USING GENETIC ALGORITHMS

Genetics algorithms (GA) are used for feature selection (reducing the running time and improving the accuracy of the Apriori algorithm) and to optimize the membership functions (Each function Z, Π and S is controlled by two parameters).

Shi [11] implemented a GA (using Luo's algorithm [10] to compute fitness) where the main goal is "to maximize the similarity of the reference rule set and normal rule sets mined from data that contains no intrusions while minimizing the similarity of the reference rule set and abnormal rule set mined from data containing intrusions" [11, pp. 57].

Features are represented with binary values, and each one is associated with the fuzzy membership function parameters (a real number). See Figure 8 (taken from Shi [11]). The upper box indicates whether the "feature" is selected or not. The bottom represents the fuzzy membership function parameters to be optimized.

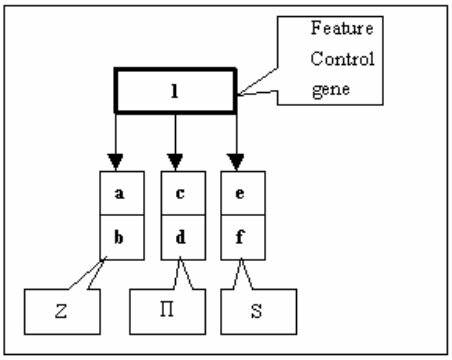

Figure 8. The representation of a single fuzzy variable

This algorithm uses roulette selection, uniform crossover and a bit flipping mutation strategy.

We have implemented the following improvements to Shi's algorithm:

1) Inclusion of the pruning step for relative support and confidence (see Section IV) in the execution of Luo's algorithm for the fitness evaluation.

2) Modification of the fitness function. The fitness function is computed with (6). $S_{ref,norm}$ is the similarity between the reference rule set and the normal rule set, and $S_{ref,abnorm}$ is the similarity between the reference rule set and the abnormal rule set.

$$fitness = \frac{S_{ref,norm}}{S_{ref,abnorm}} \quad (6)$$

Shi [11] correctly stated that by using this equation a high fitness value could be achieved by evolving parameters that caused no rules to be mined. He tried to solve the problem by using a threshold value to penalize chromosomes that produce a very small number of rules for the reference set. We have modified this approach to decrease the value of the fitness of an individual (Equation 6) taking into account *Rr* (number of reference set rules), *Nr* (number of normal set rules), *Ar* (number of abnormal set rules), and *p* (a user threshold):

$$Tr = p * \frac{Rr + Nr + Ar}{100}$$
$$if\ (Nr < Tr) \rightarrow Fitness = Fitness * \frac{Nr}{Tr}$$
$$if\ (Ar < Tr) \rightarrow Fitness = Fitness * \frac{Ar}{Tr}$$

In order to prevent *overfitting* and to give more exploration to the system, we have changed the fitness evaluation framework. In [11] the general approach of the fitness function is:

Use ReferenceRuleSet
compare with NormalRuleSet (similarity must be high)
compare with AbnormalRuleSet(similarity must be low)

The new approach uses different reference rule sets. Thus, the fitness function selects randomly a reference set each time an individual is being evaluated:

Select ANY file from the set of ReferenceRuleSet files
compare with NormalRuleSet (similarity must be high)
compare with AbnormalRuleSet(similarity must be low)

When the GA is completed successfully, we save the fuzzy membership function parameters and the selected features. Since the GA learned from different reference sets, the optimization is more general. As an example consider Figure 9. We have randomly selected a reference set (with normal data) and we have executed the fuzzy association rules algorithm with and without the GA optimization. The data shown for the GA optimization is the average of 10 executions. The GA did not select more than 7 features (reducing about 60% of the execution time of the algorithm), however, it tried to increase the similarity between the reference set and the normal profile (using 600 seconds), while tried to decrease the similarity between the reference set and the mailbomb attack (using 1800 seconds).

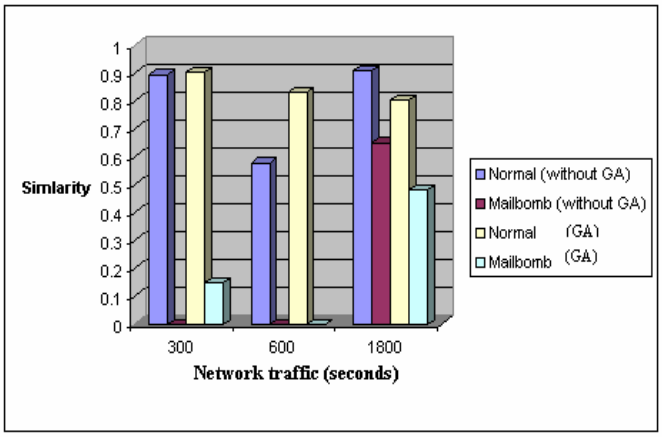

Figure 9. Comparison of the fuzzy association rules algorithm with and without GA optimization

VI. PREFIX TREES

Christian. Borgelt [3], from the Otto-von-Guericke-University of Magdeburg, implemented an efficient version of the Apriori algorithm (*Find Association Rules/Hyperedges with Apriori Algorithm*). The use of this software is under the terms of the GNU Public License, and it was downloaded from http://fuzzy.cs.uni-magdeburg.de/~borgelt/apriori/apriori.html (Accessed on August, 2001).

Borgelt uses prefix trees to store itemsets with enough support and confidence (among others constraints). We have modified the software to include the following features:

1) Computation of fuzzy membership values for each item in the input dataset (the fuzzy functions Z, Π and S are applied to the input dataset before Borgelt's program is executed).
2) Computation of the total support and confidence of a fuzzy itemset with the use of the operator MULT (multiplication of the membership of each item) or MIN (minimum membership of the items).
3) Inclusion of the pruning step for relative support and confidence (see Section IV).

As an example of this algorithm, consider the small dataset in Table 2 with three items (columns): A, B, and C. Each item has associated a membership value for each transaction (row). The fuzzy prefix tree is shown in Figure 10 (using MULT): A has support of 50% (2.0 units) and 50% of confidence, A ← B has support of 8.25% (0.33 units) and 35.67% confidence, and C ← A B has support of 1.80% (0.072 units) and 21.82% confidence.

TABLE 2. A DATABASE WITH MEMBERSHIPS

0.5A	0.3B	
	0.4B	0.9C
0.6A		0.1C
0.9A	0.2B	0.4C

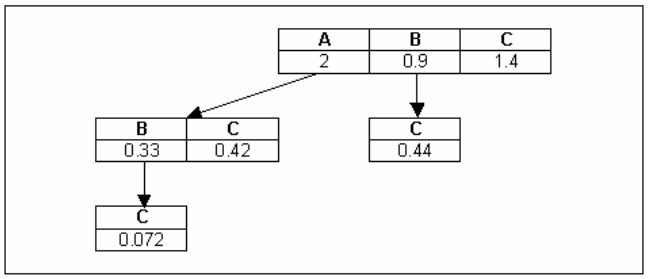

Figure 10. Prefix tree with fuzzy memberships

The main advantage of this approach is efficiency. There is not need for candidate generation since the frequent itemsets of size k+1 (i.e. the new levels in the tree) are obtained directly by reading the dataset. It also stores the fuzzy frequency of each one of the possible itemsets, so the time spent in the generation of the association rules is reduced dramatically. However, this algorithm only builds rules with one item in the LHS.

A program was written to compute the similarity between two set of rules generated by our version of Borgelt's algorithm. Fig. 11 shows a comparison of the running time of the fuzzy association rules algorithms. The prefix-tree implementation of the Apriori algorithm is much faster than the naïve Apriori. Although we are generating rules with only one item in the LHS, the algorithm is able to distinguish between normal and suspicious behavior. Table 3 shows the accuracy of the Apriori algorithms with the GA optimization, K=0.6, *min_sup*=0.3 and *min_conf*=0.85.

TABLE 3. COMPARISON OF THE ACCURACY OF THE FUZZY ASSOCIATION RULES ALGORITHMS

	SIMILARITY			
	APRIORI		PREFIX-TREE	
Network traffic (seconds)	Normal	Mailbomb	Normal	Mailbomb
300	0.89	0	0.18	0
600	0.57	0	0.81	0
1800	0.908	0.65	0.72	0.03
3600	0.801	0.115	0.9	0.1

VII. CONCLUSIONS

Fuzzy association rules can be used to implement a network intrusion detection system based upon the assumption that an attack can be identified as burst traffic in audit logs. In order to improve the accuracy of such IDS, we have been using 11 features of the TCP header packet. However, with this relatively large number of features we obtain many uninteresting rules (since the number of possible combinations of the fuzzy variables is quite large) and the running time of the Apriori algorithm increases dramatically.

To solve this problem we have introduced a new pruning rule in the candidate generation of the Apriori algorithm to penalize itemsets with few items. Experiments showed that this pruning step not only reduces the number of rules, but also increase the accuracy of the system. We have also improved a genetic algorithm to optimize the fuzzy association rules. The GA is able to learn from different reference sets, eliminating the overfitting problem. Prefix trees seems to be an efficient alternative to find association rules with only one item in the LHS. We have included fuzzy logic management in the software provided by Borgelt, and we have obtained similar accuracy with respect to the basic Apriori algorithm. However, the running time of Borgelt's algorithm is much lower, and its use may enable online detection.

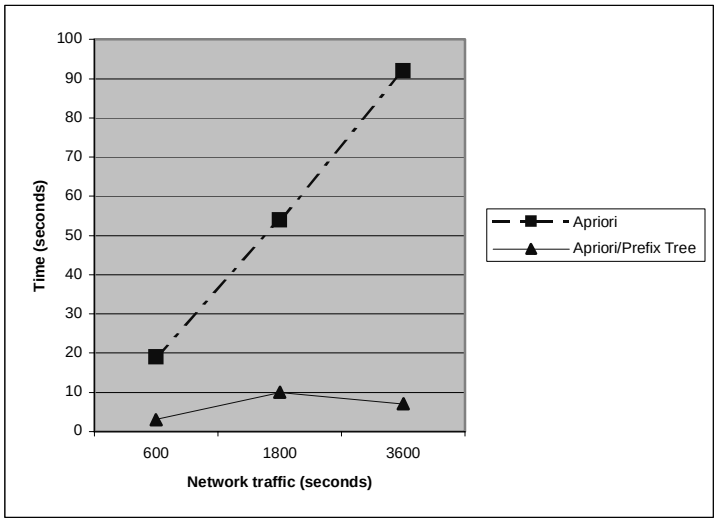

Figure 11. Comparison of the running time of the fuzzy association rules algorithms

REFERENCES

[1] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner. 2000. State of the Practice of Intrusion Detection Technologies. CMU/SEI-99-TR-028. Carnegie Mellon Software Engineering Institute

[2] R.Agrawal,, and R. Srikant. 1994. Fast algorithms for mining association rules. In *Proceedings of the 20 th international conference on very large databases held in Santiago, Chile, September 12-15, 1994*, 487-99.

[3] C. Borgelt, 2001. Association Rule Induction. http://fuzzy.cs.uni-magdeburg.de/~borgelt.

[4] S.M. Bridges, and Rayford M. Vaughn. 2000. Fuzzy data mining and genetic algorithms applied to intrusion detection. In Proceedings 23 rd National Information Systems Security Conference, Oct. 16-19, 2000, Baltimore, MD, pp. 13-31.

[5] J. Han, and M. Kamber. *Data mining: Concepts and techniques*. Morgan Kaufmann. San Francisco, CA. 2001

[6] K. Kendall, *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems*, Master's thesis, Massachusetts Institute of Technology, 1998.

[7] C. Kuok, A. Fu, and M. Wong. 1998. Mining fuzzy association rules in databases. *SIGMOD Record* 27(1): 41-6.

[8] W. Lee, S. Stolfo, and K. Mok. 1998. Mining audit data to build intrusion detection models. In *Proceedings of the fourth international conference on knowledge discovery and data mining held in New York, New York, August 27-31, 199*8, edited by Rakesh Agrawal, and Paul Stolorz, 66-72. New York, NY: AAAI Press.

[9] R.P. Lippmann, D.J. Fried, I. Graf, J.W. Haines,K.R. Kendall, D. McClung, D. Weber, et al. Evaluating Intrusion Detection Systems: the 1998 DARPA Off-Line Intrusion Detection Evaluation, *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, 2000, Vol. 2

[10] J. Luo, and Susan M. Bridges. 2000. Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection. *International Journal of Intelligent Systems* 15: 687-703.

[11] F. Shi, *Genetic algorithms for feature selection in an intrusion detection application*. Master's thesis, Dept. Computer Science, Mississippi State University, 2000.