9. Evolutionary Computation in Intelligent Network Management

Ajith Abraham

Natural Computation Lab
Department of Computer Science
Oklahoma State University,USA
ajith.abraham@ieee.org

Abstract: Data mining is an iterative and interactive process concerned with discovering patterns, associations and periodicity in real world data. This chapter presents two real world applications where evolutionary computation has been used to solve network management problems. First, we investigate the suitability of linear genetic programming (LGP) technique to model fast and efficient intrusion detection systems, while comparing its performance with artificial neural networks and classification and regression trees. Second, we use evolutionary algorithms for a Web usage-mining problem. Web usage mining attempts to discover useful knowledge from the secondary data obtained from the interactions of the users with the Web. Evolutionary algorithm is used to optimize the concurrent architecture of a fuzzy clustering algorithm (to discover data clusters) and a fuzzy inference system to analyze the trends. Empirical results clearly shows that evolutionary algorithm could play a major rule for the problems considered and hence an important data mining tool.

9.1 Intrusion Detection Systems

Security of computers and the networks that connect them is increasingly becoming of great significance. Computer security is defined as the protection of computing systems against threats to confidentiality, integrity, and availability. There are two types of intruders: the external intruders who are unauthorized users of the machines they attack, and internal intruders, who have permission to access the system with some restrictions. The traditional prevention techniques such as user authentication, data encryption, avoiding programming errors and firewalls are used as the first line of defense for computer security. If a password is weak and is compromised, user authentication cannot prevent unauthorized use, firewalls are vulnerable to errors in configuration and ambiguous or undefined security policies. They are generally unable to protect against malicious mobile code, insider attacks and unsecured modems. Programming errors cannot be avoided as the complexity of the system and application software is changing rapidly leaving behind some exploitable weaknesses. Intrusion detection is therefore required as an additional wall for protecting systems [9.10, 9.15]. Intrusion detection is use-

ful not only in detecting successful intrusions, but also provides important information for timely countermeasures [9.18, 9.19]. An intrusion is defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource. An attacker can gain access because of an error in the configuration of a system. In some cases it is possible to fool a system into giving access by misrepresenting oneself. An example is sending a TCP packet that has a forged source address that makes the packet appear to come from a trusted host. Intrusions may be classified into several types [9.19] .

- Attempted break-ins, which are detected by typical behavior profiles or violations of security constraints.
- Masquerade attacks, which are detected by atypical behavior profiles or violations of security constraints.
- Penetration of the security control system, which are detected by monitoring for specific patterns of activity.
- Leakage, which is detected by atypical use of system resources.
- Denial of service, which is detected by atypical use of system resources.
- Malicious use, which is detected by atypical behavior profiles, violations of security constraints, or use of special privileges.

The process of monitoring the events occurring in a computer system or network and analyzing them for sign of intrusions is known as Intrusion detection. Intrusion detection is classified into two types: misuse intrusion detection and anomaly intrusion detection.

- Misuse intrusion detection uses well-defined patterns of the attack that exploit weaknesses in system and application software to identify the intrusions. These patterns are encoded in advance and used to match against the user behavior to detect intrusion.
- Anomaly intrusion detection uses the normal usage behavior patterns to identify the intrusion. The normal usage patterns are constructed from the statistical measures of the system features, for example, the CPU and I/O activities by a particular user or program. The behavior of the user is observed and any deviation from the constructed normal behavior is detected as intrusion.

We have two options to secure the system completely, either prevent the threats and vulnerabilities which come from flaws in the operating system as well as in the application programs or detect them and take some action to prevent them in future and also repair the damage. It is impossible in practice, and even if possible, extremely difficult and expensive, to write a completely secure system. Transition to such a system for use in the entire world would be an equally difficult task. Cryptographic methods can be compromised if the passwords and keys are stolen. No matter how secure a system is, it is vulnerable to insiders who abuse their privileges. There is an inverse

relationship between the level of access control and efficiency. More access controls make a system less user-friendly and more likely of not being used.

An Intrusion Detection system is a program (or set of programs) that analyzes what happens or has happened during an execution and tries to find indications that the computer has been misused. An Intrusion detection system does not eliminate the use of preventive mechanism but it works as the last defensive mechanism in securing the system. Data mining approaches are a relatively new technique for intrusion detection.

9.1.1 Intrusion Detection - a Data Mining Approach

Data mining is a relatively new approach for intrusion detection. Data mining approaches for intrusion detection was first implemented in Mining Audit Data for Automated Models for Intrusion Detection [9.14] . The raw data is first converted into ASCII network packet information which in turn is converted into connection level information. These connection level records contain within connection features like service, duration etc. Data mining algorithms are applied to this data to create models to detect intrusions. Data mining algorithms used in this approach are RIPPER (rule based classification algorithm), meta-classifier, frequent episode algorithm and association rules. These algorithms are applied to audit data to compute models that accurately capture the actual behavior of intrusions as well as normal activities.

The RIPPER algorithm was used to learn the classification model in order to identify normal and abnormal behavior [9.8] . Frequent episode algorithm and association rules together are used to construct frequent patterns from audit data records. These frequent patterns represent the statistical summaries of network and system activity by measuring the correlations among system features and sequential co-occurrence of events. From the constructed frequent patterns the consistent patterns of normal activities and the unique intrusion patterns are identified and analyzed, and then used to construct additional features. These additional features are useful in learning the detection model more efficiently in order to detect intrusions. RIPPER classification algorithm is then used to learn the detection model. Meta classifier is used to learn the correlation of intrusion evidence from multiple detection models and produce combined detection model. The main advantage of this system is automation of data analysis through data mining, which enables it to learn rules inductively replacing manual encoding of intrusion patterns. However, some novel attacks may not be detected.

Audit Data Analysis and Mining combines association rules and classification algorithm to discover attacks in audit data [9.5] . Association rules are used to gather necessary knowledge about the nature of the audit data as the information about patterns within individual records can improve the classification efficiency. This system has two phases, training phase and detection phase. In the training phase database of frequent item sets is created

for the attack-free items from using only attack-free data set. This serves as a profile against which frequent item sets found later will be compared. Next a sliding-window, on-line algorithm is used to find frequent item sets in the last D connections and compares them with those stored in the attack-free database, discarding those that are deemed normal. In this phase classifier is also trained to learn the model to detect the attack. In the detection phase a dynamic algorithm is used to produce item sets that are considered as suspicious and used by the classification algorithm already learned to classify the item set as attack, false alarm (normal event) or as unknown. Unknown attacks are the ones which are not able to detect either as false alarms or as known attacks. This method attempts to detect only anomaly attacks.

9.1.2 Linear Genetic Programming (LGP)

Linear genetic programming is a variant of the GP technique that acts on linear genomes. Its main characteristics in comparison to tree-based GP lies in that the evolvable units are not the expressions of a functional programming language (like LISP), but the programs of an imperative language (like c/c ++) [9.7] . An alternate approach is to evolve a computer program at the machine code level, using lower level representations for the individuals. This can tremendously hasten up the evolution process as, no matter how an individual is initially represented, finally it always has to be represented as a piece of machine code, as fitness evaluation requires physical execution of the individuals. The basic unit of evolution here is a native machine code instruction that runs on the floating-point processor unit (FPU). Since different instructions may have different sizes, here instructions are clubbed up together to form instruction blocks of 32 bits each. The instruction blocks hold one or more native machine code instructions, depending on the sizes of the instructions. A crossover point can occur only between instructions and is prohibited from occurring within an instruction. However the mutation operation does not have any such restriction. One of the most serious problems of standard genetic programming is the convergence of the population. It has been often observed that unless convergence is achieved within certain number of generations, the system will never converge. Parallel populations or demes may possess different parameter settings that can be explored simultaneously, or they may cooperate with the same set of parameters, while each working on different individuals. In this research, we used circular movement of evolved programs among the demes, i.e., program movement can take place only between adjacent demes in the circle. Steady state genetic programming approach was used to manage the memory more effectively.

9.1.3 Decision Trees (DT) as Intrusion Detection Model

Intrusion detection can be considered as classification problem where each connection or user is identified either as one of the attack types or normal

based on some existing data. Decision trees work well with large data sets. This is important as large amounts of data flow across computer networks. The high performance of Decision trees makes them useful in real-time intrusion detection. Decision trees construct easily interpretable models, which is useful for a security officer to inspect and edit. These models can also be used in the rule-based models with minimum processing [9.13] . Generalization accuracy of decision trees is another useful property for intrusion detection model. There will always be some new attacks on the system, which are small variations of known attacks after the intrusion detection models are built. The ability to detect these new intrusions is possible due to the generalization accuracy of decision trees.

9.1.4 Support Vector Machines (SVM)

Support Vector Machines have been proposed as a novel technique for intrusion detection. SVM maps input (real-valued) feature vectors into a higher dimensional feature space through some nonlinear mapping. SVMs are powerful tools for providing solutions to classification, regression and density estimation problems. These are developed on the principle of structural risk minimization. Structural risk minimization seeks to find a hypothesis h for which one can find lowest probability of error. The structural risk minimization can be achieved by finding the hyper plane with maximum separable margin for the data [9.22] .

Computing the hyper plane to separate the data points i.e. training a SVM leads to quadratic optimization problem. SVM uses a feature called kernel to solve this problem. Kernel transforms linear algorithms into nonlinear ones via a map into feature spaces. There are many kernel functions; some of them are Polynomial, radial basis functions, two layer sigmoid neural nets etc. The user may provide one of these functions at the time of training classifier, which selects support vectors along the surface of this function. SVMs classify data by using these support vectors, which are members of the set of training inputs that outline a hyper plane in feature space. The main disadvantage is SVM can only handle binary-class classification whereas intrusion detection requires multi-class classification.

9.1.5 Intrusion Detection Data

In 1998, DARPA intrusion detection evaluation program created an environment to acquire raw TCP/IP dump data for a network by simulating a typical U.S. Air Force LAN [9.16]. The LAN was operated like a real environment, but being blasted with multiple attacks. For each TCP/IP connection, 41 various quantitative and qualitative features were extracted. Of this database a subset of 494021 data were used for our studies, of which 20% represent normal patterns [9.12]. Different categories of attacks are summarized in Fig. 9.1. Attack types fall into four main categories:

1. **Probing: surveillance and other probing**
   Probing is a class of attacks where an attacker scans a network to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use the information to look for exploits. There are different types of probes: some of them abuse the computer's legitimate features; some of them use social engineering techniques. This class of attacks is the most commonly heard and requires very little technical expertise.

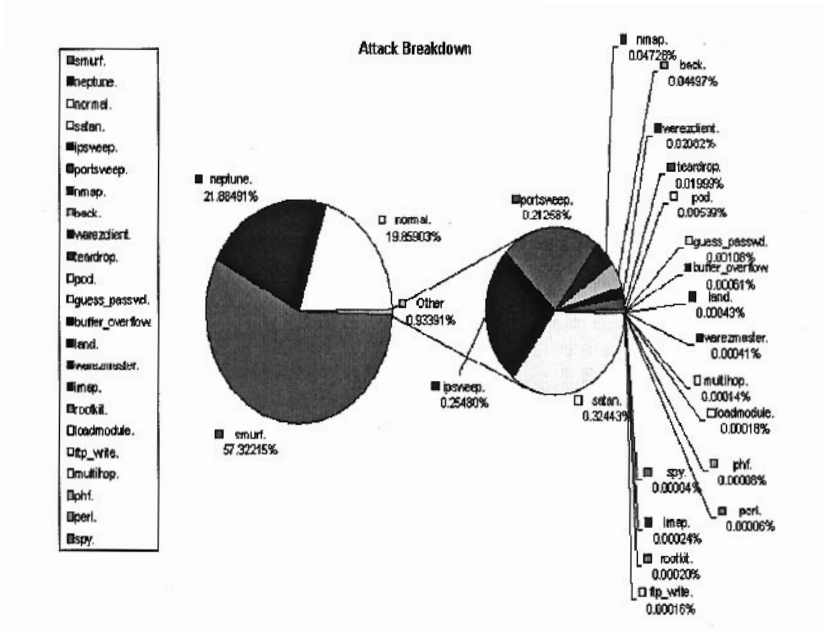

Fig. 9.1. Intrusion detection data distribution

2. **DoS: denial of service**
   Denial of Service (DoS) is a class of attacks where an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, thus denying legitimate users access to a machine. There are different ways to launch DoS attacks: by abusing the computers legitimate features; by targeting the implementations bugs; or by exploiting the system's misconfigurations. DoS attacks are classified based on the services that an attacker renders unavailable to legitimate users.
3. **U2Su: unauthorized access to local super user (root) privileges**
   User to root (U2Su) exploits are a class of attacks where an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system. Most common

exploits in this class of attacks are regular buffer overflows, which are caused by regular programming mistakes and environment assumptions.

4. **R2L: unauthorized access from a remote machine**
   A remote to user (R2L) attack is a class of attacks where an attacker sends packets to a machine over a network, then exploits machine's vulnerability to illegally gain local access as a user. There are different types of R2U attacks; the most common attack in this class is done using social engineering.

Experimentation setup and results. We performed a 5-class classification. The (training and testing) data set contains 11982 randomly generated points from the data set representing the five classes, with the number of data from each class proportional to its size, except that the smallest class is completely included. The set of 5092 training data and 6890 testing data are divided in to five classes: normal, probe, denial of service attacks, user to super user and remote to local attacks. Where the attack is a collection of 22 different types of instances that belong to the four classes described earlier and the other is the normal data. The normal data belongs to class 1, probe belongs to class 2, denial of service belongs to class 3, user to super user belongs to class 4, remote to local belongs to class 5. Two randomly generated separate data sets of sizes 5092 and 6890 are used for training and testing the LGP, DT and SVM respectively.

Experiments using linear genetic programming. The settings of various linear genetic programming system parameters are of utmost importance for successful performance of the system. The population space has been subdivided into multiple subpopulation or demes. Migration of individuals among the subpopulations causes evolution of the entire population. It helps to maintain diversity in the population, as migration is restricted among the demes. Moreover, the tendency towards a bad local minimum in one deme can be countered by other demes with better search directions. The various LGP search parameters are the mutation and the crossover frequencies. The crossover operator acts by exchanging sequences of instructions between two tournament winners. After a trial and error approach, the following parameter settings were used to develop IDS.

Figs. 9.2, 9.3, 9.4, 9.5 and 9.6 demonstrates the growth in program length during 120,000 tournaments and the average fitness values for all the five classes. Test data classification accuracy is depicted in Table 9.2.

Experiments using support vector machines. Our trial experiments revealed that the polynomial kernel option often performs well on most of the data sets. Classification accuracies for the different types of attacks (test data) are depicted in Table 9.2

Experiments using decision trees. First a classifier is constructed using the training data and then testing data is tested with the constructed classifier to classify the data into normal or attack. Table 9.2 summarizes the results of the test data.

Table 9.1. Parameter settings for linear genetic programming

Parameter	Normal	Probe	DoS	U2Su	R2L
Population size	2048	2048	2048	2048	2048
Maximum no of tournaments	120000	120000	120000	120000	120000
Tournament size	8	8	8	8	8
Mutation frequency (%)	85	82	75	86	85
Crossover frequency (%)	75	70	65	75	70
Number of demes	10	10	10	10	10
Maximum program size	256	256	256	256	256

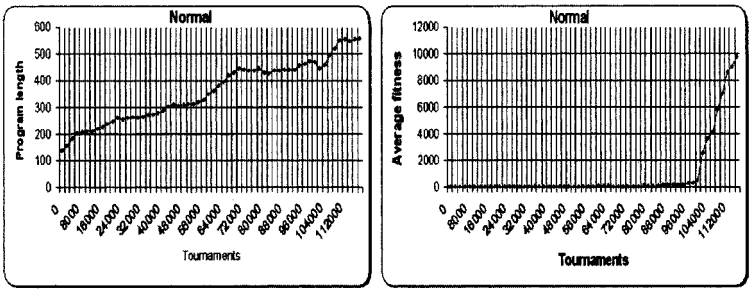

Fig. 9.2. Detection of normal patterns (a) growth in program length (b) average fitness

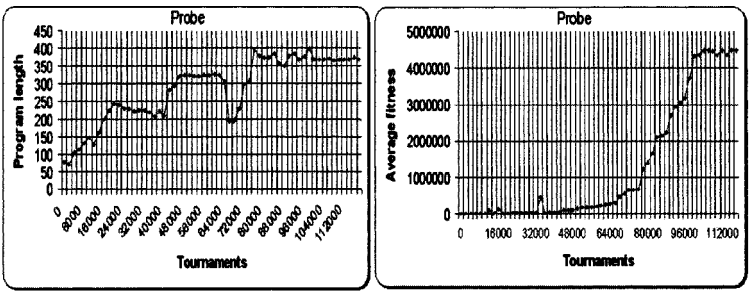

Fig. 9.3. Detection of probe (a) Growth in program length (b) average training fitness

Table 9.2. Parameter settings for linear genetic programming

Class type	Classification accuracy (%)		
	DT	SVM	LGP
Normal	99.64	99.64	99.73
Probe	99.86	98.57	99.89
DOS	96.83	99.92	99.95
U2R	68.00	40.00	64.00
R2L	84.19	33.92	99.47

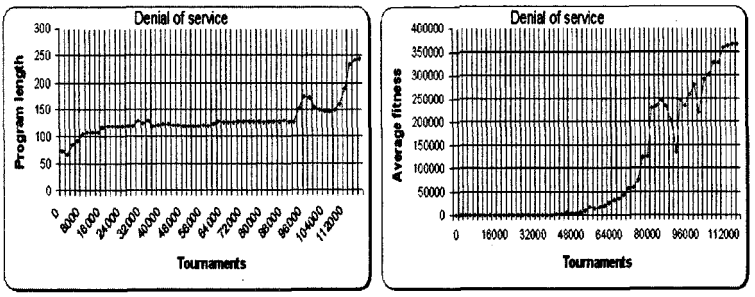

Fig. 9.4. Detection of DoS (a) growth in program length (b) average training fitness

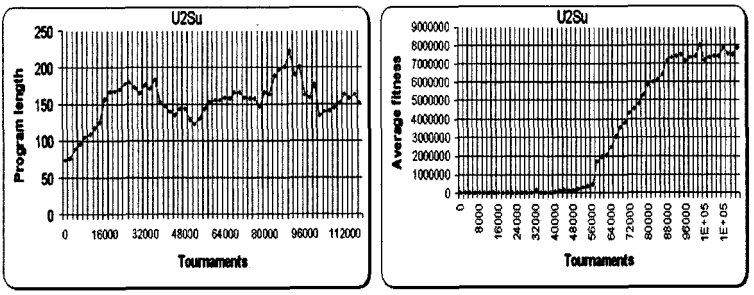

Fig. 9.5. Detection of U2Su (a) growth in program length (b) average training fitness

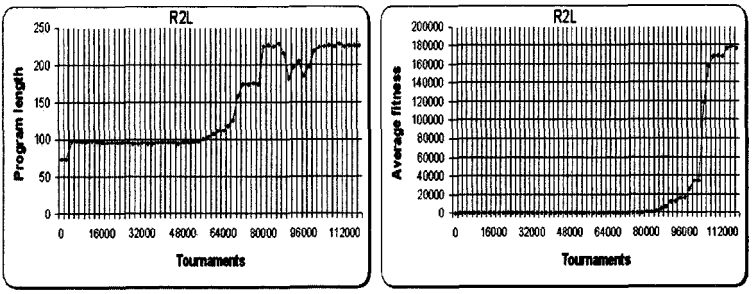

Fig. 9.6. Detection of R2L (a) growth in program length (b) average training fitness

9.1.6 Discussions

A number of observations and conclusions are drawn from the results illustrated in Table 9.1. LGP outperformed decision trees and support vector machines in terms of detection accuracies (except for one class). Decision trees could be considered as the second best, especially for the detection of U2R attacks. In some classes the accuracy figures tend to be very small and may not be statistically significant, especially in view of the fact that the 5 classes of patterns differ in their sizes tremendously. More definitive conclusions can only be made after analyzing more comprehensive sets of network traffic data.

9.2 Web usage Mining using Intelligent Miner (i-Miner)

The WWW continues to grow at an amazing rate as an information gateway and as a medium for conducting business. From the business and applications point of view, knowledge obtained from the Web usage patterns could be directly applied to efficiently manage activities related to e-business, e-services, e-education and so on. Web usage could be used to discover the actual contents of the web pages (text, images etc.), organization of the hyperlink architecture (HTML/XML links etc.) of different pages and the data that describes the access patterns (Web server logs etc.) [9.1, 9.20] . A typical Web log format is depicted in Fig. 9.7. When ever a visitor access the server it leaves the IP, authenticated user ID, time/date, request mode, status, bytes, referrer, agent and so on. The available data fields are specified by the HTTP protocol. In the case of Web mining, data could be collected at the server level, client level, proxy level or some consolidated data. These data could differ in terms of content and the way it is collected etc. The usage data collected at different sources represent the navigation patterns of different segments of the overall Web traffic, ranging from single user, single site browsing behavior to multi-user, multi-site access patterns. Web server log does not accurately contain sufficient information for inferring the behavior at the client side as they relate to the pages served by the Web server. Pre-processed and cleaned data could be used for pattern discovery, pattern analysis, Web usage statistics and generating association/ sequential rules.

We present a hybrid Web usage mining framework (i-miner) as depicted in Fig. 9.8 [9.4, 9.2]. by clustering the visitors and analyzing the trends using some function approximation algorithms. The hybrid framework optimizes a fuzzy clustering algorithm using an evolutionary algorithm and a Takagi-Sugeno fuzzy inference system using a combination of evolutionary algorithm and neural network learning. The raw data from the log files are cleaned and pre-processed and a fuzzy C means algorithm is used to identify the number of clusters. The developed clusters of data are fed to a Takagi-Sugeno fuzzy inference system to analyze the trend patterns. The if-then rule structures are

```
64.68.82.66 - - [17/May/2003:03:41:23 -0500] "GET /marcin HTTP/1.0" 404 318
192.114.47.54 - - [17/May/2003:03:41:33 -0500] "GET /~aa/isda2002/isda2002.html HTTP/1.1" 404 350
216.239.37.5 - - [17/May/2003:03:41:43 -0500] "GET /~ijcr/Vol5/vol10no1.html HTTP/1.0" 200 4568
218.244.111.106 - - [17/May/2003:03:41:51 -0500] "GET /~aa/his/ HTTP/1.1" 404 332
64.68.82.18 - - [17/May/2003:03:42:15 -0500] "GET /~pdcp/cfp/cfpBookReviews.html HTTP/1.0" 304 -
212.98.136.62 - - [17/May/2003:03:43:21 -0500] "GET /cs3373/programs/pgm03.dat HTTP/1.1" 200 498
212.98.136.62 - - [17/May/2003:03:43:26 -0500] "GET /cs3373/programs/pgm04.html HTTP/1.1" 200 55722
212.98.136.62 - - [17/May/2003:03:43:38 -0500] "GET /cs3373/images/WaTor.gif HTTP/1.1" 200 39021
212.29.232.2 - - [17/May/2003:03:43:40 -0500] "GET /welcome.html HTTP/1.0" 200 5253
```

Fig. 9.7. Sample entries from a Web server access log

learned using an iterative learning procedure by an evolutionary algorithm and the rule parameters are fine-tuned using a backpropagation algorithm.

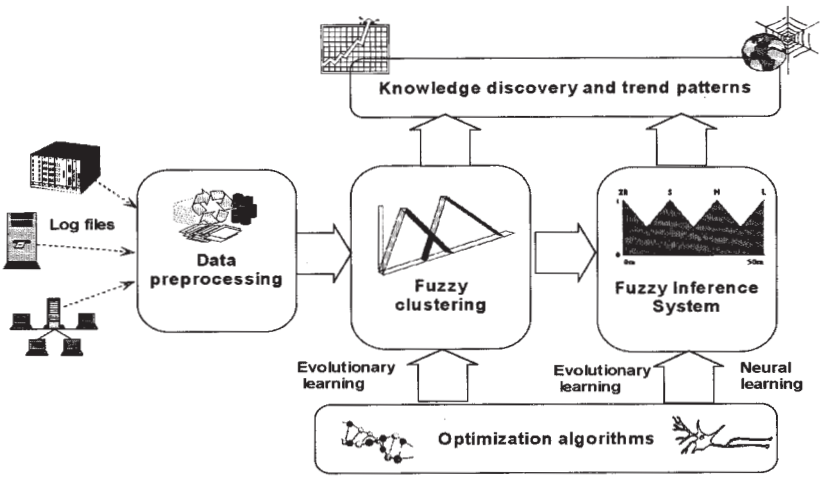

Fig. 9.8. i-Miner framework

The hierarchical distribution of the i-Miner is depicted in Fig. 9.9. The arrow direction depicts the speed of the evolutionary search. The optimization of clustering algorithm progresses at a faster time scale in an environment decided by the inference method and the problem environment.

9.2.1 Optimization of Fuzzy Clustering Algorithm

One of the widely used clustering methods is the fuzzy c-means (FCM) algorithm developed by Bezdek [9.6] . FCM partitions a collection of n vectors $x_i = 1, 2, ..., n$ into c fuzzy groups and finds a cluster center in each group such that a cost function of dissimilarity measure is minimized. To accommodate the introduction of fuzzy partitioning, the membership matrix U is allowed to have elements with values between 0 and 1.The FCM objective function takes the form

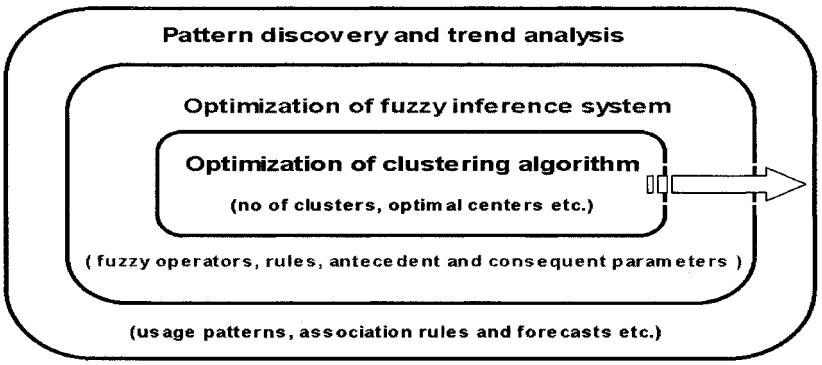

Fig. 9.9. Hierarchical architecture of i-Miner

$$J(U, c_1, ...c_c) = \sum_{i=1}^{c} Ji = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m d_{ij}^2, \tag{9.1}$$

where u_{ij}, is a numerical value between [0,1]; c_i is the cluster center of fuzzy group i; $d_{ij} = \|c_i - x_j\|$ is the Euclidian distance between i^{th} cluster center and j^{th} data point; and m is called the exponential weight which influences the degree of fuzziness of the membership (partition) matrix. Usually a number of cluster centers are randomly initialized and the FCM algorithm provides an iterative approach to approximate the minimum of the objective function starting from a given position and leads to any of its local minima [9.6] . No guarantee ensures that FCM converges to an optimum solution (can be trapped by local extrema in the process of optimizing the clustering criterion). The performance is very sensitive to initialization of the cluster centers. An evolutionary algorithm is used to decide the optimal number of clusters and their cluster centers. The algorithm is initialized by constraining the initial values to be within the space defined by the vectors to be clustered. A very similar approach is given in [9.11] .

9.2.2 Optimization of the Fuzzy Inference System

We used the EvoNF framework [9.3], which is an integrated computational framework to optimize fuzzy inference system using neural network learning and evolutionary computation. Solving multi-objective scientific and engineering problems is, generally, a very difficult goal. In these particular optimization problems, the objectives often conflict across a high-dimension problem space and may also require extensive computational resources. The hierarchical evolutionary search framework could adapt the membership functions (shape and quantity), rule base (architecture), fuzzy inference mechanism (T-norm and T-conorm operators) and the learning parameters of neural network learning algorithm. In addition to the evolutionary learning (global

search) neural network learning could be considered as a local search technique to optimize the parameters of the rule antecedent/consequent parameters and the parameterized fuzzy operators. The hierarchical search could be formulated as follows: For every fuzzy inference system, there exist a global search of neural network learning algorithm parameters, parameters of the fuzzy operators, if-then rules and membership functions in an environment decided by the problem. The evolution of the fuzzy inference system will evolve at the slowest time scale while the evolution of the quantity and type of membership functions will evolve at the fastest rate. The function of the other layers could be derived similarly. Hierarchy of the different adaptation layers (procedures) will rely on the prior knowledge (this will also help to reduce the search space). For example, if we know certain fuzzy operators will work well for a problem then it is better to implement the search of fuzzy operators at a higher level. For fine-tuning the fuzzy inference system all the node functions are to be parameterized. For example, the Schweizer and Sklar's T-norm operator can be expressed as:

$$T(a,b,p) = [max\{0,(a^{-p}+b^{-p}-1\})]^{-\frac{1}{p}}.$$

It is observed that

$$\begin{aligned} lim_{p\to 0}T(a,b,p) &= ab \\ lim_{p\to\infty}T(a,b,p) &= min\{a,b\}, \end{aligned} \tag{9.2}$$

which correspond to two of the most frequently used T-norms in combining the membership values on the premise part of a fuzzy if-then rule.

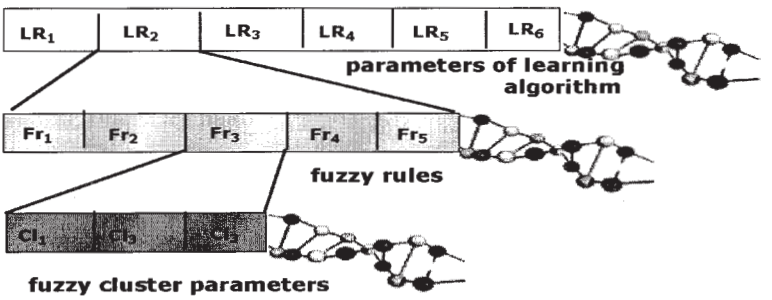

Fig. 9.10. Chromosome structure of the i-Miner

Chromosome modelling and representation. Hierarchical evolutionary search process has to be represented in a chromosome for successful modelling of the i-Miner framework. A typical chromosome of the i-Miner would appear as shown in Fig. 9.10 and the detailed modelling process is as follows.

Layer 1: The optimal number of clusters and initial cluster centers is represented this layer.

Layer 2: This layer is responsible for the optimization of the rule base. This includes deciding the total number of rules, representation of the antecedent and consequent parts. The number of rules grows rapidly with an increasing number of variables and fuzzy sets. We used the grid-partitioning algorithm to generate the initial set of rules [9.3]. An iterative learning method is then adopted to optimize the rules [9.9] . The existing rules are mutated and new rules are introduced. The fitness of a rule is given by its contribution (strength) to the actual output. To represent a single rule a position dependent code with as many elements as the number of variables of the system is used. Each element is a binary string with a bit per fuzzy set in the fuzzy partition of the variable, meaning the absence or presence of the corresponding linguistic label in the rule. For a three input and one output variable, with fuzzy partitions composed of 3,2,2 fuzzy sets for input variables and 3 fuzzy sets for output variable, the fuzzy rule will have a representation as shown in Fig. 9.5.

Layer 3: This layer is responsible for the selection of optimal learning parameters. Performance of the gradient descent algorithm directly depends on the learning rate according to the error surface. The optimal learning parameters decided by this layer will be used to tune the parameterized rule antecedents/consequents and the fuzzy operators. The rule antecedent/consequent parameters and the fuzzy operators are fine tuned using a gradient descent algorithm to minimize the output error

$$E = \sum_{k=1}^{N} (d_k - x_k)^2 \cdot$$

where d_k is the k^{th} component of the r^{th} desired output vector and x_k is the k^{th} component of the actual output vector by presenting the r^{th} input vector to the network. All the gradients of the parameters to be optimized, namely the consequent parameters $\frac{\partial E}{\partial P_n}$ for all rules Rn and the premise parameters $\frac{\partial E}{\partial \sigma_i}$ and $\frac{\partial E}{\partial c_i}$ for all fuzzy sets F_i (σ and c represents the MF width and center of a Gaussian MF).

Once the three layers are represented in a chromosome C, and then the learning procedure could be initiated as follows:

1. Generate an initial population of N numbers of C chromosomes. Evaluate the fitness of each chromosome depending on the output error.

2. Depending on the fitness and using suitable selection methods reproduce a number of children for each individual in the current generation.
3. Apply genetic operators to each child individual generated above and obtain the next generation.
4. Check whether the current model has achieved the required error rate or the specified number of generations has been reached. Go to Step b.
5. End

Experimentation setup, training and performance evaluation. To demonstrate the efficiency of the proposed frameworks, Web access log data at the Monash University's Web site [9.17] were used for experimentations. We used the statistical/ text data generated by the log file analyzer from 01 January 2002 to 07 July 2002. Selecting useful data is an important task in the data pre-processing block. After some preliminary analysis, we selected the statistical data comprising of domain byte requests, hourly page requests and daily page requests as focus of the cluster models for finding Web users' usage patterns. It is also important to remove irrelevant and noisy data in order to build a precise model. We also included an additional input 'index number' to distinguish the time sequence of the data. The most recently accessed data were indexed higher while the least recently accessed data were placed at the bottom. Besides the inputs 'volume of requests' and 'volume of pages (bytes)' and 'index number', we also used the 'cluster information' provided by the clustering algorithm as an additional input variable. The data was re-indexed based on the cluster information. Our task is to predict (few time steps ahead) the Web traffic volume on a hourly and daily basis. We used the data from 17 February 2002 to 30 June 2002 for training and the data from 01 July 2002 to 06 July 2002 for testing and validation purposes.

Table 9.3. Parameter settings of i-Miner

Population size	30
Maximum no of generations	35
Fuzzy inference system	Takagi Sugeno
Rule antecedent membership functions	3 membership functions per input variable (parameterized Gaussian) linear parameters
Rule consequent parameters	
Gradient descent learning	10 epochs
Ranked based selection	0.50
Elitism	5 %
Starting mutation rate	0.50

The initial populations were randomly created based on the parameters shown in Table 9.1. We used a special mutation operator, which decreases the mutation rate as the algorithm greedily proceeds in the search space [9.9] . If the allelic value x_i of the i-th gene ranges over the domain a_i and b_i the mutated gene is drawn randomly uniformly from the interval $[a_i,b_i]$.

$$x_i = x_i + \triangle(t, b_i - x_i), if\omega = 0$$
$$= x_i + \triangle(t, b_i - a_i), if\omega = 1 \tag{9.3}$$

where ω represents an unbiased coin flip p(ω =0) = p(ω =1) = 0.5, and

$$\triangle(t, x) = x\left(1 - \gamma^{(1-\frac{t}{t_{max}})^b}\right)$$

defines the mutation step, where γ is the random number from the interval [0,1] and t is the current generation and t_{max} is the maximum number of generations. The function computes a value in the range [0,x] such that the probability of returning a number close to zero increases as the algorithm proceeds with the search. The parameter b determines the impact of time on the probability distribution $\triangle$ over [0,x]. Large values of b decrease the likelihood of large mutations in a small number of generations. The parameters mentioned in Table 9.1 were decided after a few trial and error approaches. Experiments were repeated 3 times and the average performance measures are reported. Figs. 9.11 and 9.12 illustrates the meta-learning approach combining evolutionary learning and gradient descent technique during the 35 generations.

Table 9.4 summarizes the performance of the developed i-Miner for training and test data. Performance is compared with the previous results [23] wherein the trends were analyzed using a Takagi-Sugeno Fuzzy Inference System (ANFIS) learned using neural network learning techniques and Linear Genetic Programming (LGP). The Correlation Coefficient (CC) for the test data set is also given in Table 9.4.

Figs. 9.13 and 9.14 illustrate the actual and predicted trends for the test data set. FCM approach created 9 data clusters for daily traffic according to the input features compared to 7 data clusters (Fig. 9.15) for the hourly requests.

Table 9.4. Performance of the different paradigms

Method	Period					
	Daily (1 day ahead)			Hourly (1 hour ahead)		
	RMSE		CC	RMSE		CC
	Train	Test		Train	Test	
i-Miner	0.0044	0.0053	0.9967	0.0012	0.0041	0.9981
TKFIS	0.0176	0.0402	0.9953	0.0433	0.0433	0.9841
LGP	0.0543	0.0749	0.9315	0.0654	0.0516	0.9446

The 35 generations of meta-learning approach created 62 if-then Takagi-Sugeno type fuzzy rules (daily traffic trends) and 64 rules (hourly traffic

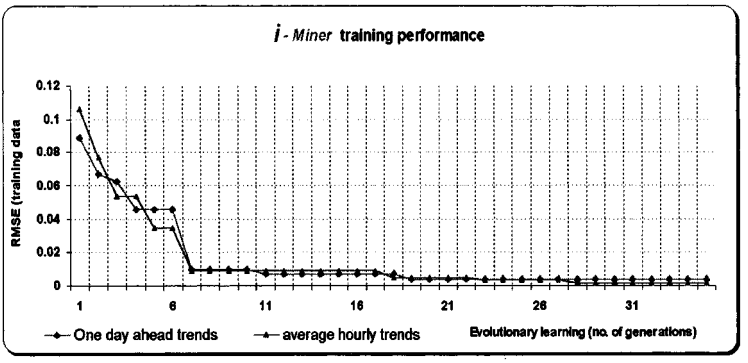

Fig. 9.11. Meta-learning performance (training) of i-Miner

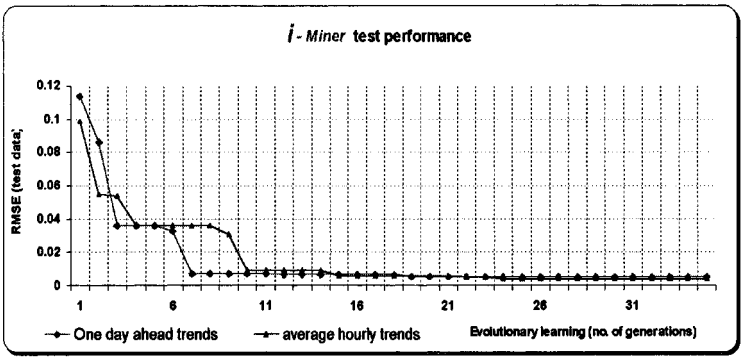

Fig. 9.12. Meta-learning performance (testing) of i-Miner

trends). Fig. 9.16 depicts the hourly visitor information according to domain names from an FCM cluster. Fig. 9.17 illustrates the volume of visitors in each FCM cluster according to the day of access.

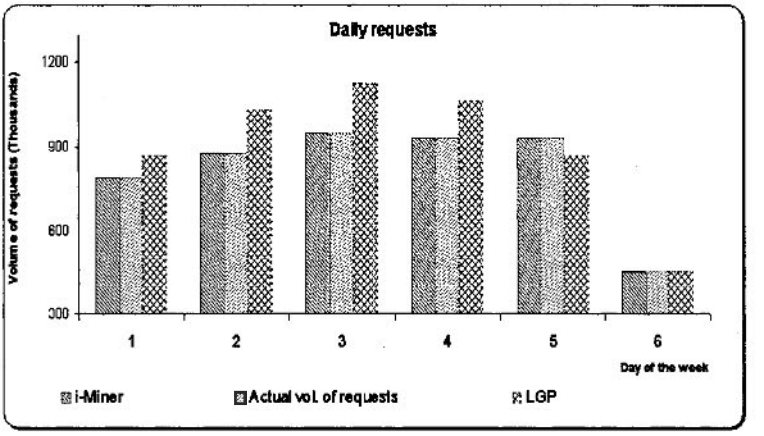

Fig. 9.13. Test results of the daily trends for 6 days

9.2.3 Discussions

Recently Web usage mining has been gaining a lot of attention because of its potential commercial benefits. Empirical results show that the proposed i-Miner framework seems to work very well for the problem considered. i-Miner framework gave the overall best results with the lowest RMSE on test error and the highest correlation coefficient. An important disadvantage of i-Miner is the computational complexity of the algorithm. When optimal performance is required (in terms of accuracy and smaller structure) such algorithms might prove to be useful as evident from the empirical results. In i-Miner evolutionary algorithm was used to optimize the various clustering and fuzzy inference system parameters. It is interesting to note that even LGP as a function approximator could pick up the trends accurately.

9.3 Conclusions

In this chapter, we have illustrated the importance of evolutionary algorithms for the two network management related problems. For real time intrusion detection systems LGP would be the ideal candidate as it could be manipulated at machine code level. Experiments using the Web data has revealed

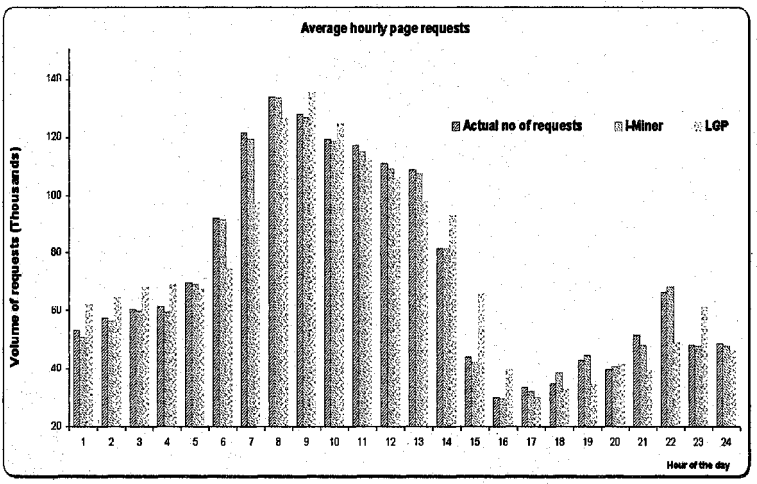

Fig. 9.14. Test results of the average hourly trends for 6 days

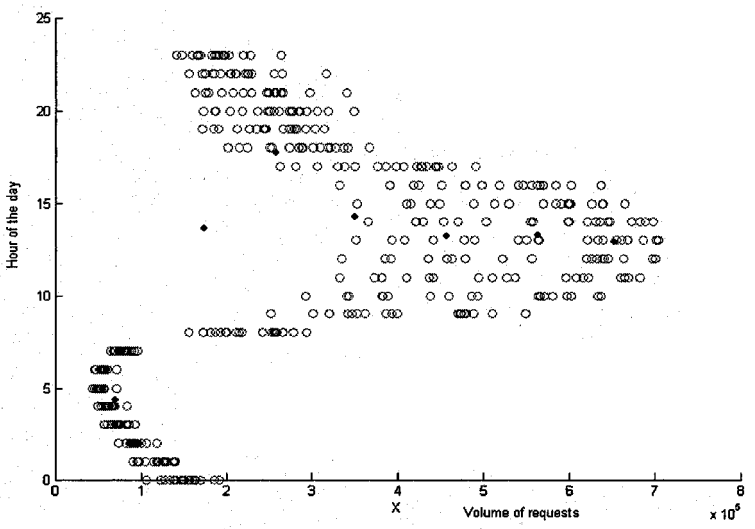

Fig. 9.15. Evolutionary FCM clustering: hour of the day and volume of requests. The dark dots indicate the cluster centers

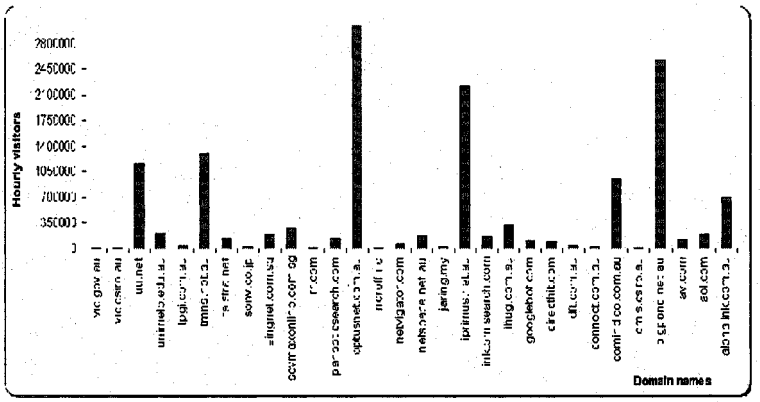

Fig. 9.16. Hourly visitor information according to the domain names from an FCM cluster

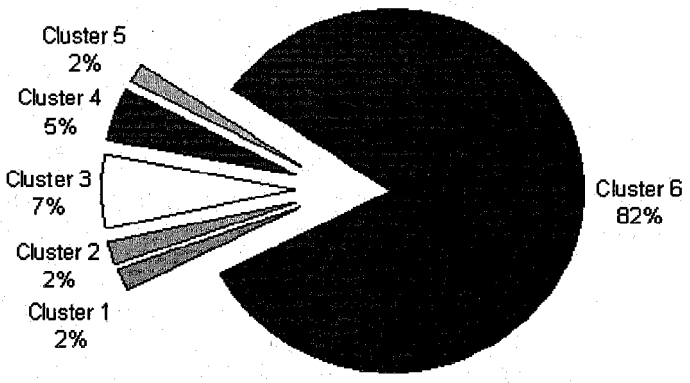

Fig. 9.17. Clustering of visitors based on the day of access from an FCM cluster

the importance of the optimization of fuzzy clustering algorithm and fuzzy inference system. Among the various trend analysis algorithms considered, LGP has again shown the capability as a robust function approximator.

Acknowledgements

Author wishes to thank Ms. Sandhya Peddabachigari (Oklahoma State University, USA), Mr. Vivek Gupta (IIT Bombay), Mr. Srinivas Mukkamala (New Mexico Tech, USA) and Ms. Xiaozhe Wang (Monash University, Australia) for all the valuable contributions during the different stages of this research.

References

9.1 Abraham, A., Ramos, V. (2003): Web usage mining using artificial ant colony clustering and genetic programming, 2003 IEEE Congress on Evolutionary Computation (CEC2003), Australia, IEEE Press, 1384-1391

9.2 Abraham, A. (2003): Business intelligence from web usage mining, Journal of Information & Knowledge Management (JIKM), World Scientific Publishing Co., Singapore, **2**

9.3 Abraham, A. (2002): EvoNF: A framework for optimization of fuzzy inference systems using neural network learning and evolutionary computation, In Proceedings of 17th IEEE International Symposium on Intelligent Control, IEEE Press, 327-332

9.4 Abraham, A. (2003): i-Miner: a web usage mining framework using hierarchical intelligent systems, The IEEE International Conference on Fuzzy Systems FUZZ-IEEE'03, IEEE Press, 1129-1134

9.5 Barbara, D., Couto, J., Jajodia, S., Wu, N. (2001): ADAM: a testbed for exploring the use of data mining in intrusion detection. SIGMOD Record, **30**, 15-24

9.6 Bezdek, J. C. (1981): Pattern recognition with fuzzy objective function algorithms, New York: Plenum Press.

9.7 Brameier, M., Banzhaf, W. (2001): A comparison of linear genetic programming and neural networks in medical data mining. Evolutionary Computation," IEEE Transactions on, **5**, 17-26

9.8 Cohen, W. (199): Learning trees and rules with set-valued features, American Association for Artificial Intelligence (AAAI).

9.9 Cordón, O., Herrera, F., Hoffmann, F., Magdalena, L. (2001): Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases, World Scientific Publishing Company, Singapore.

9.10 Denning, D. (1987): An intrusion-detection model, IEEE Transactions on Software Engineering, **13**, 222-232

9.11 Hall, L.O., Ozyurt, I.B., Bezdek J. C. (1999): Clustering with a genetically optimized approach, IEEE Transactions on Evolutionary Computation, **3**, 103-112

9.12 KDD cup (1999): intrusion detection data set: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz>

9.13 Brieman, L., Friedman, J., Olshen, R., Stone, C. (198): Classification of regression trees. Wadsworth Inc.
9.14 Lee, W., Stolfo, S., Mok, K. (1999): A data mining framework for building intrusion detection models. In Proceedings of the IEEE Symposium on Security and Privacy.
9.15 Luo, J., Bridges, S. M. (2000): Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection. International Journal of Intelligent Systems, John Wiley & Sons, **15**, 687-704
9.16 MIT Lincoln Laboratory.
<http://www.ll.mit.edu/IST/ideval/>
9.17 Monash University Web site:
<http://www.monash.edu.au>
9.18 Mukkamala, S., Sung, A. H., Abraham A. (2003): Intrusion detection using ensemble of soft computing paradigms. Third International Conference on Intelligent Systems Design and Applications, Intelligent Systems Design and Applications, Advances in Soft Computing, Springer Verlag, Germany, 239-248
9.19 Peddabachigari, S., Abraham, A., Thomas, J. (2003): Intrusion detection systems using decision trees and support vector machines, International Journal of Applied Science and Computations, USA
9.20 Srivastava, J., Cooley, R., Deshpande, M., Tan, P. N. (2000): Web usage mining: discovery and applications of usage patterns from web data, SIGKDD Explorations, **1**, 12-23
9.21 Summers, R. C. (1997): Secure computing: threats and safeguards. McGraw Hill, New York
9.22 Vapnik, V. N. (2002): The nature of statistical learning theory. Springer in Wang, X., Abraham, A., Smith, K. A. Soft computing paradigms for web access pattern analysis. Proceedings of the 1st International Conference on Fuzzy Systems and Knowledge Discovery, 631-635