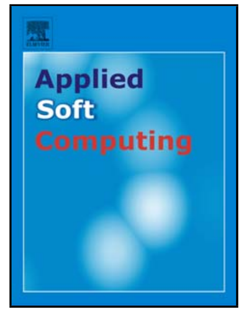
Accepted Manuscript



Title: The Use of Computational Intelligence in Intrusion Detection Systems: A Review

Authors: Shelly Xiaonan Wu, Wolfgang Banzhaf

PII:	S1568-4946(09)00090-8
DOI:	doi:10.1016/j.asoc.2009.06.019
Reference:	ASOC 626
To appear in:	*Applied Soft Computing*
Received date:	2-5-2008
Revised date:	2-6-2009
Accepted date:	28-6-2009

Please cite this article as: S.X. Wu, W. Banzhaf, The Use of Computational Intelligence in Intrusion Detection Systems: A Review, *Applied Soft Computing Journal* (2008), doi:10.1016/j.asoc.2009.06.019

The Use of Computational Intelligence in Intrusion Detection Systems: A Review

Shelly Xiaonan Wu * and Wolfgang Banzhaf
Computer Science Department, Memorial University of Newfoundland, St John's, NL A1B 3X5, Canada

Abstract

Intrusion detection based upon computational intelligence is currently attracting considerable interest from the research community. Characteristics of computational intelligence (CI) systems, such as adaptation, fault tolerance, high computational speed and error resilience in the face of noisy information, fit the requirements of building a good intrusion detection model. Here we want to provide an overview of the research progress in applying CI methods to the problem of intrusion detection. The scope of this review will encompass core methods of CI, including artificial neural networks, fuzzy systems, evolutionary computation, artificial immune systems, swarm intelligence, and soft computing. The research contributions in each field are systematically summarized and compared, allowing us to clearly define existing research challenges, and to highlight promising new research directions. The findings of this review should provide useful insights into the current IDS literature and be a good source for anyone who is interested in the application of CI approaches to IDSs or related fields.

Key words: Survey, Intrusion detection, Computational intelligence, Artificial neural networks, Fuzzy systems, Evolutionary computation, Artificial immune systems, Swarm intelligence, Soft computing

1. Introduction

Traditional intrusion prevention techniques, such as firewalls, access control or encryption, have failed to fully protect networks and systems from increasingly sophisticated attacks and malwares. As a result, intrusion detection systems (IDS) have become an indispensable component of security infrastructure to detect these threats before they inflict widespread damage.

When building an IDS one needs to consider many issues, such as data collection, data pre-processing, intrusion recognition, reporting, and response. Among them, intrusion recognition is most vital. Audit data is compared with detection models, which describe the patterns of intrusive or benign behavior, so that both successful and unsuccessful intrusion attempts can be identified.

Since Denning first proposed an intrusion detection model in 1987 [80], many research efforts have been focused on how to effectively and accurately construct detection models. Between the late 1980s and the early 1990s, a combination of expert systems and statistical approaches was very popular. Detection models were derived from the domain knowledge of security experts. From the mid-1990s to the late 1990s, acquiring knowledge of normal or abnormal behavior had turned from manual to automatic. Artificial intelligence and machine learning techniques were used to discover the underlying models from a set of training data. Commonly used methods were rule based induction, classification and data clustering.

The process of automatically constructing models from data is not trivial, especially for intrusion detection problems. This is because intrusion detection faces problems such as huge network traffic volumes, highly imbalanced data distribution, the difficulty to realize decision boundaries between normal and abnormal behavior, and a requirement for continuous adaptation to a constantly changing environment. Artificial intelligence and machine learning have shown limitations in achieving high detection accuracy and fast processing times when confronted with these requirements. For example, the detection model in the winning entry of the KDD99 competition was composed of 50 $\times$ 10 C5 decision trees. The second-placed entry consisted of a decision forest with 755 trees [92]. Fortunately, computational intelligence techniques, known for their ability to adapt and to exhibit fault tolerance, high computational

* Corresponding author. Tel.: +1 709 737 6947; fax: +1 709 737 2009.
Email addresses: xiaonan@cs.mun.ca (Shelly Xiaonan Wu), banzhaf@cs.mun.ca (Wolfgang Banzhaf).

speed and resilience against noisy information, compensate for the limitations of these two approaches.

The aim of this review is twofold: the first is to present a comprehensive survey on research contributions that investigate utilization of computational intelligence (CI) methods in building intrusion detection models; the second aim is to define existing research challenges, and to highlight promising new research directions. The scope of the survey is the core methods of CI, which encompass artificial neural networks, fuzzy sets, evolutionary computation methods, artificial immune systems, swarm intelligence and soft computing. Soft computing, unlike the rest of the methods, has the synergistic power to intertwine the pros of these methods in such a way that their cons will be compensated. Therefore, it is an indispensable component in CI.

The remainder of this review is organized as follows. Section 2 defines IDSs and computation intelligence. Section 3 introduces commonly used datasets and performance evaluation measures, with the purpose of removing the confusion found in some research work. Section 4 categorizes, compares and summarizes core methods in CI that have been proposed to solve intrusion detection problems. Section 5 compares the strengths and limitations of these approaches, and identifies future research trends and challenges. Section 6 concludes.

2. Background

2.1. *Intrusion Detection*

An intrusion detection system dynamically monitors the events taking place in a system, and decides whether these events are symptomatic of an attack or constitute a legitimate use of the system [77]. Figure 1 depicts the organization of an IDS where solid lines indicate data/control flow, while dashed lines indicate responses to intrusive activities.

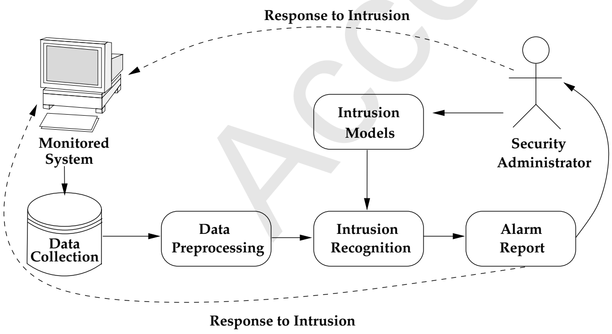


Fig. 1. Organization of a generalized intrusion detection system

In general, IDSs fall into two categories according to the detection methods they employ, namely (i) misuse detection and (ii) anomaly detection. Misuse detection identifies intrusions by matching observed data with pre-defined descriptions of intrusive behavior. Therefore, well-known intrusions can be detected efficiently with a very low false alarm rate. For this reason, the approach is widely adopted in the majority of commercial systems. However, intrusions are usually polymorph, and evolve continuously. Misuse detection will fail easily when facing unknown intrusions. One way to address this problem is to regularly update the knowledge base, either manually which is time consuming and laborious, or automatically with the help of supervised learning algorithms. Unfortunately, datasets for this purpose are usually expensive to prepare, as they require labeling of each instance in the dataset as normal or a type of intrusion. Another way to address this problem is to follow the anomaly detection model proposed by Denning [80].

Anomaly detection is orthogonal to misuse detection. It hypothesizes that abnormal behavior is rare and different from normal behavior. Hence, it builds models for normal behavior and detects anomaly in observed data by noticing deviations from these models. There are two types of anomaly detection [54]. The first is static anomaly detection, which assumes that the behavior of monitored targets never changes, such as system call sequences of an Apache service. The second type is dynamic anomaly detection. It extracts patterns from behavioral habits of end users, or usage history of networks/hosts. Sometimes these patterns are called profiles.

Clearly, anomaly detection has the capability of detecting new types of intrusions, and only requires normal data when building profiles. However, its major difficulty lies in discovering boundaries between normal and abnormal behavior, due to the deficiency of abnormal samples in the training phase. Another difficulty is to adapt to constantly changing normal behavior, especially for dynamic anomaly detection.

In addition to the detection method, there are other characteristics one can use to classify IDSs, as shown in Figure 2.

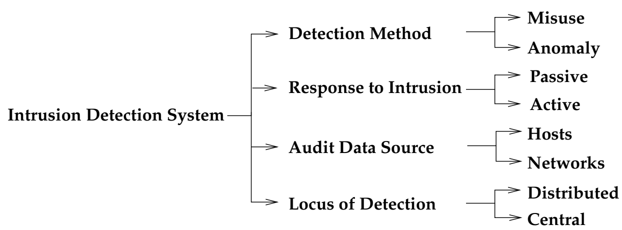


Fig. 2. Characteristics of intrusion detection systems

2.2. *Computational Intelligence*

Computational Intelligence (CI) is a fairly new research field with competing definitions. For example, in Computational Intelligence - A Logical Approach [241], the authors defined CI as:

> Computational Intelligence is the study of the design of intelligent agents. ... An intelligent agent is a system that acts intelligently: What it does is appropriate for its circumstances and its goal, it is flexible to changing environments and changing goals, it learns from experi-

ence, and it makes appropriate choices given perceptual limitations and finite computation.

In contrast, Bezdek [39] defined CI as:

A system is computational intelligent when it: deals with only numerical (low-level) data, has pattern recognition components, does not use knowledge in the artificial intelligence sense; and additionally when it (begins to) exhibit i) computational adaptivity, ii) computational fault tolerance, iii) speed approaching human-like turnaround, and iv) error rates that approximate human performance.

The discussion in [63] and [89] further confirm the characteristics of computational intelligence systems summarized by Bezdek's definition. Therefore, in this review, we subscribe to Bezdek's definition.

CI is different from the well-known field of Artificial Intelligence (AI). AI handles symbolic knowledge representation, while CI handles numeric representation of information; AI concerns itself with high-level cognitive functions, while CI is concerned with low-level cognitive functions. Furthermore, AI analyzes the structure of a given problem and attempts to construct an intelligent system based upon this structure, thus operating in a top-down manner, while the structure is expected to emerge from an unordered beginning in CI, thus operating in a bottom-up manner [63, 89].

Although there is not yet full agreement on what computational intelligence exactly is, there is a widely accepted view on which areas belong to CI: artificial neural networks, fuzzy sets, evolutionary computation, artificial immune systems, swarm intelligence, and soft computing. These approaches, except for fuzzy sets, are capable of autonomously acquiring and integrating knowledge, and can be used in either supervised or unsupervised learning mode.

In the intrusion detection field, supervised learning usually produces classifiers for misuse detection from class-labeled training datasets. Classifiers are basically viewed as a function mapping data samples to corresponding class labels. Unsupervised learning distinguishes itself from supervised learning by the fact that no class-labeled data is available in the training phase. It groups data points based upon their similarities. Unsupervised learning satisfies the requirement of anomaly detection, hence it is usually employed in anomaly detection.

3. Datasets and Performance Evaluation

In this section, we will summarize popular benchmark datasets and performance evaluation measures in the intrusion detection domain, with the purpose of clarifying some mistaken terms we found during the review process.

3.1. *Datasets*

Data in the reviewed research work is normally collected from three sources: data packets from networks, command sequences from user input, or low-level system information, such as system call sequences, log files, and CPU/memory usage. We list some commonly used benchmarks in Table 1. All of these datasets have been used in either misuse detection or anomaly detection.

Here, we focus on two benchmarks: The DARPA-Lincoln datasets and the KDD99 datasets. The DARPA-Lincoln datasets were collected by MIT's Lincoln laboratory, under the DARPA ITO and Air Force Research Laboratory sponsorship, with the purpose of evaluating the performance of different intrusion detection methodologies. The datasets, collected in 1998, contain seven weeks of training data and two weeks of test data. The attack data included more than 300 instances of 38 different attacks launched against victim UNIX hosts, falling into one of the four categories: Denial of Service (DoS), Probe, Users to Root (U2R), and Remote to Local (R2L). For each week, inside and outside network traffic data, audit data recorded by the Basic Security Module (BSM) on Solaris hosts, and file system dumped from UNIX hosts were collected. In 1999, another series of datasets was collected, which included three weeks of training and two weeks of test data. More than 200 instances of 58 attack types were launched against victim UNIX and Windows NT hosts and a Cisco router. In 2000, three additional scenario-specific datasets were generated to address distributed DoS and Windows NT attacks. Detailed descriptions of these datasets can be found at [2].

The KDD99 dataset was derived in 1999 from the DARPA98 network traffic dataset by assembling individual TCP packets into TCP connections. It was the benchmark dataset used in the International Knowledge Discovery and Data Mining Tools Competition, and also the most popular dataset that has ever been used in the intrusion detection field. Each TCP connection has 41 features with a label which specifies the status of a connection as either being normal, or a specific attack type [4]. There are 38 numeric features and 3 symbolic features, falling into the following four categories:

(i) Basic Features: 9 basic features were used to describe each individual TCP connection.
(ii) Content Features: 13 domain knowledge related features were used to indicate suspicious behavior having no sequential patterns in the network traffic.
(iii) Time-based Traffic Features: 9 features were used to summarize the connections in the past two seconds that had the same destination host or the same service as the current connection.
(iv) Host-based Traffic Features: 10 features were constructed using a window of 100 connections to the same host instead of a time window, because slow scan attacks may occupy a much larger time interval than two seconds.

The training set contains 4,940,000 data instances, covering normal network traffic and 24 attacks. The test set contains 311,029 data instances with a total of 38 attacks, 14 of which do not appear in the training set. Since the training set is prohibitively large, another training set which contains 10% of the data is frequently used.

McHugh [219] published an in-depth critical assessment

Table 1
Summary of Popular Datasets in the Intrusion Detection Domain

Data Source	Dataset Name	Abbreviation
Network Traffic	DARPA 1998 TCPDump Files [2]	DARPA98
	DARPA 1999 TCPDump Files [2]	DARPA99
	KDD99 Dataset [4]	KDD99
	10% KDD99 Dataset [4]	KDD99-10
	Internet Exploration Shootout Dataset [3]	IES
User Behavior	UNIX User Dataset [6]	UNIXDS
System Call Sequences	DARPA 1998 BSM Files [2]	BSM98
	DARPA 1999 BSM Files [2]	BSM99
	University of New Mexico Dataset [5]	UNM

of the DARPA datasets, arguing that some methodologies used in the evaluation are questionable and may have biased the results. For example, normal and attack data have unrealistic data rates; training datasets for anomaly detection are not adequate for its intended purpose; no efforts have been made to validate that false alarm behavior of IDSs under test shows no significantly difference on real and synthetic data. Malhony and Chan [215] confirmed McHugh's findings by experiments, which discovered that many attributes had small and fixed ranges in simulation, but large and growing ranges in real traffic.

By sharing the same root with the DARPA datasets, the KDD99 dataset inherits the above limitations. In addition, the empirical study conducted by Sabhnani *et al.* [246] states that "the KDD training and test data subsets represent dissimilar target hypotheses for U2R and R2L attack categories". According to their analysis, 4 new attacks constitute 80% of U2R data, and 7 new attacks constitute more than 60% of R2L data in the test dataset. This may well explain why the detection results for U2R and R2L attacks are not satisfactory in most IDSs.

Despite all this criticism, however, both the DARPA-Lincoln and the KDD99 datasets continue to be the largest publicly available and the most sophisticated benchmarks for researchers in evaluating intrusion detection algorithms or machine learning algorithms.

Instead of using benchmarks listed in Table 1, sometimes researchers prefer to generate their own datasets. However, in a real network environment, it is hard to guarantee that supposedly normal data are indeed intrusion free. The robust approach introduced by Rhodes *et al.* [244] is able to remove anomalies from collected training data. A further reason for using self-produced datasets is incomplete training datasets, which tend to decrease the accuracy of IDSs. Therefore, artificial data is generated and merged within training sets [21, 95, 116, 128, 144, 264].

3.2. *Performance Evaluation*

The effectiveness of an IDS is evaluated by its ability to make correct predictions. According to the real nature of a given event compared to the prediction from the IDS, four possible outcomes are shown in Table 2, known as the confusion matrix. True negatives as well as true positives correspond to a correct operation of the IDS; that is, events are successfully labeled as normal and attacks, respectively. False positives refer to normal events being predicted as attacks; false negatives are attack events incorrectly predicted as normal events.

Table 2
Confusion Matrix

		Predicted Class	
		Negative Class (Normal)	Positive Class (Attack)
Actual Class	Negative Class (Normal)	True Negative (TN)	False Positive (FP)
	Positive Class (Attack)	False Negative (FN)	True Positive (TP)

Based on the above confusion matrix, a numerical evaluation can apply the following measures to quantify the performance of IDSs:

- True Negative Rate (TNR): $\frac{TN}{TN+FP}$, also known as Specificity.
- True Positive Rate (TPR): $\frac{TP}{TP+FN}$, also known as Detection Rate (DR) or Sensitivity. In information retrieval, this is called Recall.
- False Positive Rate (FPR): $\frac{FP}{TN+FP} = 1 - specificity$, also known as False Alarm Rate (FAR).
- False Negative Rate (FNR): $\frac{FN}{TP+FN} = 1 - sensitivity$.
- Accuracy: $\frac{TN+TP}{TN+TP+FN+FP}$
- Precision: $\frac{TP}{TP+FP}$, which is another information retrieval term, and often is paired with "Recall".

The most popular performance metrics are detection rate (DR) together with false alarm rate (FAR). An IDS should have a high DR and a low FAR. Other commonly used combinations include Precision and Recall, or Sensitivity and Specificity.

4. Algorithms

In this section, we will review the core computational intelligence approaches that have been proposed to solve intrusion detection problems. We shall discuss artificial neural networks, fuzzy sets, evolutionary computation, artificial immune systems, swarm intelligence and soft computing.

4.1. *Artificial Neural Networks*

An Artificial Neural Network (ANN) consists of a collection of processing units called neurons that are highly interconnected in a given topology. ANNs have the ability of learning-by-example and generalizion from limited, noisy, and incomplete data; they have, hence, been successfully employed in a broad spectrum of data-intensive applications. In this section, we will review their contributions to and performance in the intrusion detection domain. This section is organized by the types of ANNs as illustrated in Figure 3.

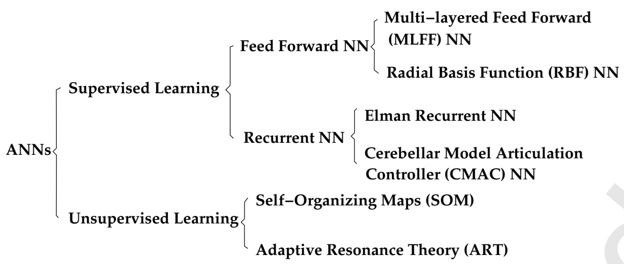


Fig. 3. Types of ANNs reviewed in this section.

4.1.1. *Supervised Learning*

4.1.1.1. *Feed Forward Neural Networks* Feed forward neural networks are the first and arguably the simplest type of artificial neural networks devised. Two types of feed forward neural networks are commonly used in modeling either normal or intrusive patterns.

Multi-layered Feed Forward (MLFF) Neural Networks MLFF networks use various learning techniques, the most popular being back-propagation (MLFF-BP). In early development of IDSs, MLFF-BP networks were applied primarily to anomaly detection on user behavior level, e.g. [264] and [245]. [264] used information, such as command sets, CPU usage, login host addresses, to distinguish between normal and abnormal behavior, while [245] considered the patterns of commands and their frequency.

Later, research interests shifted from user behavior to software behavior described by sequences of system calls. This is because system call sequences are more stable than commands. Ghosh *et al.* built a model by MLFF-BP for the *lpr* program [116] and the DARPA BSM98 dataset [115], respectively. A leaky bucket algorithm was used to remember anomalous events diagnosed by the network, so that the temporal characteristics of program patterns were accurately captured.

Network traffic is another indispensable data source. Canady *et al.* [46] applied MLFF-BP on 10,000 network packets collected from a simulated network environment for misuse detection purposes. Although the training/test iterations required 26.13 hours to complete, their experiments showed the potential of MLFF-BP as a binary classifier to correctly identify each of the embedded attacks in the test data. MLFF-BP can also be used as a multi-class classifier (MCC). Such neural networks either have multiple output neurons [226] or assemble multiple binary neural network classifiers together [294]. Apparently, the latter is more flexible than the former when facing a new class.

Except for the BP learning algorithm, there are many other learning options for MLFF networks. [227] compared 12 different learning algorithms on the KDD99 dataset, and found that resilient back propagation achieved the best performance in terms of accuracy (97.04%) and training time (67 epochs).

Radial Basis Function Neural Networks Radial Basis Function (RBF) neural networks are another popular type of feed forward neural networks. Since they perform classification by measuring distances between inputs and the centers of the RBF hidden neurons, RBF networks are much faster than time consuming back-propagation, and more suitable for problems with large sample size [52].

Research, such as [151], [206], [243], [295], employed RBFs to learn multiple local clusters for well-known attacks and for normal events. Other than being a classifier, the RBF network was also used to fuse results from multiple classifiers [52]. It outperformed five different decision fusion functions, such as a Dempster-Shafer combination and Weighted Majority Vote.

Jiang *et al.* [168] reported a novel approach which integrates both misuse and anomaly detections in a hierarchical RBF network. In the first layer, an RBF anomaly detector identifies whether an event is normal or not. Anomaly events then pass an RBF misuse detector chain, with each detector being responsible for a specific type of attack. Anomaly events which could not be classified by any misuse detectors were saved to a database. When enough anomaly events were gathered, a C-Means clustering algorithm clustered these events into different groups; a misuse RBF detector was trained on each group, and added to the misuse detector chain. In this way, all intrusion events were automatically and adaptively detected and labeled.

Comparison between MLFF-BP and RBF networks Since RBF and MLFF-BP networks are widely used, a comparison between them is natural. [168] and [295] compared the RBF and MLFF-BP networks for misuse and anomaly detection on the KDD99 dataset. Their experiments have shown that for misuse detection, BP has a slightly better performance than RBF in terms of detection rate and false positive rate, but requires longer training time. For anomaly detection, the RBF network improves performance with a high detection rate and a low false positive rate, and requires less training time (cutting it down from hours to minutes). All in all, RBF networks achieve better performance. The same conclusion was drawn by Hofmann *et al.* on the DARPA98 dataset [150, 151].

Another interesting comparison has been made between the binary and decimal input encoding schemes for MLFF-BP and RBF [206]. The results show that binary encodings have lower error rates than decimal encodings, because decimal encodings only compute the frequency without considering the order of system calls. However, decimal encodings handle noise better and require less data in training. Furthermore, there are fewer input nodes in decimal encodings than in binary encodings, which decreases the training and test time and simplifies the network structure.

4.1.1.2. *Recurrent Neural Networks* Detecting attacks spread over a period of time, such as slow port scanning attempts, is important but difficult. In order to capture the temporal locality in either normal patterns or anomaly patterns, some researchers used time windows and similar mechanisms [115, 151, 206, 296], or chaotic neurons [288] to provide BP networks with external memory. However, window size should be adjustable in predicting user behavior. When users perform a particular job, their behavior is stable and predictable. At such times a large window size is needed to enhance deterministic behavior; when users are switching from one job to another, behavior becomes unstable and stochastic, so a small window size is needed in order to quickly forget meaningless past events [78]. The incorporation of memory in neural networks has led to the invention of recurrent links, hence the name Recurrent Neural Networks (RNN) or Elman network, as shown in Figure 4.

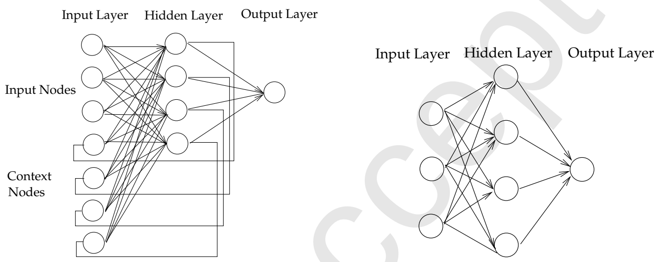


Fig. 4. Compared with MLFF, parts of the output of RNN at time t are inputs in time $t+1$, thus creating internal memories of the neural network.

Recurrent networks were initially used for forecasting, where a network predicted the next event in an input sequence. When there is sufficient deviation between a predicted output and an actual event, an alarm is issued. Debar *et al.* [76, 78] modified the traditional Elman recurrent model by accepting input in both time *t-1* and time *t*. The accuracy of predicting the next command, given a sequence of previous commands, could reach up to 80%. Ghosh *et al.* [114] compared the recurrent network with an MLFF-BP network for forecasting system call sequences. The results showed that recurrent networks achieved the best performance, with a detection accuracy of 77.3% and zero false positives.

Recurrent networks were also trained as classifiers. Cheng *et al.* [57] employed a recurrent network to detect network anomalies in the KDD99 dataset, since network traffic data has the temporal locality property. A Truncated-Back-Propagation-Through-Time learning algorithm was chosen to accelerate training speed. The authors argued for the importance of payload information in network packets. Retaining the information in the packet header but discarding the payload leads to an unacceptable information loss. Their experiment indicated that an Elman network with payload information outperformed an Elman network without such information. Al-Subaie *et al.* [21] built a classifier with an Elman network for the UNM system calls dataset. Their paper is a good source on the comparison of Elman and MLFF networks in terms of network structure, computational complexity, and classification performance. Both works confirm that recurrent networks outperform MLFF networks in detection accuracy and generalization capability. Al-Subaie *et al.*, in addition, point out a performance overhead being associated with the training and operation of recurrent networks.

The Cerebellar Model Articulation Controller (CMAC) neural network is another type of recurrent network, which has the capability for incremental learning. It avoids retraining a neural network every time when a new intrusion appears. This is the main reason why Cannady [47, 48] applied CMAC to autonomously learning new attacks. The author modified a traditional CMAC network by adding feedback from the environment. This feedback would be any system status indicators, such as CPU load or available memory. A modified least mean square learning algorithm was adopted. A series of experiments demonstrated that CMAC effectively learned new attacks, in real time, based on the feedback from the protected system, and generalized well to similar attack patterns.

4.1.2. *Unsupervised Learning*

Self-Organizing Maps and Adaptive Resonance Theory are two typical unsupervised neural networks. Similar to statistical clustering algorithms, they group objects by similarity. They are suitable for intrusion detection tasks in that normal behavior is densely populated around one or two centers, while abnormal behavior and intrusions appear in sparse regions of the pattern space outside of normal clusters.

4.1.2.1. *Self-Organizing Maps* Self-organizing maps (SOM), also known as Kohonen maps, are single-layer feed forward networks where outputs are clustered in a low dimensional (usually 2D or 3D) grid [186]. It preserves topological relationships of input data according to their similarity.

SOMs are the most popular neural networks to be trained for anomaly detection tasks. For example, Fox *et al.* first employed SOMs to detect viruses in a multiuser machine in 1990 [110]. Later, other researchers [154, 277] used SOMs to learn patterns of normal system activities. Nevertheless, SOMs have been used in the misuse detection as well, where a SOM functioned as a data pre-processor to cluster input data. Other classification algorithms, such as feed forward neural networks, were then trained on the output from the SOM [40, 49, 169].

Sometimes, SOMs map data from different classes into one neuron. Therefore, in order to solve the ambiguities in these heterogeneous neurons, Sarasamma *et al.* [247] suggested to calculate the probability of a record mapped to a heterogeneous neuron being of a type of attack. A confidence factor was defined to determine the type of record that dominated the neuron.

Rhodes *et al.* [244], after examining network packets carefully, stated that every network protocol layer has a unique structure and function, so malicious activities aiming at a specific protocol should be unique too. It is unrealistic to build a single SOM to tackle all these activities. Therefore, they organized a multilayer SOM, each layer corresponding to one protocol layer. Sarasamma *et al.* [247] drew similar conclusions that different subsets of features were good at detecting different attacks. Hence, they grouped the 41 features of the KDD99 dataset into 3 subsets. A three-layer SOM model was built, accepting one subset of features and heterogeneous neurons from the previous SOM layer. Results showed that false positive rates were significantly reduced in hierarchical SOMs compared to single layer SOMs on all test cases.

Lichodzijewski *et al.* employed a two-layer SOM to detect anomalous user behavior [202] and anomalous network traffic [201]. The first layer comprised 6 parallel SOMs, each map clustering one feature. The SOM in the second layer combined the results from the first layer SOMs to provide an integrated view. Kayacik *et al.* [170, 172, 173] extended Lichodzijewski's work by introducing a third SOM layer, while keeping the first two layers unchanged. The SOM in the third layer was intended to resolve the confusion caused by heterogeneous neurons. In both Kayacik *et al.*'s and Lichodzijewski *et al.*'s work, a Potential Function Clustering method was used between the first and second layer. This clustering algorithm significantly reduced the dimensions seen by neurons in the second layer. When comparing their results with the best supervised learning solutions, because suitable boosting algorithms are not available for unsupervised learning, their methods showed a similar detection rate but a higher FP rate.

Zanero [290, 292] was another proponent of the analysis of payload of network packets. He proposed a multi-layer detection framework, where the first layer used a SOM to cluster the payload, effectively compressing it into a single feature. This compressed payload feature was then passed on to the second layer as input, together with other features in packet headers. Many classification algorithms can be used in the second tier. Unfortunately, the high dimensionality of (from 0 to 1460 bytes) payload data greatly decreased the performance of the first layer. Zanero later conceived the *K-means+* [291] algorithm to avoid calculating the distance between each neuron, thus greatly improving the runtime efficiency of the algorithm.

Unlike other unsupervised approaches, SOMs can be used to visualize the analysis. Girardin introduced a visual approach for analyzing network activities [118], which best took advantage of topology-preserving and dimensionality-reducing properties of SOMs. Network events are projected onto a two dimensional grid of neurons, and then each neuron is portrayed as a square within the grid. The foreground color of the square indicates the weights of each neuron. Thus similar network events have similar foreground color, and are grouped together closely. The background color indicates the quality of the mapping. The size of the square identifies the number of events mapped to the unit. Users can, therefore, easily identify rare and abnormal events in the graph, which facilitates exploring and analyzing anomaly events.

If we are to use a SOM to visualize the structural features of the data space, SOMs discussed in the previous work would be inappropriate, because they contain only small numbers of neurons, which prohibits the emergence of intrinsic structural features on the map. Emergent SOMs (ESOM), based on simple SOMs, contain thousands or tens of thousands of neurons, which are necessary to achieve emergence, observe overall structures and disregard elementary details. An ESOM with U-Matrix was employed in [222, 223, 224], focusing on the detection of DoS attacks in the KDD99 dataset. Although their work showed very high accuracy (between 98.3% to 99.81%) and a low false alarm rate (between 2.9% to 0.1%), the training procedure required a large computational overhead, especially with training sets of size over 10,000.

4.1.2.2. *Adaptive Resonance Theory (ART)* The Adaptive Resonance Theory (ART) embraces a series of neural network models that perform unsupervised or supervised learning, pattern recognition, and prediction. Unsupervised learning models include ART-1, ART-2, ART-3, and Fuzzy ART. Various supervised networks are named with the suffix "MAP", such as ARTMAP, Fuzzy ARTMAP, and Gaussian ARTMAP. Compared with SOMs who cluster data objects based on the absolute distance, ARTs cluster objects based on the relative similarity of input patterns to the weight vector.

Amini *et al.* compared the performance of ART-1 (accepting binary inputs) and ART-2 (accepting continuous inputs) on KDD99 data in [23]. They concluded that ART-1 has a higher detection rate than ART-2, while ART-2 is 7 to 8 times faster than ART-1. This observation is consistent with results obtained in [206]. Later, Amini *et al.* [24] further conducted research on self-generated network traffic. This time they compared the performance of ARTs and SOMs. The results showed that ART nets have better intrusion detection performance than SOMs on either offline or online data.

Fuzzy ART nets combine fuzzy set theory and adaptive resonance theory. This combination is faster and more stable than ART nets alone in responding to arbitrary input sequences. The works of Liao *et al.* [199] and Durgin *et al.* [90] are two examples of using Fuzzy ART to detect anomalies. Liao *et al.* deployed Fuzzy ART in an adaptive learning framework which is suitable for dynamic changing environments. Normal behavior changes are efficiently accommodated while anomalous activities can still be identified. Durgin *et al.* observed that both SOMs and Fuzzy ARTs showed promising results in detecting network abnormal behavior, but the sensitivity of Fuzzy ARTs seems to be much higher than that of SOMs.

4.1.3. *Summary*

In this section, we reviewed research contributions on artificial neural networks in intrusion detection. Various supervised and unsupervised ANNs were employed in misuse and anomaly detection tasks. These research works took advantage of ANNs' ability to generalize from limited, noisy, and incomplete data. Some researchers also attempted to address disadvantages of ANNs. For example, [57, 226, 290, 295] tried to reduce the long training time; [168, 244, 294] used an ensemble approach to solve the retraining problem of ANNs when facing a new class of data; to address the black box nature of ANNs, [151] extracted attack patterns from the trained ANNs in comprehensible format of if-then rules.

To improve detection accuracy, the following practices have proven useful in ANNs:

- Temporal locality property. Studies [114, 115] have confirmed that the temporal locality property exists in normal as well as in intrusive behavior in the intrusion detection field. Normally, time in ANNs is represented either explicitly or implicitly, but [24] and [202] concluded that explicitly representing time does not accurately identify intrusions. When it comes to implicitly representing time, researchers either adopted neural networks with short-term memory, such as recurrent nets, or mapped temporal patterns to spatial patterns for networks without memory. Most of the research work chose sliding windows, which gather n successive events in one vector and use it as input of ANNs (e.g., [40, 46, 151, 154, 173, 190, 201, 206]). Other mechanisms include the leaky bucket algorithm [115], layer-window statistical preprocessors [296], chaotic neurons [288], and using the time difference between two events [24]. All these results confirm that designing a detection technique that capitalizes on the temporal locality characteristic of data can contribute to better results.
- Network structure. Intrusions are evolving constantly. Sometimes attacks are aiming at a specific protocol, while at other times they are aiming at a specific operating system or application. Therefore it would be unreasonable to expect a single neural network to successfully characterize all such disparate information. Previous research reminds us that networks with ensemble or hierarchical structure achieve better performance than single layer networks, no matter whether learning is supervised or unsupervised ([46, 168, 173, 194, 247, 294]).
- Datasets and features. Neural networks only recognize whatever is fed to them in the form of inputs. Although they have the ability to generalize, they are still unable to recognize some unseen patterns. One cause of this difficulty is incomplete training sets. To address this problem, randomly generated anomalous inputs ([21, 116, 264]) are inserted into the training set with the purpose of exposing the network to more patterns, hence making training sets more complete. Selecting good feature sets is another way to improve performance. [247] identified that different subsets of features are good at detecting certain types of attacks. Kayacik *et al.* [173] conducted a series of experiments on a hierarchical SOM framework with KDD99 data. They found that 6 basic features are sufficient for recognizing a wide range of DoS attacks, while 41 features are necessary to minimize the FP rate. Among the 6 basic features, protocol and service type appear to be the most significant.

4.2. *Fuzzy Sets*

The past decades have witnessed a rapid growth in the number and variety of applications of fuzzy logic. Fuzzy logic, dealing with the vague and imprecise, is appropriate for intrusion detection for two major reasons. First, the intrusion detection problem involves many numeric attributes in collected audit data, and various derived statistical measures. Building models directly on numeric data causes high detection errors. For example, an intrusion that deviates only slightly from a model may not be detected or a small change in normal behavior may cause a false alarm. Second, the security itself includes fuzziness, because the boundary between the normal and abnormal is not well defined. This section will spell out how fuzzy logic can be utilized in intrusion detection models.

4.2.1. *Fuzzy Misuse Detection*

Fuzzy misuse detection uses fuzzy models, such as fuzzy rules or fuzzy classifiers to detect various intrusive behavior. When fuzzy logic was initially introduced to the intrusion detection domain, it was integrated with expert systems. Fuzzy rules substituted ordinary rules so as to map knowledge represented in natural language more accurately to computer languages. Fuzzy rules were created by security experts based on their domain knowledge. For example, the Fuzzy Intrusion Recognition Engine (FIRE) proposed by Dickerson *et al.* used fuzzy rules to detect malicious network activities [86, 87]. Although fuzzy sets and their membership functions were decided by a fuzzy C-means algorithm, hand-encoded rules were the main limitation of this work.

Avoiding hand-encoded fuzzy rules is the a main research topic in fuzzy misuse detection. To generate fuzzy rules, commonly employed methods are based on a histogram of attribute values [14, 15], or based on a partition of overlapping areas [14, 15, 193], or based on fuzzy implication tables [298], or by fuzzy decision trees [203], association rules [91] or SVMs [286]. Due to the rapid development of computational intelligence, approaches with learning and adaptive capabilities have been widely used to automatically construct fuzzy rules. These approaches are artificial neural networks, evolutionary computation, and artificial immune systems. We will investigate them in detail in Section 4.6 on "Soft Computing".

Another application of fuzzy logic is decision fusion, which means that fuzzy logic fuses outputs from different models to prepare a final fuzzy decision. For instance, Cho *et al.* [62] trained multiple HMMs to detect normal behavior sequences. The evaluations from HMMs were sent to the fuzzy inference engine, which gave a fuzzy normal or abnormal result. Similar fuzzy inference systems were used to combine decisions of multiple decision trees [266], multiple neuro-fuzzy classifiers [268], and other models [248].

4.2.2. *Fuzzy Anomaly Detection*

Fuzzy logic plays an important role in anomaly detection, too. Current research interests are to build fuzzy normal behavior profiles with the help of data mining.

Bridges *et al.* suggested the use of fuzzy association rules and fuzzy sequential rules to mine normal patterns from audit data [42, 43]. Their work was an extension of the fuzzy association rule algorithm proposed by Kuok *et al.* [189] and the fuzzy sequential rule algorithm by Mannila and Toivonen [216]. To detect anomalous behavior, fuzzy association rules mined from new audit data were compared with rules mined in the training phase. Hence, a similarity evaluation function was developed to compare two association rules [210, 211]. Florez *et al.* [101] later described an algorithm for computing the similarity between two fuzzy association rules based on prefix trees to achieve better running time and accuracy. El-Semary *et al.* [91] directly compared the test data samples against fuzzy association rules by a fuzzy inference engine.

Fuzzy logic also worked with another popular data mining technique, outlier detection, for anomaly detection. According to the hypothesis of IDSs, malicious behavior is naturally different from normal behavior. Hence, abnormal behavior should be considered as outliers. Fuzzy C-Medoids algorithms [253] and fuzzy C-Means algorithms [58, 59, 60, 148] are two common clustering approaches to identify outliers. Like all clustering techniques, they are affected by the "curse of dimensionality", thus suffering performance degradation when confronted with datasets of high dimensionality. Feature selection is therefore a necessary data pre-processing step. For example, Principal Component Analysis [148, 253] and Rough Sets [58, 59, 60] can be applied on datasets before they are being clustered.

4.2.3. *Summary*

Fuzzy logic, as a means of modeling the uncertainty of natural language, constructs more abstract and flexible patterns for intrusion detection, and thus greatly increases the robustness and adaptation ability of detection systems. Two research directions are currently active in the fuzzy logic area; (i) algorithms with learning and adaptive capabilities are investigated with the purpose of automatically designing fuzzy rules. Popular methods include, but are not limited to, association rules, decision trees, evolutionary computation, and artificial neural networks; (ii) fuzzy logic helps to enhance the understandability and readability of some machine learning algorithms, such as SVMs or HMMs. The use of fuzzy logic smooths the abrupt separation of normality and abnormality. From the research work reviewed in this section, and the work will be mentioned later in the Soft Computing section, the popularity of fuzzy logic clearly demonstrates the successfulness of fuzzy logic in fulfill these two roles. We believe that fuzzy logic will remain an active research topic in the near future.

4.3. *Evolutionary Computation*

Evolutionary Computation (EC), a creative process gleaned from evolution in nature, is capable of addressing real-world problems with great complexity. These problems normally might involve randomness, complex nonlinear dynamics, and multimodal functions, which are difficult to conquer for traditional algorithms [102]. In this section, we will review the role of EC in the intrusion detection field. Some important issues, such as evolutionary operators, niching, and fitness functions will be discussed.

This survey focuses on Genetic Algorithms (GA) [156] and Genetic Programming (GP) [37, 188]. GA and GP differ with respect to several implementation details, with GP working on a superset of representations compared to GAs [37]. Generally speaking, evolution in GAs and GP can be described as a two-step iterative process, consisting of variation and selection, as shown in Figure 5.

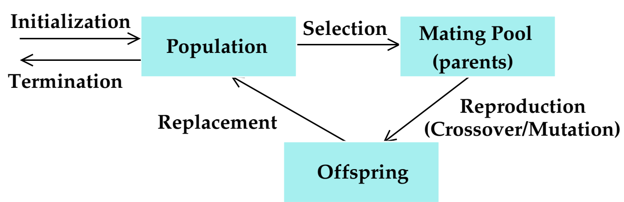


Fig. 5. The flow chart of a typical evolutionary algorithm

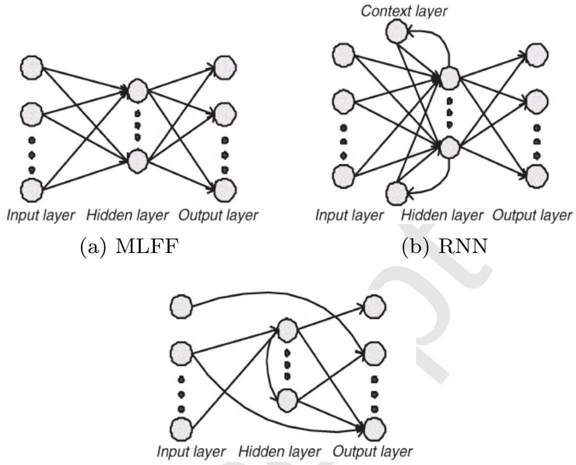
4.3.1. *The Roles of EC in IDS*

EC can be applied on a number of tasks in IDSs. We discuss them in detail below.

4.3.1.1. *Optimization* Some researchers are trying to analyze the problem of intrusion detection by using a multiple fault diagnosis approach, somewhat analogous to the process of a human being diagnosed by a physician when suffering from a disease. For a start, an events-attacks matrix is defined, which is known as pre-learned domain knowledge (analogous to knowledge possessed by a physician). The occurrence of one or more attacks is required to be inferred from newly observed events (analogous to symptoms). Such a problem is reducible to a zero-one integer problem, which is NP-Complete. Dass [70] and Mé [220] both employed GAs as an optimization component. Mé used a standard GA, while Dass used a micro-GA in order to reduce the time overhead normally associated with a GA. Both works coded solutions in binary strings, where the length of a string was the number of attacks, and 1's or 0's in a genome indicated if an attack was present. The fitness function was biased toward individuals able to predict a large number of intrusion types (number of 1's in chromosomes), while avoiding warnings of attacks that did not exist (unnecessary 1's in chromosomes). Diaz-Gomez *et al.* corrected the fitness definition used in [220] after careful analysis [83, 84] and mathematical justification [82], and further refined it in [85].

4.3.1.2. *Automatic Model Structure Design* ANNs and clustering algorithms are two popular techniques to build intrusion detection models. The problematic side of them is that one has to decide on an optimal network structure for the former, and the number of clusters for the latter. To remedy these drawbacks, evolutionary algorithms are introduced for automatic design purpose.

Hofmann *et al.* [151] evolved an RBF neural network to classify network traffic for the DARPA98 dataset. A GA was responsible for learning the structure of RBF nets, such as the type of basis function, the number of hidden neurons, and the number of training epochs. Evolving Fuzzy Neural Network (EFuNN) is another example of this kind. It implemented a Mamdani-type fuzzy inference system where all nodes were created during learning [53, 199]. In contrast to evolving networks with fixed topologies and connections, Han *et al.* [140] proposed an Evolutionary Neural Network (ENN) algorithm to evolve an ANN for detecting anomaly system call sequences. A matrix-based genotype representation was implemented, where the upper right triangle was the connectivity information between nodes, and the lower left triangle described the weights between nodes. Consequently, this network has no structural restrictions, and is more flexible, as shown in Figure 6. Xu *et al.* [285] presented a misuse detection model constructed by the understandable Neural Network Tree (NNTree). NNTree is a modular neural network with the overall structure being a decision tree, but each non-terminal node being an expert NN. GAs recursively designed these networks from the root node. The designing process was, in fact, solving a multiple objective optimization problem, which kept



(a) MLFF (b) RNN

(c) ENN

Fig. 6. Comparing different structures of ANNs [140].

the partition ability of the networks high, and the size of trees small. Chen *et al.* [56] investigated the possibility of evolving ANNs by an Estimation of Distribution Algorithm (EDA), a new branch of EC. The modeling and sampling step in an EDA improves search efficiency, because sampling is guided by global information extracted through modeling to explore promising areas.

Experimental results of the above works all confirmed that automatically designed networks outperform conventional approaches in detection accuracy. Han *et al.* [140] further verified that evolutionary approaches reduce training time.

As for clustering algorithms, evolutionary algorithms shorten the tedious and time-consuming process of deciding appropriate cluster centers and the number of clusters. Leno *et al.* [195] first reported work for combining unsupervised niche clustering with fuzzy set theory for anomaly detection, and applied it to network intrusion detection. Here "unsupervised" means that the number of clusters is automatically determined by a GA. An individual, representing a candidate cluster, was determined by its center, an n-dimensional vector with n being the dimension of the data samples, and a robust measure of its scale (or dispersion) δ^2. The scale was updated every generation based on the density of a hypothetical cluster. Lu *et al.* [207, 209] applied a GA to decide the number of clusters based upon Gaussian Mixture Models (GMM). This model assumes that the entire data collection can be seen as a mixture of several Gaussian distributions, each potentially being a cluster. An entropy-based fitness function was defined to measure how well the GMMs approximated the real data distribution. Thereafter, a K-means clustering algorithm was invoked to locate the center of each cluster. [297], in contrast, reversed the order of the K-means and evolutionary approaches. K-means was used to decide potential cluster centers, followed by the GA refining cluster centers.

4.3.1.3. *Classifiers* Evolutionary algorithms can be used to generate two types of classifiers: classification rules and transformation functions. A classification rule is the rule with an if-then clause, where a rule antecedent (IF part) contains a conjunction of conditions on predicting attributes, and the rule consequent (THEN part) contains the class label. As depicted in Figure 7, the task of EC is to search for classification rules (represented as circles) that cover the data points (denoted as "+") of unknown concepts (represented as shaded regions). In this sense, evolving classification rules can be regarded as concept learning.

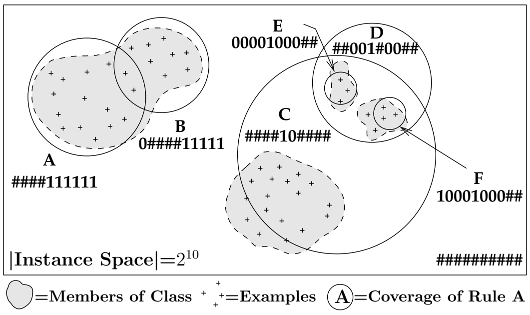


Fig. 7. Classification rules are represented as circles who cover the data points (denoted as "+") of unknown concepts (represented as shaded regions) [157]

Research work that explores the evolution of classification rules for intrusion detection is summarized in Table 3. The difference between binary classifiers and multi-classifiers is the representation.

Table 3
Evolving Classification Rules by EC

	Type	Research Work
GA	Binary Classifiers	[120], [121], [197], [255], [221], [230], [281]
	Multi-classifiers	[36], [65], [124], [240], [250], [251], [249], [252]
Tree GP	Binary Classifiers	[64], [208], [287]
	Multi-classifiers	[103], [104]

A GA uses fixed length vectors to represent classification rules. Antecedents and class label in if-then rules are encoded as genes in a chromosome (shown in Figure 8). Either

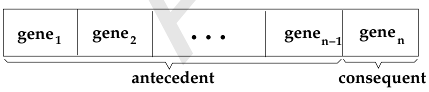
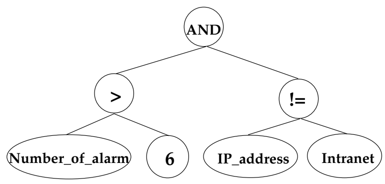


Fig. 8. GA chromosome structures for classification

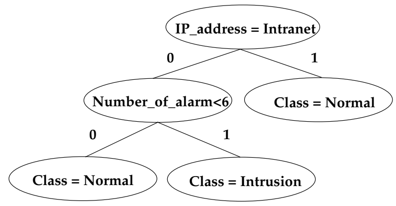
binary [167, 221, 230] or real-number [124, 197, 198, 240, 255] encoding schemes are conceived. A "don't care" symbol, $*$, is included [124, 167, 197, 198, 221, 230, 240, 255] as a wild card that allows any possible value in a gene, thus improving the generality of rules. For binary classification, the consequent part of rules are usually omitted from the representation, because of the same class label in all rules.

All research work listed for GAs employs the Michigan approach [155] as the learning approach, but is based on various GA models. [255, 197, 240, 124, 36] use classic GAs with niching to help covering all data instances with a minimum set of accurate rules. [221, 230] use the REGAL to model normal network traffic. REGAL [117] is a distributed genetic algorithm-based system. It shows several novelties, such as a hybrid Pittsburgh and Michigan learning approach, a new selection operator allowing the population to asymptotically converge to multiple local optima, a new model of distribution and migration, etc. [65, 250, 251, 249, 252] report initial attempts to extend XCS, an evolutionary Learning Classifier System (LCS), to intrusion detection problems. Although XCSs have shown excellent performance on some data mining tasks, many enhancements, such as mutation and deletion operators, and a distance metric for unseen data in the test phase, are still needed to tackle hard intrusion detection problems [65].

GP, on the other hand, uses different variable length structures for binary and multi-class classification. Originally, GP was confined to tree structures which provided the basis for the first IDS applications. For instance, the parse tree shown in Figure 9(a) for binary classification [64, 208, 287], and a decision tree shown in Figure 9(b) for multiple class classification [103, 104]. Compared with a GA which connects conditions in the antecedent only by the "AND" operator, tree-based GP has richer expressive power as it allows more logic operators, such as "OR", "NOT", etc. Crosbie [64] and Folino *et al.* [103, 104] improved the performance of such a GP system by introducing cooperation between individuals. The former use autonomous agents, each being a GP-evolved program to detect intrusions from only one data source. The latter deployed their system in a distributed environment by using the island model.



(a) Tree GP Chromosome for Binary Classification



(b) Tree GP Chromosome for Multiple Class Classification [261]

Fig. 9. Chromosome structures for classification

Namely, classification can also be achieved by a transformation function, which transforms data into a low dimensional space, i.e. 1D or 2D, such that a simple line can best separate data in different classes (shown in Figure 10).

The simplest transformation function is a linear function with the following format: $C(\chi) = \sum_{j=1}^{n}(w_j \times \chi_j)$, where n

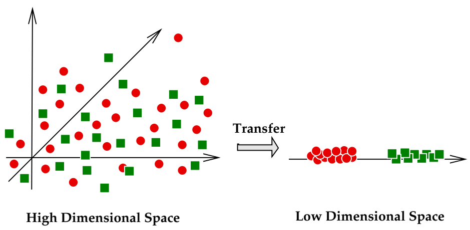


Fig. 10. Transformation Functions as Classifiers. A transformation function is an equation which transforms data in a high dimensional space into a specific value or a range of values in a low dimensional space according to different class labels.

is the number of attributes, w_j is a weight [282] or coefficient [61] of attribute χ_j. A GA usually searches for the best set of weights or coefficient that map any data in normal class to a value larger than δ ($C(\chi) > \delta$) and any data from anomaly class to a value less than δ ($C(\chi) < \delta$). δ is a user defined threshold. Individuals in this case contain n genes, each for a weight or coefficient.

Compared with GAs, transformation functions evolved by GP have more complex structures, normally nonlinear functions. Both tree-based GP (shown in Figure 9(a)) and linear GP (shown in Figure 11) are suitable for evolving the functions. Linear GP (LGP) is another major approach to GP [37, 41]. LGP works by evolving sequences of instructions from an imperative programming language or from a machine language. Figure 11 contains two typical examples of instructions in LGP. LGP boosts the evolutionary process because individuals are manipulated and executed directly without passing an interpreter during fitness calculation. Only arithmetic operators, such as "+", "-", "×", "÷", "log", and numeric values are allowed to appear in the representation of the functions. Categorical attributes have to convert their value to numeric beforehand.

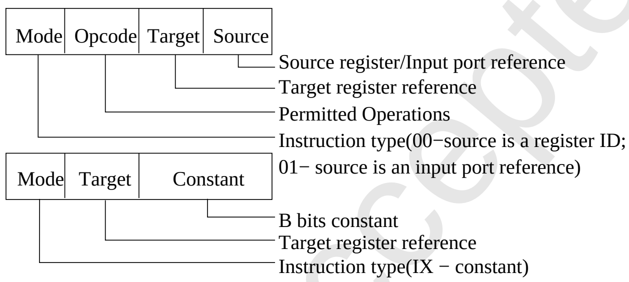


Fig. 11. Linear GP Chromosome [261]

Abraham *et al.* [12, 13, 138, 228] and Song *et al.* [259, 260, 261] are two major research groups working on LGP and its application in intrusion detection. Abraham *et al.* focused on investigating basic LGP and its variations, such as Multi-Expression Programming (MEP) [232] and Gene Expression Programming (GEP) [100], to detect network intrusion. Experiments, in comparing LGP, MEP, GEP and other machine learning algorithms, showed that LGP outperformed SVMs and ANNs in terms of detection accuracy at the expense of time [227, 228]; MEP outperformed LGP for Normal, U2R and R2L classes and LGP outperformed MEP for Probe and DoS classes [12, 13, 138]. Song *et al.* implemented a page-based LGP with a two-layer subset selection scheme to address the binary classification problem. Page-based LGP means that an individual is described in terms of a number of pages, where each page has the same number of instructions. Page size was dynamically changed when the fitness reached a "plateau" (i.e. fitness does not change for several generations). Since intrusion detection benchmarks are highly skewed, they pointed out that the definition of fitness should reflect the distribution of class types in the training set. Two dynamic fitness schemes, dynamic weighted penalty and lexicographic fitness, were introduced. The application of their algorithms to other intrusion detection related research can be found in [191, 192].

The above mentioned transformation functions evolved by GP are only used for binary classification. Therefore, Faraoun *et al.* [96] and Lichodzijewski *et al.* [200] investigated the possibilities of GP in multi-category classification. Faraoun *et al.* implemented multi-classification in two steps. In the first step, a GP maps input data to a new one-dimensional space, and in the second step, another GP maps the output from the first step to different class labels; Lichodzijewski *et al.* proposed a bid-based approach for coevolving LGP classifiers. This approach coevolved a population of learners that decompose the instance space by the way of their aggregate bidding behavior.

Research work that investigates evolving transformation functions for intrusion detection is summarized in Table 4.

Table 4
Evolving Transformation Functions by EC

Type		Research Work
Binary Classifiers	GA	[61], [282]
	LGP	[12], [13], [138], [145], [191], [192], [228], [259], [260], [261]
Multi-classifiers	Tree-based GP	[96]
	LGP	[200]

4.3.2. *Niching and Evolutionary Operators*

4.3.2.1. *Niching* Most EC applications have focused on optimization problems, which means that individuals in the population compete with others to reach a global optimum. However, pattern recognition or concept learning is actually a multimodal problem in the sense that multiple rules (see Figure 7) or clusters [195] are required to cover the unknown knowledge space (also known as "set covering" problem). In order to locate and maintain multiple local optima instead of a single global optimum, niching is introduced. Niching strategies have been proven effective in creating subpopulations which converge on local optima, thus maintaining diversity of the population [109].

Within the context of intrusion detection, both sharing and crowding are applied to encourage diversity. [171, 197, 198] employed fitness sharing, while [255] employed crowding and [195] employed deterministic crowding (DC). DC is an improved crowding algorithm, which nearly eliminates replacement errors in De Jong's crowding. Consequently, DC is effective in discovering multiple local optima, compared to no more than 2 peaks in De Jong's [214]. Unfortunately, there is no experimental result available in [255], so we cannot justify the limitations of De Jong's crowding in the intrusion detection domain. Hamming distance [197, 198, 255] or Euclidean distance [171] were used to measure the similarity between two individuals in both niching schemes.

However, defining meaningful and accurate distance measures and selecting an appropriate niching radius are difficult. In addition, computational complexity is an issue for these algorithms. For example, the shared fitness evaluation requires, in each generation, a number of steps proportional to M^2, with M being the cardinality of the population [117]. So, Giordana *et al.* introduced a new selection operator in REGAL, called *Universal Suffrage*, to achieve niching [117]. The individuals to be mated are not chosen directly from the current population, but instead indirectly through the selection of an equal number of data points. It is important to notice that only individuals covering the same data points compete, and the data points (stochastically) "vote" for the best of them. In XCS, the niching mechanism was demonstrated via reward sharing. Simply, an individual shares received rewards with those who are similar to them in some way [65].

Lu *et al.* [208] implemented niching neither via fitness sharing nor via crowding, but via token competition [196]. The idea is as follows: a token is allocated to each record in the training dataset. If a rule matches a record, its token will be seized by the rule. The priority of receiving the token is determined by the strength of the rules. On the other hand, the number of tokens an individual acquires also helps to increase its fitness. In this way, the odds of two rules matching the same data are decreased, hence the diversity of the population is maintained.

4.3.2.2. *Evolutionary Operators* In EC, during each successive generation, some individuals are selected with certain probabilities to go through crossover and mutation for the generation of offspring. Table 5 summarizes commonly used selection, crossover and mutation operators employed in intrusion detection tasks.

Table 5
Evolutionary Operators Employed in Intrusion Detection Tasks

Operators		Research Work
Selection	Roulette wheel	[65], [96], [167]
	Tournament	[70], [85], [145], [259]
	Elitist	[151], [124]
	Rank	[140], [281]
Crossover	Two-point	[65], [70], [96], [124], [167], [208], [221], [230], [287]
	One-point	[36], [140], [195], [281], [285]
	Uniform	[151], [221], [230]
	Arithmetical	[151]
	Homologous	[145], [192], [191], [259], [260], [261]
Mutation	Bit-flip	[65], [70], [151], [167], [195], [221], [230], [281], [285]
	Inorder mutation	[240]
	Gaussian	[151]
	One point	[96], [208], [287]

Some special evolutionary operators were introduced to satisfy the requirements of representation. For example, page-based LGP algorithms [192, 191, 259, 260, 261] restricted crossover to exchanging pages rather than instructions between individuals. Mutation was also conducted in two ways: in the first case the mutation operator selected two instructions with uniform probability and performed an XOR on the first instruction with the second one; the second mutation operator selected two instructions in the same individual with uniform probability and then exchanged their positions. Hansen *et al.* [145] proposed a homologous crossover in LGP, attempting to mimic natural evolution more closely. With homologous crossover, the two evolved programs were juxtaposed, and the crossover was accomplished by exchanging sets of continuous instruction blocks having the same length and the same position between the two evolved programs.

Most researchers have confirmed the positive role mutation played in the searching process. However, they held different opinions about crossover in multimodal problems whose population contains niches. Recombining arbitrary pairs of individuals from different niches may cause the formation of unfit or lethal offspring. For example, if a crossover is conducted on the class label part, which means rules in different classes exchange their class labels, it would cause a normal data point to be anomalous, or vice versa. Hence, a mating restriction is considered when individuals of different niches are crossed over. [240] only applied mutation, not crossover, to produce offspring; [70] restricted mutation and crossover to the condition-part of rules; [195] introduced an additional restriction on the deterministic crowding selection for controlling the mating between members of different niches.

Except for these three operators, many others were conceived for improving detection rate, maintaining diversity or other purposes. Among them, seeding and deletion are two emerging operators that are adopted by many EC algorithms in intrusion detection applications.

– Seeding [65, 117]. As discussed earlier, evolving classification rules can be regarded as a "set covering" problem. If some instances are not yet covered, seeding operators will dynamically generate new individuals to cover them. Normally, this method is used to initialize the first population at the beginning of the search.

– Deletion [65]. EC works with a limited population size. When a newly generated individual is being inserted into the population, but the maximum population size is reached, some old individuals have to be removed from the population. In traditional EC with a global optimum target, the less fit individuals are preferably replaced. However, for multimodal problems, other criteria in addition to fitness, such as niches or data distribution, should be considered to avoid replacement errors. [65] extended the deletion operator of XCS by considering class distribution, especially for highly skewed datasets. For example, normal instances constitute approximately 75% of total records in the KDD99 dataset. Therefore, rules which cover normal data points will have a higher fitness than others, which implies that rules for the normal class have a much lower chance to be deleted compared to rules for other classes. So, integrating class distribution into the deletion operator allows it to handle minority classes.
– Adding and Dropping. These two operators are variations of mutation. When evolving rules, dropping means to remove a condition from the representation, thus resulting in a generalized rule [208, 287]. On the contrary, adding conditions results in a specialized rule. Han *et al.* [140] employed adding and dropping to add a new connection between neurons, and to delete the connection between neurons, respectively in an evolutionary neural network.

4.3.3. *Fitness Function*

An appropriate fitness function is essential for EC as it correlates closely with the algorithm's goal, thus guiding the search process. Intrusion detection systems are designed to identify intrusions as accurately as possible. Therefore, accuracy should be a major factor when yielding a fitness function. In Table 6, we categorize the fitness function from research work we surveyed. The categorization is based on three terms: detection rate (DR), false positive rate (FPR) and conciseness.

The research contributions in the first row are all devoted to anomaly detection problems. Since no attack is presented in the training phase, DR is not available. Fitness functions may vary in format, but all look for models which cover most of the normal data. In this example, $H(C_i)$ represents the entropy of data points that belong to cluster C_i, and $H_{max}(C_i)$ is the theoretical maximum entropy for C_i.

Accuracy actually requires both, DR and FPR, since ignoring either of them will cause misclassification errors. A good IDS should have a high DR and a low FPR. The first example in the second row directly interprets this principle. Here, α stands for the number of correctly detected attacks, A the number of total attacks, β the number of false positives, and B the total number of normal connections. As we know, patterns are sometimes represented as if-then clauses in IDSs, so in the second example, the support-confidence framework is borrowed from association rules to determine the fitness of a rule. By changing weights w_1 and w_2, the fitness measure can be used for either simply identifying network intrusions, or precisely classifying the type of intrusion [124]. The third example considers the absolute difference between the prediction of EC (φ_p) and the actual outcome (φ).

Conciseness is another interesting property that should be considered. This is for two reasons: concise results are easy to understand, and concise results avoid misclassification errors. The second reason is less obvious. Conciseness can be restated as the space a model, such as a rule, or a cluster, uses to cover a dataset. If rule A and rule B have the same data coverage, but rule A is more concise than B, so A uses less space than B does when covering the same amount of data. The extra space of B is more prone to cause misclassification errors. Apparently the first example of this kind considers all three terms, where the length correlates with conciseness. The second example of this type considers the number of counterexamples (w) covered by a rule, and the ratio between the number of bits equal to 1 in the chromosome and the length of chromosome (z), which is the conciseness of a rule. A is a user-tunable parameter. The fitness function in [195] also prefers clusters with small radii if they cover the same data points.

4.3.4. *Summary*

In this section, we reviewed the research in employing evolutionary computation to solve intrusion detection problems. As is evident from the previous discussion, EC plays various roles in this task, such as searching for an optimal solution, automatic model design, and learning for classifiers. In addition, experiments reasserted the effectiveness and accuracy of EC. However, we also observed some challenges for the method, as listed below. Solving these challenges will further improve the performance of EC-based intrusion detection.

– No reasonable termination criterion. Most research work simply sets the termination criterion as a pre-specified number of iterations, or a threshold of fitness. However, the experiment of Shafi *at al.* [251] showed that such simple criteria while helpful when searching for the global optimum, are inappropriate for multiple local optima. A reasonable termination criterion will definitely improve detection accuracy and efficiency.
– Niching. Learning intrusion behavior is equivalent to concept learning, which is always looking for multiple solutions. Although niching is capable of discovering and maintaining multiple local optima, it cannot guarantee that a complete set of solutions is returned. More research work is required to investigate how to maintain a diverse, and complete solution by EC.
– Distributed EC models. Training sets in intrusion detection are normally generated from a large volume of network traffic dumps or event logs. This makes evaluating candidate solutions in EC quite expensive and time consuming. In contrast to monolithic architectures, distributed models [104, 117, 151] have the advantage

Table 6
Fitness Summary

Factors			Examples	References
DR	FPR	Conciseness		
×	√	×	$\frac{H(C_i)}{H_{max}(C_i)}$	[140], [195], [209], [207]
√	√	×	$\frac{\alpha}{A} - \frac{\beta}{B}$	[61], [85], [96], [167], [192], [240], [255], [282], [297]
			$w_1 \times support + w_2 \times confidence$	[36], [124], [208], [281], [287]
			$1 - \lvert\varphi_p - \varphi\rvert$	[31], [64], [138], [197], [198], [259]
√	√	√	$w_1 \times sensitivity + w_2 \times specificity + w_3 \times length$	[121]
			$(1 + Az) \times e^{-w}$	[70], [221], [230]

of assigning a portion of the data to each node, hence they put less burden on fitness evaluation. In addition, distributed nodes are trained simultaneously and independently, so they can be added to and removed from the system dynamically. There are, however, still many issues deserving careful investigation, such as evolutionary models or communication mechanisms in a distributed environment.

– Unbalanced data distribution. One important feature of intrusion detection benchmarks is their high skewness. Take the KDD99-10 dataset as an example: there are 391,458 instances in the DoS class while only 52 instances are in the U2R class. Both [65] and [259] point out individuals which had better performance on frequently occurring connection types would be more likely to survive, even if they performed worse than competing individuals on the less frequent types. Therefore, when designing an intrusion detection system based on EC approaches, one should consider how to improve the accuracy on relatively rare types of intrusion without compromising performance on the more frequent types.

4.4. *Artificial Immune Systems*

The *human immune system* (HIS) has successfully protected our bodies against attacks from various harmful pathogens, such as bacteria, viruses, and parasites. It distinguishes pathogens from self tissue, and further eliminates these pathogens. This provides a rich source of inspiration for computer security systems, especially intrusion detection systems. According to [175, 258], features gleaned from the HIS satisfy the requirements of designing a competent IDS [153, 175]. Hence, applying theoretical immunology and observed immune functions, its principles, and its models to IDS has gradually developed into a new research field, called *artificial immune system* (AIS).

AIS based intrusion detection systems perform anomaly detection. However, instead of building models for the normal, they generate non-self (anomalous) patterns by giving normal data only, as Figure 12 illustrated. Any matching to non-self patterns will be labeled as an anomaly.

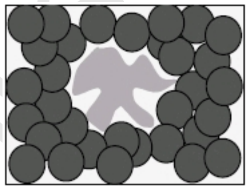


Fig. 12. The goal of AIS-based IDSs is to generate all patterns, denoted as black circles, which match none of the normal data. The shaded region represents a space containing only normal data [153].

In this section, we will review research progress on immune system inspired intrusion detection. Although review work for AISs [26, 67, 73, 105, 161] and their application to the intrusion detection domain [20, 178] exists, our review is different in that it focuses on two perspectives: tracking the framework development of AIS based IDSs, and investigating the key elements shown in Figure 13 when engineering an AIS-based intrusion detection system [73]. In recent years, research on AIS has extended to the study of innate immune systems, in particular to the danger theory proposed by Matzinger [217, 218]. Hence, the last part of this section will present IDSs motivated by the danger theory.

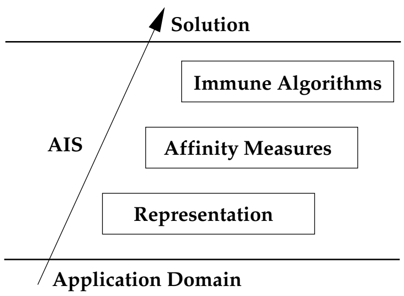


Fig. 13. The framework to engineer an AIS. Representation creates abstract models of immune cells and molecules; affinity measures quantify the interactions among these elements; algorithms govern the dynamics of the AIS [73].

4.4.1. *A Brief Overview of Human Immune System*

Before we start the discussion of AIS models, a brief overview of the HIS will be necessary. A more detailed introduction of the HIS can be found elsewhere [74]. Our human immune system has a multi-layered protection architecture, including physical barriers, physiological barriers, an innate immune system, and an adaptive immune system. Compared to the first three layers, the adaptive immune system is capable of adaptively recognizing specific types of pathogens, and memorizing them for accelerated future responses [153]. It is the main inspiration for AISs.

The adaptive immune system is a complex of a great variety of molecules, cells, and organs spread all over the body, rather than a central control organ. Among its cells, two lymphocyte types, T cells and B cells, cooperate to distinguish self from non-self (known as antigens). T cells recognize antigens with the help of *major histocompatibility complex* (MHC) molecules. *Antigen presenting cells* (APC) ingest and fragment antigens to peptides. MHC molecules transport these peptides to the surface of APCs. T cells, whose receptors bind with these peptide-MHC combinations, are said to recognize antigens. In contrast, B cells recognize antigens by binding their receptors directly to antigens. The bindings actually are chemical bonds between receptors and epitopes/peptides. The more complementary the structure and the charge between receptors and epitopes/peptides are, the more likely binding will occur. The strength of the bond is termed "affinity".

T cells and B cells develop and mature within the thymus and bone marrow tissues, respectively. To avoid autoimmunity, T cells and B cells must pass a negative selection stage, where lymphocytes which match self cells are killed. Prior to negative selection, T cells undergo positive selection. This is because in order to bind to the peptide-MHC combinations, they must recognize self MHC first. So the positive selection will eliminate T cells with weak bonds to self MHC. T cells and B cells which survive the negative selection become mature, and enter the blood stream to perform the detection task. These mature lymphocytes have never encountered antigens, so they are naive.

Naive T cells and B cells can still possibly autoreact with self cells, because some peripheral self proteins are never presented during the negative selection stage. To prevent self attack, naive cells need two signals in order to be activated: one occurs when they bind to antigens, and the other is from other sources as a "confirmation". Naive T helper cells receive the second signal from innate system cells. In the event that they are activated, T cells begin to clone. Some of the clones will send out signals to stimulate macrophages or cytotoxic T cells to kill antigens, or send out signals to activate B cells. Others will form memory T cells. The activated B cells migrate to a lymph node. In the lymph node, a B cell will clone itself. Meanwhile, somatic hypermutation is triggered, whose rate is 10 times higher than that of the germ line mutation, and is inversely proportional to the affinity. Mutation changes the receptor structures of offspring, hence offspring have to bind to pathogenic epitopes captured within the lymph nodes. If they do not bind they will simply die after a short time. If they succeed in binding, they will leave the lymph node and differentiate into plasma or memory B cells. This process is called affinity maturation. Note, clonal selection affects both T cells and B cells, but somatic mutation has only been observed in B cells. As we can see, by repeating selection and mutation, high affinity B cells will be produced, and mutated B cells adapt to dynamically changing antigens, like viruses.

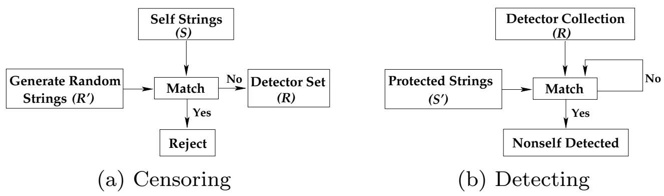
The immune response caused by activated lymphocytes is called primary response. This primary response may take several weeks to eliminate pathogens. Memory cells, on the other hand, result in quick reaction when encountering pathogens that they have seen before, or that are similar to previously seen pathogens. This process is known as secondary response, which may take only several days to eliminate the pathogens.

In summary, the HIS is a distributed, self-organizing and lightweight defense system for the body [175]. These remarkable features fulfill and benefit the design goals of an intrusion detection system, thus resulting in a scalable and robust system.

4.4.2. *Artificial Immune System Models for Intrusion Detection*

The HIS is sophisticated, hence researchers may have different visions for emulating it computationally. In this section, we will review the development of AIS models for solving intrusion detection problems.

4.4.2.1. *A self-non-self discrimination AIS model* The first AIS model suggested by Forrest *et al.* was employed in a change-detection algorithm to detect alterations in files [108] and system call sequences [107]. This model simulated the self-non-self discrimination principle of the HISs, as illustrated in Figure 14. Negative selection was the core of this model, by which invalid detectors were eliminated when they matched self data. Although not many immune features were employed, it reflected some initial steps toward a greater intellectual vision on robust and distributed protection systems for computers [106].



(a) Censoring (b) Detecting

Fig. 14. The self-non-self discrimination model. A valid detector set will be generated, and then monitor protected strings [108].

4.4.2.2. *An AIS model with lifecycle* Hofmeyr and Forrest later extended the above prototype with more components and ideas from the HIS. The new AIS model (shown in Figure 15) considered the lifecycle of a lymphocyte: immature, mature but naive, activated, memory, and death. The finite detectors' lifetime, plus costimulation, distributed tolerance and dynamic detectors contribute to eliminating autoreactive detectors, adapt to changing self sets, and improve detection rates through signature-based detection. As an application of this model, a system called LISYS

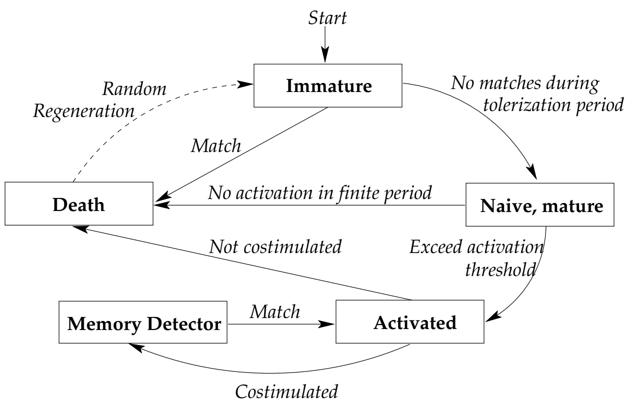


Fig. 15. The lifecycle of a detector. A set of detectors are generated randomly as immature detectors. An immature detector that matches none of normal data during its tolerization period becomes mature; otherwise it dies. When a mature detector matches sufficient input data, this detector will be activated. Alternatively, a mature detector that fails to become activated eventually dies. Within a fixed period of time, if an activated detectors receive no co-stimulation, e.g. responses from system security officers, it will die too; otherwise it becomes a memory detector [119].

(Lightweight Immune SYStem) was developed to detect intrusions in a distributed environment. Williams *et al.* employed this model to detect computer viruses [146] and network intrusions [280], but extended it with an affinity maturation step to optimize the coverage of the non-self space of antibodies [147, 280].

4.4.2.3. *An evolutionary AIS model* Kim and Bentley proposed an AIS model [175] based on three evolutionary stages: gene library evolution, negative selection and clonal selection, shown in Figure 16. The gene library stores potentially effective genes. Immature detectors, rather than generated randomly, are created by selecting and rearranging useful genes. Genes in successful detectors are added to the library, while those in failed detectors are deleted. In a sense, the library evolves; the negative selection removes false immature detectors by presenting self without any global information about self; the clonal selection detects various intrusions with a limited number of detectors, generates memory detectors, and drives the gene library evolution. Hofmeyr's lifecycle model was adopted in their model.

4.4.2.4. *A multi-level AIS model* T cells and B cells are two primary but complex immunological elements in the HIS. Focusing on their functions and interactions, Dasgupta *et al.* [69] proposed a model that considers detecting intrusions and issuing alarms in a multi-level manner (see Figure 17). T cells recognize the peptides extracted from foreign proteins, while B cells recognize epitopes on

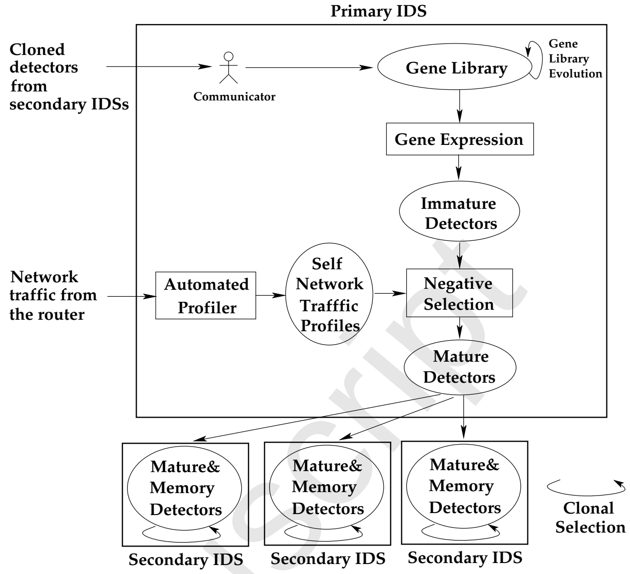


Fig. 16. Conceptual architecture of Kim and Bentley's AIS Model. The central primary IDS generates valid detectors from gene library, and transfers unique detector subsets to distributed secondary IDSs. Secondary IDSs execute detection task, as well as proliferate successful detectors [175].

the surface of antigens. Therefore, in their computational model, T-detectors (analogous to T cells) performed a low-level continuous bitwise match, while the B-detectors (analogous to B cells) performed a high-level match at non-contiguous positions of strings. To prevent the system from raising false alarms, T-suppression detectors (analogous as T-suppression cells) are introduced, which decide the activation of T-detectors. Activated T-detectors will further provide a signal to help activate B-detectors. This model further simulated negative selection, clonal selection and somatic hypermutation of mature T cells and B cells.

4.4.2.5. *Artificial Immune Network Model* Artificial Immune Networks (AIN) are based on the immune network theory proposed by Jerne [158]. This theory hypothesizes that the immune system maintains an idiotypic network of interconnected B cells for antigen recognition. These B cells stimulate or suppress each other to keep the network stable. In AIN, antigens are randomly selected from the training set and presented to B cells. The stimulation effects between B cells and antigens (binding) are calculated. Meanwhile, the stimulation and suppression effects between B cells are also calculated. B cells will be selected to clone and mutate based on the total interaction effects. Useless B cells are removed from the network, while new B cells are created randomly and incorporated into the network, and links among all B cells are reorganized. A network is returned for detection when the stopping criterion is met. Based on Jerne's work, many AIN models have been developed [112], as shown in Figure 18. AINs have been proposed for problem solving in areas such as data analysis, pattern recognition, autonomous navigation and function optimization.

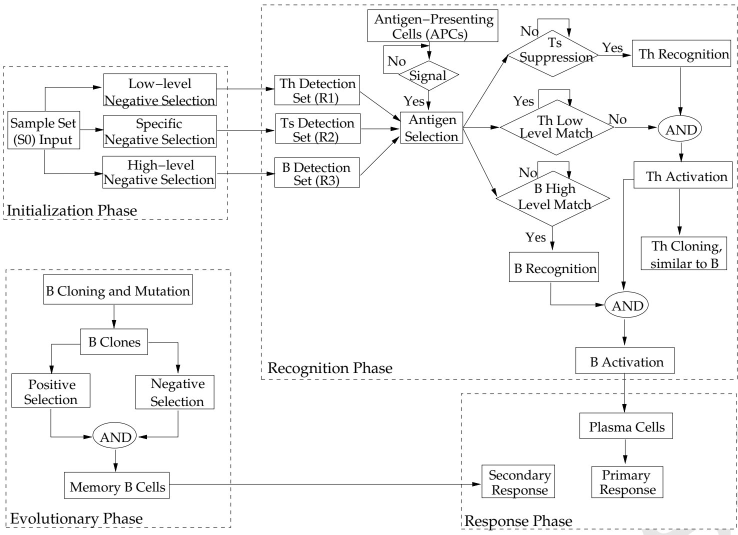


Fig. 17. A multi-level AIS model proposed by Dasgupta *et al.* [69].

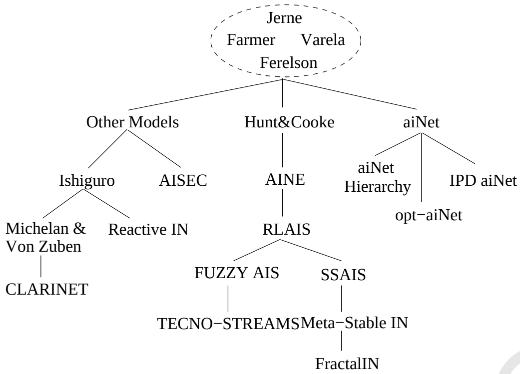


Fig. 18. Genealogical tree of AIN models: each model is a modification or is based on its parent [112].

4.4.2.6. *Other AIS models* Millions of lymphocytes circulate in the blood stream and lymph nodes, and perform the role of immune surveillance and response. Therefore, Dasgupta [66] and Hamer [146] both proposed a model for mapping the mobility of cells into an AIS by mobile agents. Lymphocytes, antibodies and other cells are mapped into agents roaming around a protected system to perform sensing, recognizing, deleting and cleaning jobs. Luther *et al.* [213] presented a cooperative AIS framework in a P2P environment. Different AIS agents collaborate by sharing their detection results and status. Twycross *et al.* [273] incorporated ideas from innate immunity into artificial immune systems (AISs) and presented an *libtissue* framework.

4.4.3. *Representation Scheme and Affinity Measures*

The core of the HIS is self and non-self discrimination performed by lymphocytes. To engineer such a problem in computational settings, the key steps are appropriately representing lymphocytes and deciding the matching rules.

Antibodies are generated by random combinations of a set of gene segments. Therefore, a natural way to represent detectors is to encode them as gene sequences, comparable to chromosomes in genetic algorithms. Each gene represents an attribute in the input data. Normally, a detector is interpreted as an if-then rule, such as Figure 19 has shown. The affinity, when mapped into the intrusion detection domain, means the similarity between detectors and data.

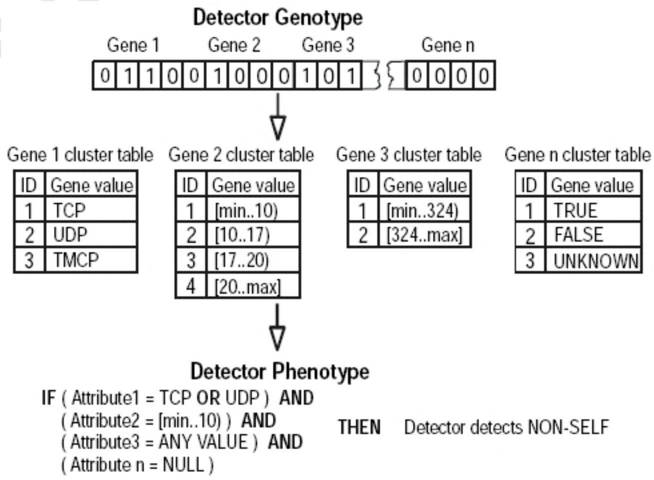


Fig. 19. Detector genotype and phenotype [175].

Binary strings are the most commonly adopted coding schemes. There are two ways to represent detectors in binary strings. The difference lies in how to determine the number of nucleotides. Suppose the number of nucleotides in a gene is denoted as N_n, and the number values of an attribute is denoted as N_a. N_n can either be equal to N_a [180, 175] or be the minimum integer which satisfies $2^{N_n} >= N_a$ [26, 108, 119, 146, 153, 280]. The first representation allows a single attribute of each detector to have more than one value, but requires more space. Affinity measures for binary strings are r-contiguous bits matching (rcb) [108], r-chunks matching [32], landscape-affinity matching [146], Hamming distance and its variations. Compared to perfect matching, these partial matchings provide generalization for a learning algorithm. Homer compared rcb, landscape-affinity matching, Hamming distance and its variations on a randomly generated dataset [146]. The results showed that the *Rogers and Tanimoto* (*R&T*), a variation of the Hamming distance, produced the best performance.

González [127] further compared *R&T* with r-chunks, *rcb* and Hamming distance on two **real-valued** datasets. Although r-chunks outperformed others, it still showed a very high false positive rate. This can be explained by the intrinsic meaning of difference or similarity in numeric data. Affinity measures suitable for binary strings do not correctly reflect the distance in numeric meanings.

Therefore, two real-valued representations were suggested by Dasgupta's research group to encode numeric information. In the first coding scheme, a gene in a detector has two nucleotides: one saves the lower bound value of an attribute, and the other one saves the upper bound [68]. Hence, a chromosome actually defines a hypercube. In the second coding scheme, a detector has n+1 genes, where the first n genes represent the center of an n-dimensional hypersphere, and the last gene represents the radius [128]. Major matching rules used in real-valued representation include: Euclidean distance, generalized distances of different norms in Euclidean space (including special cases; Manhattan distance (1-norm), Euclidean distance (2-norm), λ-norm distance for any λ, and infinity norm distance), interval-based matching, and other distance metrics [166].

Representations combining the two approaches were adopted, too [143]. Numeric attributes are encoded in real-valued format, and category attributes are encoded in strings. Matching rules were accordingly applied.

4.4.4. *Negative Selection Algorithms*

The *negative selection* (NS) algorithm simulates the process of selecting nonautoreactive lymphocytes. Consequently, given a set of normal data, it will generate a set of detectors which match none of these normal data samples. These detectors are then applied to classify new (unseen) data as self (normal) or non-self (abnormal). In this section, various NS algorithms will be summarized; then some key issues, such as detector generation, controlling the FP rate and FN rate, and coverage estimation will be discussed.

4.4.4.1. *Development of Negative Selection Algorithms*

The negative selection algorithm was first suggested by Forrest *et al.*, already shown in Figure 14. This algorithm started with a population of randomly generated detectors. These potential detectors, analogous to immature lymphocytes, were exposed to normal data. Those which matched normal data were removed from the population immediately and replaced by new detectors. Detectors which survived this selection process were used in the detection phase (shown in Figure 14b). In this model, self data and detectors were encoded as binary strings, and *rcb* matching rules decided the affinity.

Since the empirical study [127] supported the advantages of real-valued representations on numeric data, Dasgupta and his group extended the initial negative selection algorithm to a series of real-valued NS algorithms. Figure 20 lists NS algorithms proposed by that group and by other researchers. Dasgupta *et al.* hypothesized that each self sample and its vicinity is normal, so they considered a variability range (called vr) as the radius for a normal point. Obviously, representing normal data points by a hypersphere achieved generalization for unseen data. An example showing how a self region might be covered by circles in 2-dimension is given in Figure 21a.

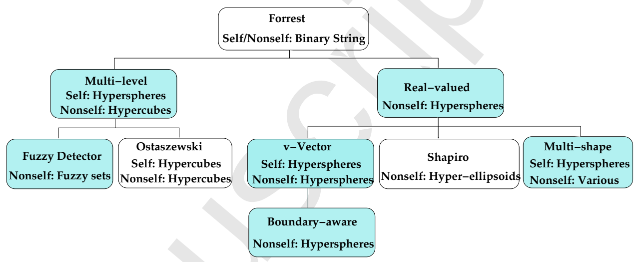
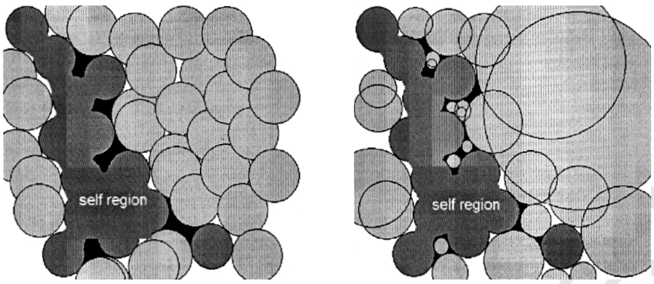


Fig. 20. Genealogical tree of real-valued NS algorithms: each model is a modification or is based on its parent. Dark rectangulars denote research work by Dasgupta groups, and white ones by other researchers.

Features of these NS algorithms can be summarized as follows:

- Multi-level: By changing the parameter vr of self hypersphere, a set of detectors with hierarchical levels of deviation were generated. Such a hierarchical detector collection characterized a noncrisp description for the non-self space [68]. A variation of this algorithm integrated fuzzy systems to produce fuzzy detectors [130].
- Real-valued: Instead of inefficiently throwing away detectors who match self samples, this algorithm gave these detectors a chance to move away from the self set during a period of adaptation. Detectors would eventually die if they still matched self sets within a given time frame. Meanwhile, detectors moved apart from each other in order to minimize the overlap in the non-self space [126]. In the end, this algorithm generated a set of constant-sized (because of constant radius) hypersphere detectors covering non-self space, as demonstrated in Figure 21a for a 2-dimensional space. Shapiro *et al.* expressed detectors by hyper-ellipsoids instead of hyperspheres [254].
- v-Vector: Clearly in real-valued NS algorithms, large numbers of constant-sized detectors are needed to cover the large area of non-self space, while no detectors may fit in the small area of non-self space, especially near the boundary between self and non-self. Hence a variable radius was suggested in the v-Vector algorithm [162, 163]. The core idea of this algorithm is illustrated in Figure 21b in a 2-dimensional space.
- Boundary-aware: Previous algorithms took each self sample and its vicinity as a self region, but deciding vicinity is difficult, especially for self samples that are close to the boundary between self and non-self. This algorithm aims to solve the "boundary dilemma" by considering the distribution of self samples.

– Multi-shape: Different geometric shapes, such as hyper-rectangles [68, 130], hyper-spheres [126, 162, 163] and hyper-ellipses [254], were used for covering the non-self space. This algorithm thus incorporated these multiple hyper-shape detectors together [28, 29]. Detectors with suitable size and shape were generated according to the space to be covered. As an application, this algorithm was used to detect intrusions in Ad-Hoc networks [30].
– Ostaszewski: Ostaszewski *et al.* argued that detectors generated by the multi-level NS algorithm cannot completely cover the non-self space, due to the shape conflict between the structures used for self (hypersphere) and non-self (hypercubes). Hence, in their algorithm, both self and non-self patterns were hypercubes. Self patterns, instead of self data, were used in the NS algorithm. The conversion of large self data space into comparatively small schemata space was effective, and the conversion compressed the number of inputs of the NS algorithm. A similar conversion was also suggested by Hang and Dai [142, 144].



(a) Constant-sized detectors (b) Variable-sized detectors

Fig. 21. The main concept of v-Vector. The dark area represents self region. The light gray circles are the possible detectors covering the non-self region [163].

New NS algorithms are continuously being published. For example, a NS algorithm, enhanced by state graphs [212], is able to locate all occurrences of multi-patterns in an input string by just one scan operation; a feedback NS algorithm was proposed to solve the anomaly detection problem [293].

Recently concerns were raised on the applicability of NS algorithms. Garrett [113] concluded that NS algorithms are distinct, and are suitable for certain applications only. Freitas *et al.* [111] criticized NS algorithms used as a general classification method because they are one-class based. Stibor *et al.* [262, 263] pointed out that a real-valued NS algorithm, defined over the hamming shape-space, is not well suited for real-world anomaly detection problems. To tackle these issues, Ji *et al.* [165] clarified some confusion that may have mislead the applicability of negative selection algorithms. [128, 144] also suggested another potential of NS algorithms as non-self data generators. The artificial non-self data can be mixed with self data to train classifiers, which helps to identify the boundary between normal and abnormal data.

4.4.4.2. *Detector Generation* The typical way of generating detectors in NS algorithms is random or exhaustive, as described in the model (Figure 14) originally proposed by Forrest *et al.*, later being frequently adopted in other research work [69, 125, 126, 153, 160, 163].

Instead of inefficiently throwing away detectors who match self samples, Ayara *et al.* [27] and González *et al.* [126] both decided to give these detectors a chance to move away from the self set in a period of time before eliminating them. Ayara *et al.* further compared their algorithm (NSMutation) with exhaustive, linear [81], greedy [81], and binary template [279] detector generating algorithms in terms of time and space complexities. The results can be found in [27]. They concluded that though NSMutation was more or less an exhaustive algorithm, it eliminated redundancy and provided tunable parameters that were able to induce a different performance.

Recent trends are applying evolutionary algorithms to evolve detectors to cover the non-self space, since a similar evolution process was observed in antibodies. The evolutionary negative selection algorithm (ENSA) is shown in Figure 22, where a negative selection algorithm is embedded in a standard evolutionary process as an operator. Detectors which match the self data will either be penalized by decreasing their fitness or even removed from the population. Removed ones are replaced by newly generated detectors.

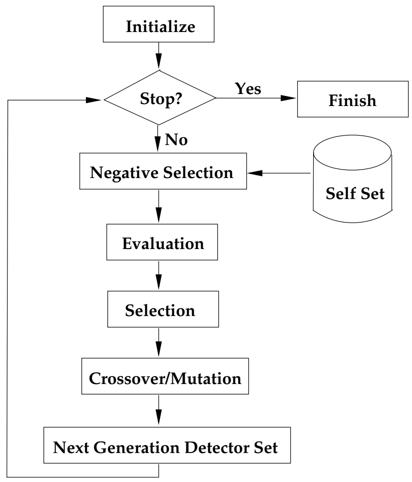


Fig. 22. Generating Detectors by Evolutionary Algorithms.

Kim *et al.* [176] introduced niching to the ENSA to maintain diversity. Diversity is necessary for ENSA because a set of solutions (detectors) collectively solves the problem (covering non-self space). Kim implemented niching in a way similar to the token competition. A self sample and several detectors were randomly selected. Only the detector which showed least similarity with the self sample had the chance of increasing its fitness.

Dasgupta's group claimed the detector generation was not only a multimodal optimization problem, but also a multiobjective problem [68]. Hence, they used sequential niching to achieve multimodal, and defined three reasonable criteria to evaluate a detector: a good detector must not cover self space; it should be as general as possible; and it has minimum overlap with the rest of the detectors. Therefore, the fitness function was defined as:

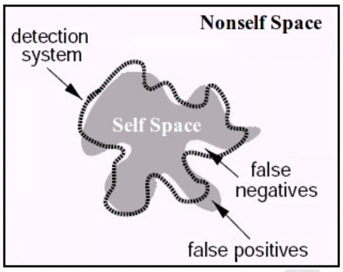
$$f(x) = volume(x) - (C \times num\_elements(x) + overlapped\_volume(x)) \tag{1}$$

where $volume(x)$ is the space occupied by detector x; $num\_elements(x)$ is the number of self samples matched by x; C is the coefficient. It specifies the penalty x suffers if it covers normal samples; $overlapped\_volume(x)$ is the space x overlaps with other detectors. Obviously, the first part is the reward, while the second part is the penalty. This multi-objective multimodal ENSA was applied in their multi-level NS [68], fuzzy NS [130] and multi-shape NS algorithms [28, 29]. Ostaszewski *et al.* also used this fitness definition in their work. The multi-shape NS used a structure-GA while the rest used standard GAs.

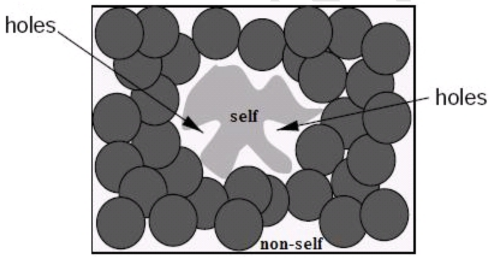
With the development of EC, ENSA is gradually strengthened by new evolutionary features. González and Cannady [131] implemented a self-adaptive ENSA, where the mutation step size was adjustable in a Gaussian mutation operator. Their method avoided trial and error when determining the values of tunable parameters in NSMutation; Ostaszewski *et al.* [233, 234, 235] employed co-evolution in their ENSA. A competitive co-evolutionary model helped detectors to discover overlooked regions. The anomaly dataset and the detector set took their turn as predators and prey. Detectors were trying to beat down anomaly data points by covering them. The fitness of data points not covered by any detector were increased, thus resulting in a high possibility of these points to be presented to detectors again. Haag *et al.* [139] employed a multi-objective evolutionary algorithm to measure the tradeoff among detectors with regard to two independent objectives: best classification fitness and optimal hyper-volume size.

4.4.4.3. *Controlling False Positive and False Negative Errors* Inaccurate boundaries between self and non-self space (see Figure 23a), and incomplete non-self patterns (see Figure 23b) are two main causes of false positive and false negative errors in AISs.

Self samples in training sets are never complete. As a result, some autoreactive detectors cannot be eliminated during negative selection. These detectors fail to recognize unseen normal data, thus causing false positives, as shown in Figure 23a. To avoid false positive errors, Hofmeyr [153] introduced the activation threshold (τ), sensitivity level (δ), and costimulation. Instead of signaling an alarm every time a match happens, a detector has to wait until it is matched at least τ times within a limited time period. However, if attacks are launched from different sources, a single detector cannot be matched repeatedly. Therefore, δ is intended to consider the matches of all detectors in a host. An alarm will be triggered when the contributions of multiple detectors exceeds δ within a limited time period. Costimulation requires a confirmation from a human operator whenever an activated detector raises an alarm.



(a) Inaccurate Boundaries



(b) Incomplete Non-self Patterns

Fig. 23. Reasons for FPR and FNR in AISs [153].

Giving generality to self samples is another way to address incomplete self samples problem. As previously discussed, Dasgupta's group used a hyper-sphere area around self samples in the NS algorithm. Although their methods successfully avoid overfitting, it unfortunately produces an over-generalization problem. Over-generalization will cause false negative errors as shown in Figure 23a. Therefore, Ji *et al.* proposed a boundary-aware algorithm [159]; Ostaszewski *et al.* presented the self samples by variable-sized hyper-rectangles; Hang *et al.* [142, 144] employed a co-evolutionary algorithm to evolve self patterns.

Incomplete non-self patterns in AISs are mainly caused by holes, which are the undetectable negative space (shown in Figure 23b). They are desirable to the extent that they prevent false positives if unseen self samples are falling into them. They are undesirable to the extent that they lead to false negatives if non-self samples are falling into them. Balthrop *et al* [32] and Esponda *et al.* [93, 94] pointed out that matching rules are one reason for inducing holes. For example, the r-contiguous bit matching rule induces either length-limited holes or crossover holes, while the r-chunks matching rule only induces crossover holes. Their analysis is consistent with the D'haeseleer's suggestion: using different matching rules for different detectors can reduce the overall number of holes [81]. Alternatively, using different representations helps to avoid holes, too. Hofmeyr [153] introduced the concept of permutation masks to give a detector a second representation. Permutation masks are analogous to the MHC molecules in HIS. In fact, changing representation is equivalent to changing the "shape" of detectors. Dasgupta and other researchers [233] then suggested variable-sized [162, 163, 234, 235] and variable-shaped detectors (e.g. hyper-rectangular [68, 130], hypersphere [126, 163], hyper-ellipsoid [254], or a combination of them [28, 29]). Niching sometimes contributes to filling holes, because it attempts to maximize the space coverage and minimize the overlaps among them.

Holes bring another issue. Hofmeyr explained in [153] that the longer the period of time over which holes remain unchanged, the more likely an intruder will find gaps, and once found, those gaps can be exploited more often. Therefore, he proposed a combination of rolling coverage and memory cells to solve this problem. Each detector is given a finite lifetime. At the end of its lifetime, it is eliminated and replaced by a new active detector, thus resulting in a rolling coverage. Memory detectors ensure that what has been detected in the past will still be detected in the future.

4.4.4.4. *The Estimation of Coverage* No matter whether detectors are generated exhaustively or by using evolutionary algorithms, a measure is required to decide when to stop the generation process. Estimating the coverage ratio, which is also called detector coverage, is one major research subject of NA algorithms.

Forrest [108] and D'haeseleer [81] estimated the number of detectors for a given failure probability when the exhaustive generation and the r-continuous matching rule were used; later Esponda *et al.* [94] discussed the calculation of the expected number of unique detectors under the r-chunks matching rule for both the positive and negative selection algorithm.

Dasgupta *et al.* [68] and Ji [163] estimated the coverage by retry times. Later Ji used hypothesis testing to estimate the detector coverage in v-vector NS algorithm [164]. González [129] and Balachandran [29] used the Monte Carlo estimation to calculate the detector coverage.

4.4.5. *Affinity Maturation and Gene Library Evolution*

As described previously, the affinity maturation is the basic feature of an immune response to an antigenic stimulus. Clonal selection and somatic hypermutation are essentially a Darwinian process of selection and variation, guaranteeing high affinity and specificity in non-self recognition in a dynamically changing environment. Computationally, this leads to the development of a new evolutionary algorithm, clonal selection algorithm. This algorithm relies on the input of non-self data (antigens), not the self data required in the negative selection algorithms.

Forrest *et al.* [109] first used genetic algorithm with niching to emulate clone selection. Kim and Bentley [180] embedded the NS algorithm as an operator into Forrest's work. This operator filtered out invalid detectors generated by mutation. Since this algorithm only works on a static dataset, it was named static clonal selection algorithm. Later, the same authors introduced Hofmeyr's lifecycle model to this algorithm to cope with a dynamic environment. This new algorithm was called dynamic clonal selection [177]. Although this algorithm was able to incrementally learn normal behavior by experiencing only a small subset of self samples at one time, it showed high FP errors owing to the infinite lifespan of memory cells. The next step was naturally to define a lifecycle for memory cells. When an antigen detected by a memory cell turned out to be a self-antigen, this memory cell would be deleted. Such a confirmation was equivalent to the co-stimulation signal in Hofmeyr's model [181, 183]. Dasgupta *et al.* also employed the clone selection in their multi-level model [69]. Both mature B-detectors and T-detectors proliferated and were mutated depending on their affinity with antigens.

The clonal selection algorithm implementing affinity maturation is now gradually developed into a new computational paradigm. CLONALG (CLONal selection ALGorithm) [75], ARIS (Artificial Immune Recognition System) [278], and opt-aiNet [72] are well known clonal selection algorithms. These algorithms are used in performing machine-learning and pattern recognition tasks, and solving optimization problems. Although they employ the generation-based model and evolutionary operators when generating offspring, they distinguish themselves from other evolutionary algorithms by the following: firstly, cloning and mutation rates are decided by an individual's affinity. The cloning rate is proportional to the affinity, while the mutation rate is inversely proportional to the affinity. There is no crossover in clonal selection algorithms; secondly, it is a multi-modal preserving algorithm. The memory cell population (P_m) incrementally saves the best solution in each generation. P_m will be returned as the final solution when the algorithm is terminated; thirdly, the population size is dynamically adjustable. Applications of these algorithms to intrusion detection can be found in [123, 204, 205, 283]

In the biological immune system, antibodies are generated by combining fragments from gene libraries. Gene libraries, shaped by evolution, are used to guide the creation process to create antibodies with a good chance of success, while preserving the ability to respond to novel threats [51].

Perelson *et al* [239] and Cayzer *et al.* [50, 51] showed that gene libraries can enhance coverage. Cayzer *et al.*, in addition, investigated the role of gene libraries in AIS [50, 51]. Their empirical experiments suggest that gene libraries in AIS provide combinatorial efficiency, reduce the cost of negative selection, and allow targeting of fixed antigen populations.

Kim and Bentley [182, 183] employed gene library evolution to generate useful antibodies. A problem found in their extended dynamic clonal selection algorithm was that a large number of memory detectors require costimulations in order to maintain low FP rates. Because new detectors were generated randomly, they increase the possibilities of generating invalid detectors. The authors suggested taking feedbacks from previously generated detectors, such as using deleted memory detectors as the virtual gene library. They argued that these deleted memory detectors still held valid information about antibodies, so new detectors were generated by mutating the deleted detectors. Further fine-tuning of these detectors would generate a useful detector with high probabilities.

4.4.6. *Danger Theory*

The fundamental principle that guides the development of AIS is the self non-self discrimination. Immune responses are triggered when the body encounters non-self antigens. Therefore, negative selection acts as an important filter to eliminate autoreactive lymphocytes. However, questions have been raised regarding this classical theory, because it cannot explain transplants, tumors, and autoimmunity, in which some non-self antigens are not eliminated, while some self antigens are destroyed. Matzinger, therefore, proposed the Danger Model [217, 218], and claimed that immune responses are triggered by the unusual death of normal tissues, not by non-self antigens. Unusual death would indicate that there was a dangerous situation.

This theory is still debated within the immunology field. Nevertheless, it provides some fresh ideas that may benefit the design of an AIS. For example, it avoids the scaling problem of generating non-self patterns. Aickelin and his research group started to work on a "Danger Project" [1] in 2003, intended to apply Danger Theory to intrusion detection systems. The authors emphasize the crucial role of the innate immune system for guiding the adaptive immune responses. Their research specifically focuses on building more biologically-realistic algorithms which consider not only adaptive, but also innate immune reactions [17, 18]. Their work so far can be mainly summarized as one innate immunity architecture, and two danger theory based algorithms.

Before we discuss their work, the biological inspiration should be explained in more detail. Danger Theory is based on the difference between healthy and stressed/injured cells. It suggests that cells do not release alarm signals when they die by normally planned processes (known as apoptosis), whereas cells do release alarm signals when they are stressed, injured, or die abnormally (known as necrosis). A type of cells known as Dendritic Cells (DC) act as an important medium, passing the alarm signal to the adaptive immune system. DCs have three distinct states: immature (iDC), semimature (smDC), and mature (mDC). iDCs exist in the extralymphoid compartments, where they function as macrophages: clear the debris of tissue, degrade their proteins into small fragments, and capture alarm signals released from necrose cells using toll-like receptors (TLR). Once iDCs collect debris and are activated by an alarm signal, they differentiate into mDCs, and migrate from the tissue to a lymph node. However, if iDCs do not receive any activation in their lifespan but collect debris, they differentiate into smDCs, and also move to a lymph node. Once in a lymph node, mDCs and smDCs present those fragments collected in the immature stage as antigens at their cell surface using MHC molecules. When a naive T cell in the lymph node binds to these antigens, it will be activated only if the antigens it bonds to are presented by an mDC; it will not response if the antigens are presented by an smDC. This is because mDCs secrete a type of cytokines called IL-12 which activates naive T cells, while smDCs secrete a type of cytokines called IL-10 which suppresses naive T cells. In summary, DCs act as a bridge between the innate and adaptive immune system. They will trigger an adaptive immune response when danger has been detected [134, 135, 274].

From the above discussion, we can see that tissues provide an environment that can be affected by viruses and bacteria, so that signals are sent out and an immune response is initiated. Both Aickelin and Bentley proposed the idea of artificial tissues, because real-world problems sometimes are very difficult to be connected, compared, and mapped to artificial immune algorithms. Similar to the function of tissues, artificial tissues form an intermediate layer between a problem and an artificial immune algorithm, for example, providing data pre-processing for artificial immune algorithms. However, they held different perspectives about artificial tissues.

Bentley *et al.* [38] introduced two tissue growing algorithms for anomaly detection. Artificial tissue grows to form in a specific shape, structure and size in response to specific data samples. When data does not exist to support a tissue, the tissue dies. When too much, or too diverse, data exists for a tissue, the tissue divides. Danger signals are released when a tissue dies. In a sense, artificial tissues provide generic data representations, enabling them to function as an interface between a real-world problem and an artificial immune algorithm. Twycross and Aickelin, on the other hand, proposed a *libtissue* architecture in [273], which allowed researchers to implement, analyze and test new AIS algorithms, as shown in Figure 24. *libtissue* has a client/server architecture. The *libtissue* clients represent the data collected from the monitored systems as antigens and signals, and then transmit them to the *libtissue* server. The client also responds to outputs from the *libtissue* server, and changes the state of the monitored system. On the *libtissue* server, one or more tissue compartments are defined. Compartments provide an environment where immune cells, antigens and signals interact. Immune cells, which are embodied by the artificial immune algorithms, perform analysis and detection. The final decision will be sent back to the client.

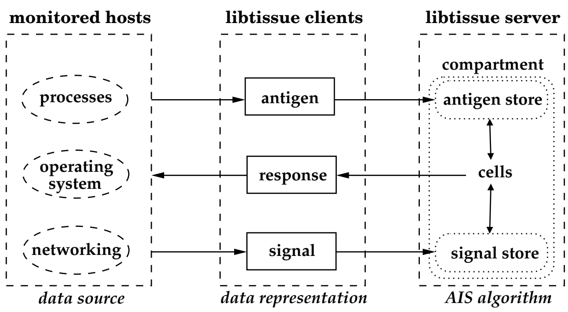


Fig. 24. The architecture of *libtissue* [273].

Another observation from the introduction of the Danger Theory is the role of DCs and their interaction with T cells. Hence, the Dendritic Cell algorithm (DCA) [132, 133, 134, 135, 136, 137] and TLR algorithm (TLRA) [274, 275, 276] were proposed by Greensmith *et al.* and Twycross *et al.*, respectively.

DCA attempts to simulate the power of DCs which are able to activate or suppress immune responses by the correlation of signals representing their environment, combined with the locality markers in the form of antigens [135]. To emulate DCs, Greensmith *et al.* defined four input signals in the DCA: pathogen associated molecular patterns (PAMPs), safe signals, danger signals and inflammatory cytokines [134]. These signals describe the context or environment of an antigen, derived either from input data or the indices of a monitored system, such as CPU usage or errors recorded by log systems. The DCA starts with creating a population of immature DCs. Each iDC collects antigens (i.e, the input data) and signals, and transforms them by an equation to three output concentrations: costimulatory molecules (*csm*), smDC cytokines (*semi*) and mDC cytokines (*mat*). *csm* tracks the maturation of a DC. When this quantity is larger than a pre-defined threshold, this DC is said to be mature. The other two outputs, *semi* and *mat*, will determine if this DC will develop to be an smDC or mDC. Matured DCs are ready for intrusion detection. In summary, the maturation phase in the DCA actually correlates signals and input data to normal or danger contexts. The DCA is deployed in the *libtissue* framework to detect port scan intrusions, specifically ping scans [132, 135] and SYN scans [133]. Kim *et al.* [179] applied this algorithm to detect misbehavior in sensor networks.

TLRA focuses on the interaction between DCs and T cells, which replaces the classical negative selection algorithm. TLRA are completed in a training and test phase. In training, only normal data is presented to DCs. Accordingly, all DCs will develop to smDCs. smDCs in a lymph node will match with randomly generated T cells. If a match happens, which means smDCs activate naive T cells, then these T cells will be killed. In the test phase, anomaly is detected when naive T cells are activated by antigens. Compared to the classical negative selection algorithms, TLRA considers the environment of the input data, not only the antigen itself, thus increasing the detection rate and decreasing the false positive rate. The TLRA was deployed in the *libtissue* framework to detect process anomaly [274, 275, 276]. Kim *et al.* [185] also emulated interactions between DCs and T cells in the CARDINAL (Cooperative Automated worm Response and Detection ImmuNe ALgorithm). However, T cells in CARDINAL will differentiate into various effector T cells, such as helper T cells and cytotoxic T cells. These effector T cells are automated responders that react to worm-related processes. They also exchange information with effector T cells from other hosts when they respond.

In summary, both DCA and TLRA employ the model of DCs, which is an important element in the innate immune system. Experimental results of both algorithms showed good detection rate, thus further confirming that incorporating innate immune response benefits the development of an AIS. The implementation of these two algorithms focuses on the different aspects of the DC model. The DCA relies on the signal processing aspect by using multiple input and output signals, while the TLRA emphasizes the interaction between DCs and T cells, and only uses danger signals. The DCA does not require a training phase; in addition, it depends on few tunable parameters, and is robust to changes in the majority of these parameters. However, choosing good signals should not be trivial, and might affect the performance of both algorithms.

4.4.7. *Summary*

In this section, we reviewed the progress in artificial immune systems and their applications to the intrusion detection domain. The successful protection principles in the human immune system have inspired great interest for developing computational models mimicking similar mechanisms. Reviewing these AIS-based intrusion detection systems or algorithms, we can conclude that the characteristics of an immune system, like uniqueness, distribution, pathogen recognition, imperfect detection, reinforcement learning and memory capacity, compensate for weaknesses of the traditional intrusion detection methods, thus resulting in dynamic, distributed, self-organized and autonomous intrusion detection.

The HIS has a hierarchical structure consisting of various molecules, cells, and organs. Therefore, researchers may have their own perspective when starting to model. Table 7 summarizes the similarities between the approaches.

From this table, evidently NS algorithms are more thoroughly investigated and widely used than other AIS approaches in intrusion detection. This is because NS algorithms lead anomaly detection to a new direction: modeling non-self instead of self patterns. We also notice the quick emergence of Danger Theory, which provides some fresh ideas that benefit the design of AISs. The lifecycle of detectors has been proven as an effective way to avoid holes and adapt to the changes in self data.

Although AIS is a relatively young field, it has received a great deal of attention, and there has been some significant developments recently. Meanwhile, researchers have shown an interest in not only developing systems, but in starting to think more carefully about why and how to develop and apply these immune inspired ideas. As a result, a number of AIS research groups published state-of-the-art reviews of AIS research in 2006 and 2007, attempting to reorganize the research efforts, to clarify terminology confusion and misunderstandings, and to reconsider the immunological metaphors before introducing more new ideas, specifically ones by Dasgupta [67], by Forrest [105], by Ji and Dasgupta [166], by Kim *et al.* [178], and by Timmis [267]. This also implies that anomaly detection is getting more focus.

Despite many successes of AIS-based IDSs, there remain some open questions:

– **Fitting to real-world environments**. Currently most of the algorithms were tested on benchmark datasets. However, real-world environments are far more compli-

Table 7: Summary of Artificial Immune System

HIS		AIS		
Layers	Immune Mechanism	Algorithm	Training Data	Research Work
Adaptive	Negative Selection (T cells and B cells)	Negative Selection	Self	[28][b], [29], [69], [107], [108], [125][a], [126], [129], [159], [162], [165] [160][a], [163], [176], [293], [254], [235], [233], [234], [143], [142], [144]
	Clonal Selection (B cells)	Clonal Selection	Non-self	[180], [177], [182], [181], [175][a], [183], [283], [205], [123], [204]
	Idiotypic Network	Immune Network	Non-self	[203]
	Cell Lifecycle	Detector Lifecycle	Self	[153][a], [152], [33], [119], [280], [146][b], [147], [182], [183]
Innate	Dendritic Cells	DC Algorithm	Self and non-self	[19], [136], [134], [137], [132], [135], [133], [184], [265]
	T Cells and Dendritic Cells	TLR Algorithm	Self	[185], [274], [276], [165], [275][a]

[a] Ph.D Thesis
[b] Master Thesis

cated. Hence, improving the efficiency of the current AIS algorithms is necessary. To take NS algorithms as an example, one needs to consider how to avoid the scaling problem of generating non-self patterns, how to detect and fill holes, how to estimate the coverage of rule sets, and how to deal with a high volume and dimensional data.

- **Adapting to changes in self data**. Normal behavior is constantly changing, and so should normal patterns. Although the concept of a detector's lifecycle contributes to adaption, co-stimulation signals from system administrators are required, which is infeasible in reality. Hence, related mechanisms from the human immune system should be further explored, and carefully mapped to solve anomaly detection problems.
- **Novel and accurate metaphors from immunology**. Current AIS algorithms oversimplify their counterparts in immunology. One needs to carefully exploit all known useful features of immune systems, as well as consider the latest discoveries in immunology. A better understanding of immunology will provide insight into designing completely new models of AIS.
- **Integrating immune responses**. The HIS not only recognizes non-self antigens, but also removes these antigens after recognition. Current AIS-based IDSs focus on self and non-self recognition. Few research so far discussed the response mechanism after detection. A response within an IDS context does not simply mean the generation of an alert, but an implemented change in the system as the result of a detection.

4.5. *Swarm Intelligence*

Swarm Intelligence (SI) is an artificial intelligence technique involving the study of collective behavior in decentralized systems [7]. It computationally emulates the emergent behavior of social insects or swarms in order to simplify the design of distributed solutions to complex problems. Emergent behavior or emergence refers to the way complex systems and patterns arise out of a multiplicity of relatively simple interactions [7]. In the past few years, SI has been successfully applied to optimization, robotics, and military applications. In this section, we will review its contributions into the intrusion detection domain by discussing two swarm motivated research methods.

4.5.1. *Swarm Intelligence Overview*

We can observe various interesting animal behavior in nature. Ants can find the shortest path to the best food source, assign workers to different tasks, or defend a territory from neighbors; A flock of birds flies or a school of fish swims in unison, changing directions in an instant without colliding with each other. These swarming animals exhibit powerful problem-solving abilities with sophisticated collective intelligence.

Swarm intelligence approaches intend to solve complicated problems by multiple simple agents without centralized control or the provision of a global model. Local interactions between agents and their environment often cause a global pattern of behavior to emerge. Hence, emergent strategy and highly distributed control are the two most important features of SI, producing a system autonomous, adaptive, scalable, flexible, robust, parallel, self organizing and cost efficient [231].

Generally speaking, SI models are population-based. Individuals in the population are potential solutions. These individuals collaboratively search for the optimum through iterative steps. Individuals change their positions in the search space, however, via direct or indirect communications, rather than the crossover or mutation operators in evolutionary computation. There are two popular swarm inspired methods in computational intelligence areas: *Ant colony optimization* (ACO) [88] and *particle swarm optimization* (PSO) [174]. ACO simulates the behavior of ants, and has been successfully applied to discrete optimization problems; PSO simulates a simplified social system of a flock of birds or a school of fish, and is suitable for solving nonlinear optimization problems with constraints.

4.5.2. *Ant Colony Optimization*

Ants are interesting social insects. Individual ants are not very intelligent, but ant colonies can accomplish complex tasks unthinkable for individual ants in a self-organized way through direct and indirect interactions. Two types of emergent behavior observed in ant colonies are particularly fascinating: foraging for food and sorting behavior.

A colony of ants can collectively find out where the nearest and richest food source is located, without any individual ant knowing it. This is because ants lay chemical substances called pheromones to mark the selected routes while moving. The concentration of pheromones on a certain path indicates its usage. Paths with a stronger pheromone concentration encourage more ants to follow, thus in turn these additional ants reinforce the concentration of pheromones. Ants who reach the food first by a short path will return to their nest earlier than others, so the pheromones on this path will be stronger than on longer paths. As a result, more ants choose the short path. However, pheromones slowly evaporate over time. The longer path will hold less or even no traces of pheromone after the same time, further increasing the likelihood for ants to choose the short path [231].

Researchers have applied this ant metaphor to solve difficult, discrete optimization problems, including the traveling salesman problem, scheduling problems, the telecommunication network or vehicle routing problem, etc. Its application to the intrusion detection domain is limited but interesting and inspiring. He *et al.* [149] proposed an Ant-classifier algorithm, which is an extension of the Ant-Miner for discovering classification rules [237]. Artificial ants forage paths from the rule antecedents to the class label, thus incrementally discovering the classification rules, as shown in Figure 25. He *et al.* noticed that using only one ant colony to find paths in all classes was inappropriate, because the pheromone level updated by a certain ant would confuse successive ants interested in another class. So more than one colony of ants (i.e. red ants and blue ants in Figure 25) were applied to find solutions for multi-class classification problems simultaneously with each colony to focus on one class. Each colony of ants deposited a different type of pheromone, and ants were only attracted by pheromones deposited by ants in the same colony. In addition, a repulsion mechanism prevented ants of different colonies from choosing the same optimal path. Banerjee *et*

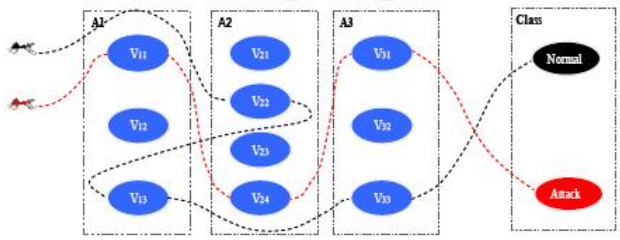


Fig. 25. A multi-class classification algorithm based on multiple ant colonies [149].

al. [34, 35] suggested to use ACO to keep track of intruder trails. The basic idea is to identify affected paths of intrusion in a sensor network by investigating the pheromone concentration. This work also emphasizes the emotional aspect of agents, in that they can communicate the characteristics of particular paths among each other through pheromone updates. Therefore, in a sensor network if the ants are placed, they could keep track the changes in the network path, following certain rules depicting the probabilities of attacks. Once a particular path among nodes is detected by the spy emotional ant, it can communicate the characteristics of that path through pheromone balancing to other ants; thereafter network administrators could be alerted.

In addition to finding the shortest path, ants also exhibit amazing abilities to sort objects. Ants group brood items at similar stages of development (e.g. larvae, eggs, and cocoons) together. In order to do sorting, ants must sense both the type of element they are carrying, and the local spatial density of that type of element. Specifically, each ant must follow some local strategy rules: it wanders a bit; if it meets an object which has a different type of objects around it and if it does not carry one, it takes that object; if it transports an object and sees a similar object in front of it, it deposits the object. By executing these local strategy rules, ants display the ability of performing global sorting and clustering of objects.

Deneubourg *et al.* [79] in 1990 first related this biological observation to an ant-based clustering and sorting algorithm. The basic ant algorithm started with randomly scattering all data items and some ants on a toroidal grid. Subsequently, the sorting phase repeated the previously mentioned local strategy rules. Computationally, the strategy rules can be described as the following: an ant deciding whether to pick up or drop an item i considers the average similarity of i to all items j in its local neighborhood. The local density of similarity ($f(o_i)$) is calculated by Equation 2a, where j denotes the neighborhood of an object o_i; function $d(o_i, o_j)$ measures the similarity of two objects; δ^2 is the size of the local neighborhood; $\alpha \in [0, 1]$ is a data-dependent scaling parameter. The probability of picking up ($P_{pick}(o_i)$) and dropping an object ($P_{drop}(o_i)$) is shown in Equation 2b and Equation 2c, respectively, where k_1 and k_2 are scaling parameter.

$$f(o_i) = max \left\{ 0, \frac{1}{\delta^2} \sum_j (1 - \frac{d(o_i, o_j)}{\alpha}) \right\} \tag{2a}$$

$$P_{pick}(o_i) = (\frac{k_1}{k_1 + f(o_i)})^2 \tag{2b}$$

$$P_{drop}(o_i) = \begin{cases} 2f(o_i) & \text{if } f(o_i) < k_2 \\ 1 & \text{if } f(o_i) \geq k_2 \end{cases} \tag{2c}$$

Romos and Abraham [242] applied this ant-based clustering algorithm to detect intrusion in a network infrastructure. The performance was comparable to the Decision Trees, Support Vector Machines and Linear Genetic Programming. The online processing ability, dealing with new classes, and the self-organizing nature make the ant-based clustering algorithms an ideal candidate for IDSs. Similar work done by Feng *et al.* can also be found at [97, 98, 99].

Tsang and Kwong [269, 270] evaluated the basic ant-based clustering algorithm and an improved version [141] on the KDD99 dataset. They found that these two algorithms suffer from two major problems on clustering large and high dimensional network data. First, many homogeneous clusters are created and are difficult to be merged when they are large in size and spatially separated in a large search space. Second, the density of similarity measures only favors cluster formation in locally dense regions of similar data objects, but cannot discriminate dissimilar objects with any sensitivity. The authors made further improvements on these algorithms, such as combining information entropy and average similarity in order to identify spatial regions of coarse clusters, and to compact clusters and incorrectly merged clusters; cluster formation and object searching were guided by two types of pheromones, respectively; local regional entropy was added to the short-term memory; a tournament selection scheme counterbalanced the population diversity and allowed to find optimal values for control parameters, e.g. α-value, or perception radius. Experiments on the KDD99 dataset showed strong performance in that their algorithm obtained three best and two second best results in five classes, when compared with the KDD99 winner, K-means, [79] and [141].

4.5.3. *Particle Swarm Optimization*

Particle swarm optimization (PSO) is a population based stochastic optimization technique, inspired by social behavior such as bird flocking or fish schooling.

A high-level view of PSO is a collaborative population-based search model. Individuals in the population are called particles, representing potential solutions. The performance of the particles is evaluated by a problem-dependent fitness. These particles move around in a multidimensional searching space. They move toward the best solution (global optimum) by adjusting their position and velocity according to their own experience (local search) or the experience of their neighbors (global search), as shown in Equation 3. In a sense, PSO combines local search and global search to balance exploitation and exploration.

$$\begin{aligned} v_i(t) = {} & w \times v_i(t-1) \\ & + c_1 \times r_1(p_i^l - x_i(t-1)) \\ & + c_2 \times r_2(p_i^g - x_i(t-1)) \end{aligned} \tag{3a}$$

$$x_i(t) = x_i(t-1) + v_i(t) \tag{3b}$$

where $i = 1, 2, \ldots, N$, population size N; $v_i(t)$ represents the velocity of particle i, which implies a distance traveled by i in generation t; $x_i(t)$ represents the position of i in generation t; p_i^l represents the previous best position of i; p_i^g represents the previous best position of the whole swarm; w is the inertia weight which balances the local and global searching pressure;c_1 and c_2 are positive constant acceleration coefficients which control the maximum step size of the particle; r_1 and r_2 are random number in the interval [0,1], and introduce randomness for exploitation.

PSO has shown good performance in solving numeric problems. In the context of intrusion detection, PSO algorithms have been used to learn classification rules. Chen *et al.* [55] demonstrated a "divide-and-conquer" approach to incrementally learning a classification rule set using a standard PSO algorithm. This algorithm starts with a full training set. One run of the PSO is expected to produce the best classifier, which is added to the rule set. Meanwhile, data covered by this classifier is deleted from the training dataset. This process is repeated until the training dataset is empty. Abadeh *et al.* [9] embedded a standard PSO into their fuzzy genetic algorithm. The GA searches for the best individual in every subpopulation. The PSO was applied to the offspring generated by crossover and mutation, aiming to improve the quality of fuzzy rules by searching in their neighborhood. Age was assigned to individuals before the start of local search. Fitter individuals live longer, thus having a longer time to perform local search. In their algorithm, the population consists N subpopulations, where N is the number of classes. Steady-state strategy was employed to update populations.

The classification task usually involves a mixing of both continuous and categorical attribute values. However, a standard PSO does not deal with categorical values: category values do not support the "+" and "-" operations shown in Equation 3. Hence Chen *et al.* mapped category values to integers. The order in mapped sequences sometimes makes no sense in the context of original nominal values, and mathematical operations applied to this artificial order may generate counter-intuitive results. Abadeh *et al.* then redefined the meaning of "+" and "-" operators in Equation 3 by the Rule Antecedent Modification (RAM) operator. The RAM operator can be explained by a simple example. Suppose a linguistic variable R has five fuzzy sets: $\{S, MS, M, ML, L\}$. Antecedent A and B in two particles may contain $\{S, M\}$ and $\{S, L\}$, respectively. $B - A = RAM(2, 3)$, which means B can be converted to A if the 2nd fuzzy set in B is replaced with the 3rd fuzzy set in R. Here $RAM(2, 3)$ is a RAM operator. $B + RAM(2, 3) = A$ means applying RAM operator $RAM(2, 3)$ to B will result in A.

4.5.4. *Summary*

In this section, Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) and their applications to intrusion detection domain were reviewed. They either can be used to discover classification rules for misuse detection, or to discover clusters for anomaly detection, or even can keep track of intruder trails. Experiments results have shown that these approaches achieve equivalent or better performance than traditional methods.

ACO and PSO both have their roots in the study of the behavior of social insects and swarms. Swarms demonstrate incredibly powerful intelligence through simple local interactions of independent agents. Such self-organizing and distributed properties are especially useful for solving intrusion detection problems, which are known for their huge volume and high dimensional datasets, for real-time detection requirement, and for diverse and constantly changing behavior. Swarm Intelligence would offer a way to decompose such a hard problem into several simple ones, each of which is assigned to an agent to work on in parallel, consequently making IDSs autonomous, adaptive, parallel, self organizing and cost efficient.

4.6. *Soft Computing*

Soft computing is an innovative approach to construct a computationally intelligent system which parallels the extraordinary ability of the human mind to reason and learn in an environment of uncertainty and imprecision [289]. Typically, soft computing embraces several computational intelligence methodologies, including artificial neural networks, fuzzy logic, evolutionary computation, probabilistic computing, and recently also subsumed artificial immune systems, belief networks, etc. These members neither are independent of one another nor compete with one another. Rather, they work in a cooperative and complementary way.

The synergism of these methods can be tight or loose. Tightly coupled soft computing systems are also known as hybrid systems. In a hybrid system, approaches are mixed in an inseparable manner. Neuro-fuzzy systems, genetic-fuzzy systems, genetic-neuro systems and genetic-fuzzy-neuro systems are the most visible systems of this type. Comparatively, loosely coupled soft computing systems, or ensemble systems, assemble these approaches together. Each approach can be clearly identified as a module.

In this section, we will discuss how to learn uncertain and imprecise intrusive knowledge using soft computing. Hence, neuro-fuzzy and genetic-fuzzy hybrid approaches are introduced first. The discussion about the genetic-neuro and genetic-fuzzy-neuro hybrid systems can be found in Section 4.3.1.2. The last part of this section will examine the role of ensemble approaches played in intrusion detection.

4.6.1. *Artificial Neural Networks and Fuzzy Systems*

Artificial neural networks model complex relationships between inputs and outputs and try to find patterns in data. Unfortunately, the output models are often not represented in a comprehensible form, and the output values are always crisp. Fuzzy systems, in contrast, have been proven effective when dealing with imprecision and approximate reasoning. However, determining appropriate membership functions and fuzzy rules is often a trial and error process.

Obviously, the fusion of neural networks and fuzzy logic benefits both sides: neural networks perfectly facilitate the process of automatically developing a fuzzy system by their learning and adaptation ability. This combination is called neuro-fuzzy systems; fuzzy systems make ANNs robust and adaptive by translating a crisp output to a fuzzy one. This combination is called fuzzy neural networks (FNN). For example, Zhang *et al.* [294] employed FNNs to detect anomalous system call sequences to decide whether a sequence is "normal" or "abnormal".

Neuro-fuzzy systems are commonly represented as a multilayer feed forward neural network, as illustrated by Figure 26. The neurons in the first layer accept input informa-

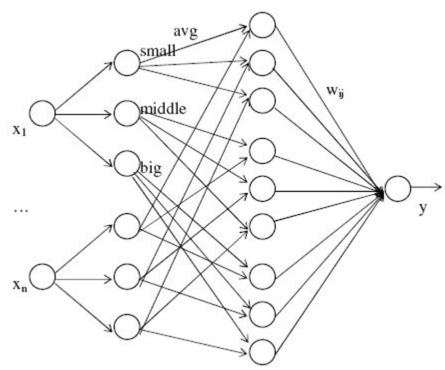


Fig. 26. A generic model of a neuro-fuzzy system [25].

tion. The second layer contains neurons which transform crisp values to fuzzy sets, and output the fuzzy membership degree based on associated fuzzy membership function. Neurons in the third layer represent the antecedent part of a fuzzy rule. Their outputs indicate how well the prerequisites of each fuzzy rule are met. The fourth layer performs defuzzification, and associates an antecedent part with an consequent part of a rule. Sometimes more than one defuzzification layer is used. The learning methods work similarly to that of ANNs. According to the errors between output values and target values, membership functions and weights between reasoning layer and defuzzification layer are adjusted. Through learning, fuzzy rules and membership function will be automatically determined.

Intrusion detection systems normally employ neuro-fuzzy systems for classification tasks. For example, Toosi *et al.* [268] designed an IDS by using five neuro-fuzzy classifiers, each for classifying data from one class in the KDD99 dataset. The neural network was only responsible for further adapting and tuning the membership functions. The number of rules and initial membership functions were determined by a subtractive clustering method. Other similar neuro-fuzzy based IDSs can be found in [25] and [225].

To avoid determining the number of rules before training a ANN, the NEFCLASS system has been introduced. The NEFCLASS system is created from scratch and starts with no rule reasoning layer at all. Rules (neurons in the rule reasoning layer) are created by using of the reinforcement learning algorithm in the first run through the training data (rule learning). In the second run, a fuzzy back propagation algorithm adapts the parameters of membership functions (fuzzy set learning). Hofmann [150] and Alshammari [22] used this method for misuse detection on the DARPA98 and DARPA99 datasets, respectively. Hofmann *et al.* compared the performance of four neural and fuzzy paradigms (multilayer perceptrons, RBF networks, NEFCLASS systems, and classifying fuzzy-k-means) on four attack types. The NEFCLASS is the first runner-up after the RBF. Alshammari *et al.* pointed out that the performance of the NEFCLASS depends on the heuristics' learning factors. Through their experiments they found that a trapezoid membership function using the weight as an aggregation function for the ANN extensively reduces the number of false positive alerts with fewer mistakes. In addition, providing more background knowledge about network traffic provided better results on classification.

Another interesting type of neuro-fuzzy systems is the Fuzzy Cognitive Map (FCM). FCM is a soft computing methodology developed by Kosko as an expansion to cognitive maps which are widely used to represent social scientific knowledge [187]. They are able to incorporate human knowledge, adapt it through learning procedures, and provide a graphical representation of knowledge that can be used for explanation of reasoning. Xin *et al.* [284] and Siraj *et al.* [256, 257] both used FCM to fuse suspicious events to detect complex attack scenarios that involve multiple steps. As Figure 27 shows, suspicious events detected by misuse detection models are mapped to nodes in FCM. The nodes in the FCM are treated as neurons that trigger alerts with different weights depicting on the causal relations between them. So, an alert value for a particular machine or a user is calculated as a function of all the activated suspicious events at a given time. This value reflects the safety level of that machine or user at that time.

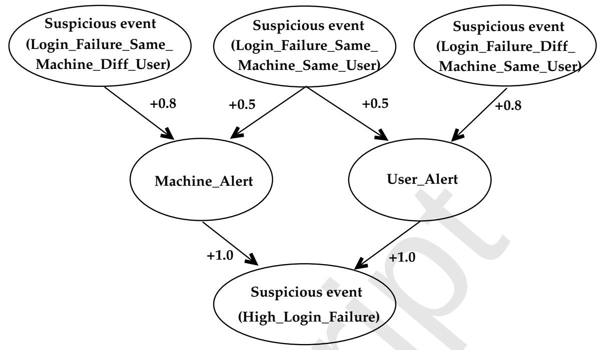


Fig. 27. A FCM to fuse suspicious events to detect complex attack scenarios that involve multiple steps [256].

4.6.2. *Evolutionary Computation and Fuzzy Systems*

Evolutionary computation is another paradigm with learning and adaptive capabilities. Hence, EC became another option for automatically designing and adjusting fuzzy rules. In Section 4.3.1, we discussed how to use EC approaches, especially GAs and GP, to generate crisp rules to classify normal or intrusive behavior. Here, evolving fuzzy rules is as an extension of that research.

Compared with crisp rules, fuzzy rules have the following form:

$$if\ x_1 = A_1\ and\ \ldots\ and\ x_n = A_n\ then\ Class\ C_j\ with\ CF = CF_j$$

where x_i is the attribute of the input data; A_i is the fuzzy set; C_j is the class label; CF_j is the degree of certainty of this fuzzy if-then rule belonging to class C_j.

Technically, evolving fuzzy rules is identical as evolving crisp if-then rules, but with two extra steps. The first step is to determine fuzzy sets and corresponding membership functions for continuous attributes before evolution. Since it is difficult to guarantee that a partition of fuzzy sets for each fuzzy variable is complete and well distinguishable. Therefore, genetic algorithms have been proven [42, 268, 271, 272] useful at tuning membership functions. The second step is to calculate the compatibility grade of each data instance with fuzzy rules either at the fitness evaluation or detection phase. Possibly the same input data instance will trigger more than one fuzzy rule at the same time. The winner-takes-all approach and majority vote are two commonly used techniques to resolve the conflict. Winner refers to the rule with maximum CF_j.

Building models for misuse detection essentially is a multi-class classification problem. Please recall that the crisp classification rules discussed in Section 4.3.1 were evolved in one population, even they have different class labels. Each individual, in a sense, represented only a partial solution to the overall learning task. They cooperatively solve the target problem. Niching was required to maintain the diversity or multimodality in a population. Normally, we call such a method Michigan approach. The XCS mentioned in Section 4.3.1 is an example of this kind. The Pittsburgh approach and the iterative rule learning are another two methods. In the Pittsburgh approach, each individual is a set of rules, representing a complete solution for the target problem. Crossover exchanges rules in two individuals, and mutation creates new rules. The iterative rule learning basically is a divide-and-conquer method. Individuals are defined in the same way as in the Michigan approach. After a pre-defined number of generations, the best classification rule is added to a population which keeps track of the best individuals found so far. The data covered by this best rule is either removed from the training dataset or decreased the probability of being selected again. Work by Chen *et al.* in Section 4.5 explained this method.

Gómez *et al.* first showed evolving fuzzy classifiers for intrusion detection in [120, 121]. Complete binary trees enriched the representation of a GA by using more logic operators, such as "AND", "OR", and "NOT". The authors defined a multi-objective fitness function, which considered sensitivity, specificity and conciseness of rules. Similar ideas were also applied to their negative selection algorithm [122, 130], but the fitness function considered the volume of the subspace represented by a rule and the penalty a rule suffered if it covered normal samples.

Recent work conducted by Tsang *et al.* [271, 272], Abadeh *et al.* [8, 10, 11] and Özyer *et al.* [236] further developed Gómez's research in the following way:

- **Parallel learning**. Tsang *et al.* and Abadeh *et al.* both suggested a parallel learning framework. Tsang *et al.* used multiple fuzzy set agents (FSA) and one arbitrator agent (AA). A FSA constructed and evolved its fuzzy system. The AA evaluated the parent and offspring FSAs by accuracy and interpretability criteria. Abadeh *et al.* [10] divided the training dataset by class labels, and sent subsets to different hosts, where a GA worked on each sub-dataset in parallel.
- **Seeding the initial population**. Instead of generating the initial population randomly, Abadeh *et al.* randomly selected a training data sample, and determined the most compatible combinations of antecedent fuzzy sets. The consequent part was decided by a heuristic method. If the consequent part was consistent with the class label of data samples it covered, then this rule was kept, otherwise the generation process was repeated. Özyer *et al.* [236] ran the fuzzy association rule algorithm first. The strongest association rules were used as seeds to generate the initial population.
- **Representation**. All the research work represent fuzzy if-then rules as string. "don't care" ($*$) symbol is included in their representation as a wild card that allows any possible value in a gene, thus improving the generality of rules.
- **Dynamically changing training data weights**. Abadeh *et al.* [8] and Özyer *et al.* [236] associated a weight to every training sample. Initially, the weights were the same. Weights of misclassified samples remained the same, while weights of correctly classified samples were decreased. Therefore, hard samples had higher probabilities to be exposed in the training algorithms.

These three contributions, of course, were different in many other ways. Mostly, they had different goals. Tsang *et al.* emphasized the importance of interpretability of fuzzy rules; Abadeh *et al.* tried to refine fuzzy rules by using local search operators [10]; Özyer *et al.* integrated boosting genetic fuzzy classifiers and data mining criteria for rule pre-screening. The three work also employed different classifier learning methods. Tsang *et al.* employed the Pittsburgh approach; Abadeh *et al.* [8] the Michigan approach; Özyer *et al.* the iterative learning approach.

4.6.3. *Ensemble Approaches*

Misuse intrusion detection is a very active and well-studied research area. Many classification approaches from artificial intelligence, machine learning, or computational intelligence have been applied to improve detection accuracy, and to reduce false positive errors as well.

However, every approach has its strengths and weaknesses, resulting in various accuracy levels on different classes. The winning entry of the KDD99 cup, for instance, assembled 50×10 C5 decision trees by cost-sensitive bagged boosting. This indicates that even models built by the same algorithm show differences in misclassification.

Abraham and his co-workers, therefore, investigated the possibility of assembling different learning approaches to detect intrusions [14, 16, 15, 54, 229, 238]. Their approach is also known as the ensemble approach. One example of their studies [16] is shown in Figure 28. In this study, they

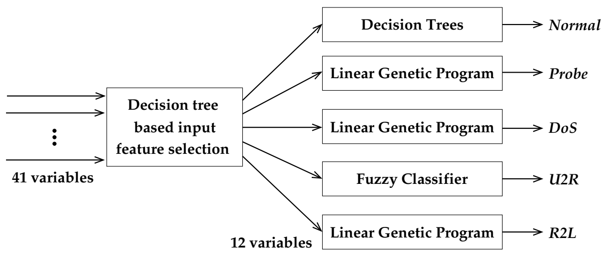


Fig. 28. A exemplar of ensemble models [16].

trained and tested a decision tree model, a linear genetic program model, and a fuzzy classifier model on the KDD99 dataset, respectively. They observed in the experiments that different models provided complementary information about the patterns to be classified. For example, LGP achieved the best accuracy on Probe, DoS and R2L classes, while the fuzzy classifier on the U2R class. So instead of using one model to classify all classes, they selected the best model for each class, and then combined them in a way that both computational efficiency and detection accuracy can be maximized. Sometimes techniques, such as majority vote or winner-takes-all, will be used to decide the output of an ensemble model when the predictions of different models conflict.

4.6.4. *Summary*

Soft computing exploits tolerance for imprecision, uncertainty, low solution cost, robustness, and partial truth to achieve tractability and better correspondence to reality [289]. Their advantages, therefore, boost the performance of intrusion detection systems. Evolutionary computation and artificial neural networks automatically construct fuzzy rules from training data, and present knowledge about intrusion in a readable format; evolutionary computation designs optimal structures of artificial neural networks. These methods in soft computing collectively provide understandable and autonomous solutions to IDS problems. In addition, research has shown the importance of using ensemble approach for modeling IDS. An ensemble helps to combine the synergistic and complementary features of different learning paradigms indirectly, without any complex hybridization. Both the hybrid and ensemble systems indicate the future trends of developing intrusion detection systems.

5. Discussion

Over the past decade intrusion detection based upon computational intelligence approaches has been a widely studied topic, being able to satisfy the growing demand of reliable and intelligent intrusion detection systems.

In our view, these approaches contribute to intrusion detection in different ways. Fuzzy sets represent and process numeric information in a linguistic format, so they make system complexity manageable by mapping a large numerical input space into a smaller search space. In addition, the use of linguistic variables is able to present normal or abnormal behavior patterns in a readable and easy to comprehend format. The uncertainty and imprecision of fuzzy sets smooth the abrupt separation of normal and abnormal data, thus enhancing the robustness of an IDS.

Methods like ANNs, EC, AISs, and SI, are all developed with inspiration from nature. Through the "intelligence" introduced via the biological metaphor, they can infer behavior patterns from data without prior knowledge of regularities in the data. The inference is implemented by either learning or searching. Meanwhile, there remain differences (see also [71]):

- **Structures**. All approaches mentioned are composed of a set of individuals or agents. Individuals are neurons in ANNs; chromosomes in EC; immune cells or molecules in AISs; ants and particles in SI. The collection of these individuals form a network in ANNs; a population in EC; repertories in AISs; colonies and swarms in SI.
- **Performance Evaluation**. The performance of individuals is evaluated. In ANNs, the goal is to minimize the error between actual and desired outputs; in EC and SI, the fitness function defines how good an individual is; in AISs, the goodness of an individual is measured by the affinity between antibodies and antigens.
- **Interactions within the collection** Individuals inside the collection interact with each other. In ANNs, neurons are connected with each other directly. The weights associated with these connections affect the input to a neuron. In the other methods, interaction between individuals is indirect. For example, in AISs, interactions can be the suppression or stimulation within artificial immune networks, or the comparison of affinities between detectors in negative selection and in clonal selection; in SI, ants interact indirectly with pheromone, and particles interact with neighboring particles.
- **Adaptation**. All of these methods demonstrate the ability of adaptation, but in different ways. In EC, adaptation is achieved by evolution. Through crossover and mutation, the genetic composition of an individual can be changed. Selection weeds out poor individuals and conserves fit individuals. As a result, the entire population will converge to an optimum. Similar selection processes are at work in negative and clonal selection in AISs. SI and ANNs achieve adaptation by learning. Weights in ANNs, pheromones in ACO and positions in PSO are updated according to feedback from the environment or from other individuals.

Applications of the above approaches revealed that each has pros and cons. Hence, soft computing either tightly (hybrid) or loosely (ensemble) couples them together in a way that they supplement each other favorably. The resulting synergy has been shown to be an effective way for building IDSs with good accuracy and real-time performance.

We further compared the performance of different CI approaches on solving intrusion detection problems, as shown in Table 8. These research works were trained on either the KDD99-10 or the KDD99 dataset, but were all tested on the KDD99 test dataset. The first five rows in this table record the detection rates obtained by each approach on each class; the last two rows are for the overall detection rate and false positive rate.

From this table, we can easily see that all research work did not perform well on class "U2R" and "R2L", because 11 attack types in these two classes only appear in the test dataset, not the training set; and they constitute more than 50% of the data. However, in general CI approaches achieve better performance than the winning entry which has 50×10 decision trees. This observation confirms that CI approaches possess the characteristics of computational adaptation, fault tolerance, less error prone to noisy information. In particular, transformation functions evolved by GP or LGP (column 6-8) have higher detection rates than evolved classification rules (column 4-5). They especially improved the detection rates on the "U2R" and "R2L". This is because classification rules have limited description power confined by the limited operators, such as "AND", "OR", and "NOT". In addition, rules are more or less a high-level abstraction of data samples. They cannot separate data in two classes very well if the two classes have overlaps. Evolved rules again cannot outperform evolved fuzzy rules (column 10-11). Fuzzy rules obtained noticeable improvement on all classes, which clearly exhibits fuzzy sets are able to increase

Table 8
Performance comparison of various CI approaches on the KDD99 test dataset

Type	Wining Entry	ANN	EC					SI	SC	
			GA		GP	LGP				
	Decision Tree	Hierarchical SOM	XCS	Rules	Transformation Function	LGP	Coevolution	ACO	Fuzzy Sets + EC	
	[92]	[173]	[65]	[104]	[96]	[261]	[200]	[270]	[272]	[268]
Normal	94.5	98.4	95.7	-	99.93	96.5	99.5	98.8	98.3645	98.4
DoS	97.1	96.9	49.1	-	98.81	99.7	97	97.3	97.2017	99.5
Probe	83.3	67.6	93	-	97.29	86.8	71.5	87.5	88.5982	89.2
U2R	13.2	15.7	8.5	-	45.2	76.3	20.7	30.7	15.7895	12.8
R2L	8.4	7.3	3.9	-	80.22	12.35	3.5	12.6	11.0137	27.3
Detection rate	90.9	90.6	-	91.0165	98	94.4	-	-	92.7672	95.3
FP rate	0.45	1.57	-	0.434	0.07	3.5	-	-	-	1.6

the robustness and adaption of IDSs. Transform functions and fuzzy rules achieve similar results, but fuzzy rules are easier to comprehend. The hierarchical SOM in column 3 and the ACO algorithm in column 9 are two unsupervised learning approaches. Since the hierarchical SOM lacks a suitable "boosting" algorithm [173], it cannot beat the ACO algorithm.

In order to have a global picture of research work carried out under the heading of CI, publication statistics according to the year of appearance is given in Figure 29. One can see clearly that the increasing number of research work indicates that IDSs are a growing research area in the computational intelligence field, notably since 2005.

From this figure, a number of trends become obvious in the surveyed work. The first trend we encounter is the popularity of EC. Among 193 papers surveyed, 85 are related to evolutionary computation. Although EC methods were introduced into IDS as early as 1997, they became popular only in recent years. There seems to be a decline in 2006 and 2007, but in fact, the practice of EC in these years merges with fuzzy sets to generate fuzzy classification rules, research classified to be in the SC category. Besides, EC plays an important role in other computational intelligence approaches, such as in negative selection or clonal selection algorithms from AISs. The PSO algorithm does not belong to EC, since no reproduction and selection is involved.

The appearance of SI is another trend. SI is a pretty new research direction for intrusion detection problems. It decomposes a hard problem into several simple sub-problems, assigning agents to work on smaller sub-problems in parallel, thus making IDSs autonomous, adaptive, self organizing and cost efficient. Currently, SI methods are mainly employed to learn classification rules and clusters. More research work in this area is expected in the near future.

We also see a trend to applying SC to intrusion detection problems. Tightly or loosely assembling different methods in a cooperative way definitely improves the performance of an IDS. The most popular combinations are genetic-fuzzy and genetic-neuro systems. The interest in integrating fuzzy sets as a part of these solutions is noticed. In our survey, 23 out of 26 research contributions in SCs utilize fuzzy sets.

Although some promising results have been achieved by current computational intelligence approaches to IDSs, there are still challenges that lie ahead for researchers in this area. First and foremost, good benchmark datasets for network intrusion detection are needed. The KDD99, and the DARPA98&99 datasets are main benchmarks used to evaluate the performance of network intrusion detection systems. However, they are suffering from a fatal drawback: failing to realistically simulate a real-world network [45, 215, 219]. An IDS working well on these datasets may demonstrate unacceptable performance in real environments. In order to validate the evaluation results of an IDS on a simulated dataset, one has to develop a methodology to quantify the similarity of simulated and real network traces, see for instance the research conducted by Brugger [44].

These datasets possess some special characteristics, such as huge volume, high dimension and highly skewed data distribution. Such features can hardly be found in other benchmarks, so they have been widely used for another purpose: challenging and evaluating supervised or unsupervised learning algorithms. However, this purpose is also under criticism [45]. For instance, i) the DARPA datasets include irregularities, such as differences in the TTL for attacks versus normal traffic, so that even a simplistic IDS could achieve a good performance [215], ii) the KDD99 training and test datasets have dissimilar target hypotheses for U2R and R2L classes [246]. Therefore, using these datasets alone is not sufficient to demonstrate the efficiency of a learning algorithm. Other benchmark datasets are recommended to use as well.

It is also worthwhile to note that the datasets shown in Table 1 were collected about 10 years ago. Maybe it is time to produce a new and high-quality dataset for the intrusion detection task. Such a dataset would also be meaningful for machine learning tasks in general. When recollecting data from networks, in addition to storing information

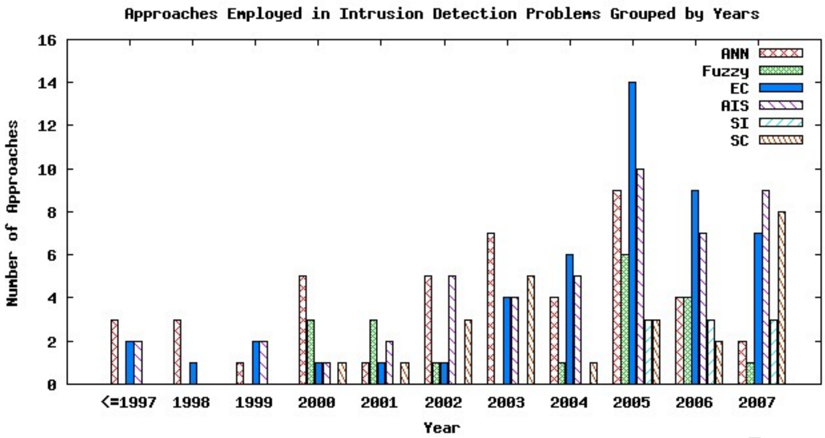


Fig. 29. Publication statistics according to the year of appearance.

in the header of individual packets, payload information [22, 57, 290, 292] and temporal locality property [114, 115] have been proven beneficial.

Secondly, an important aspect of intrusion detection is the ability of adaptation to constantly changing environments. Not only the intrusive behavior evolves continuously, but also the legitimate behavior of users, systems or networks shifts over time. If the IDS is not flexible enough to cope with behavioral changes, detection accuracy will dramatically decrease. Although adaptation is an important issue, only few research has addressed it so far. Recurrent networks introduced context nodes to remember clues from the recent past [21, 47, 48, 57, 76, 78, 114]; in AIS, the lifecycle of immune cells and molecules provides a rolling coverage of non-self space, which guarantees adaptation [153, 183]. The Dendritic Cell Algorithm in Danger theory fulfills adaptation requirements by considering signals from the environment [134, 135]. A focus on adaptation in IDSs is highly recommended.

Another challenge to confront in IDS is the huge volume of audit data that makes it difficult to build an effective IDS. For example, the widely used KDD99 training benchmark comprises about 5,000,000 connection records over a 41-dimensional feature set. Song *et al.* suggested the combination of Random Data Subset Selection and Dynamic Data Subset Selection so that linear genetic programming could process the data within an acceptable time [260, 261]. A similar method is to dynamically adjust the weights of data samples according to classification accuracy, hence changing the probability of data being selected [8, 236]. Other researchers have applied divide-and-conquer algorithms to the dataset. Data that have been classified correctly are removed from the training set. Consequently, the size of the dataset exposed to the learning algorithm shrinks. Another good way to exploit this problem is to utilize a distributed environment. Folin *et al.* [104] and Abadeh *et al.* [11] both examined distributed intrusion detection models, where each node was only assigned part of the data. An ensemble method was used to fuse decisions. Although AISs and SI have properties of self-organization and parallelism, their application to distributed IDS is not thoroughly examined.

Most of the methods discussed in this survey have their roots in the field of biology. However, the analogy between algorithms and their counterpart in biology is still relatively simple. This survey clearly shows that some researchers in this field have begun to apply a more detailed understanding of biology to intrusion detection, for instance the danger theory, swarm intelligence, or advanced topics in evolutionary computation and artificial neural networks. It is expected that new discoveries and a deepened understanding of biology suitable for the intrusion detection task will be the subject of future work.

6. Conclusion

Intrusion detection based upon computational intelligence is currently attracting considerable interest from the research community. Its characteristics, such as adaptation, fault tolerance, high computational speed and error resilience in the face of noisy information, fit the requirement of building a good intrusion detection system.

This paper presents the state-of-the-art in research progress of computational intelligence (CI) methods in intrusion detection systems. The scope of this review was on core methods in CI, including artificial neural networks, fuzzy systems, evolutionary computation methods, artificial immune systems, and swarm intelligence. However, the practice of these methods reveals that each of them has advantages and disadvantages. Soft computing has the power to combine the strengths of these methods in such a way that their disadvantages will be compensated, thus offering better solutions. We therefore included soft computing as a topic in this survey. The contributions of research work in each method are systematically summarized and compared, which allows us to clearly define existing research challenges, and highlight promising new research directions. It is hoped that this survey can serve as a useful guide through the maze of the literature.

7. Acknowledgment

W.B. would like to acknowledge support from NSERC Discovery Grants, under RGPIN 283304-07.

References

[1] Danger Theory Project Website. Retrieved January 26, 2008, from http://www.dangertheory.com/.

[2] The DARPA-Lincoln Dataset. Retrieved January 26, 2008, from http://www.ll.mit.edu/IST/ideval/data/data_index.html.

[3] The Internet Exploration Shootout Dataset. Retrieved January 26, 2008, from http://ivpr.cs.uml.edu/shootout/network.html.

[4] The KDD99 Dataset. Retrieved January 26, 2008, from http://kdd.ics.uci.edu/databases/kddcup99/task.html.

[5] The New Mexico Dataset. Retrieved January 26, 2008, from http://www.cs.unm.edu/~immsec/systemcalls.htm.

[6] The Unix User Dataset. Retrieved January 26, 2008, from http://kdd.ics.uci.edu/databases/UNIX_user_data/UNIX_user_data.html.

[7] Wikipedia. Retrieved January 26, 2008, from http://en.wikipedia.org/.

[8] M. S. Abadeh and J. Habibi. Computer intrusion detection using an iterative fuzzy rule learning approach. In *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '07)*, pages 1–6, London, UK, 23-26 July 2007. IEEE Press.

[9] M. S. Abadeh, J. Habibi, and S. Aliari. Using a particle swarm optimization approach for evolutionary fuzzy rule learning: A case study of intrusion detection. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU '06)*, Paris, France, July 2-7 2006.

[10] M. S. Abadeh, J. Habibi, Z. Barzegar, and M. Sergi. A parallel genetic local search algorithm for intrusion detection in computer networks. *Engineering Applications of Artificial Intelligence*, 20(8):1058–1069, 2007.

[11] M. S. Abadeh), J. Habibi, and C. Lucas. Intrusion detection using a fuzzy genetics-based learning algorithm. *Journal of Network and Computer Applications*, 30(1):414–428, 2007.

[12] A. Abraham and C. Grosan. Evolving intrusion detection systems. In N. Nedjah, A. Abraham, and L. de Macedo Mourelle, editors, *Genetic Systems Programming*, volume 13 of *Studies in Computational Intelligence*, pages 57–79. Springer Berlin / Heidelberg, 2006.

[13] A. Abraham, C. Grosan, and C. Martin-Vide. Evolutionary design of intrusion detection programs. *International Journal of Network Security*, 4(3):328–339, 2007.

[14] A. Abraham and R. Jain. Soft computing models for network intrusion detection systems. In S. K. Halgamuge and L. Wang, editors, *Classification and Clustering for Knowledge Discovery*, volume 4 of *Studies in Computational Intelligence*, chapter 13, pages 191–207. Springer Berlin / Heidelberg, 2005.

[15] A. Abraham, R. Jain, J. Thomas, and S. Y. Han. D-SCIDS: Distributed soft computing intrusion detection system. *Journal of Network and Computer Applications*, 30(1):81–98, 2007.

[16] A. Abraham and J. Thomas. Distributed intrusion detection systems:a computational intelligence approach. In H. Abbass and D. Essam, editors, *Applications of Information Systems to Homeland Security and Defense*, chapter 5, pages 105–135. Idea Group Inc, USA, 2005.

[17] U. Aickelin, P. Bentley, S. Cayzer, J. Kim, and J. McLeod. Danger theory: The link between AIS and IDS? In J. Timmis, P. J. Bentley, and E. Hart, editors, *Artificial Immune Systems*, volume 2787 of *Lecture Notes in Computer Science*, pages 147–155. Springer Berlin / Heidelberg, 2003.

[18] U. Aickelin and S. Cayzer. The danger theory and its application to artificial immune systems. In J. Timmis and P. J. Bentley, editors, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS '02)*, pages 141–148, Canterbury, UK, 9-11 September 2002. Unversity of Kent at Canterbury Printing Unit.

[19] U. Aickelin and J. Greensmith. Sensing danger: Innate immunology for intrusion detection. In *Information Security Technical Report , ISSN 1363-4127 (In Press)*. ELSEVIER, 2007. Retrieved January 26, 2008, from http://eprints.nottingham.ac.uk/392/.

[20] U. Aickelin, J. Greensmith, and J. Twycross. Immune system approaches to intrusion detection:a review. In G. Nicosia, V. Cutello, P. J. Bentley, and J. Timmis, editors, *Artificial Immune Systems*, volume 3239 of *Lecture Notes in Computer Science*, pages 316–329. Springer Berlin / Heidelberg, 2004.

[21] M. Al-Subaie and M. Zulkernine. The power of temporal pattern processing in anomaly intrusion detection. In *IEEE International Conference on Communications (ICC '07)*, pages 1391–1398, Glasgow, Scotland, 24-28 June 2007. IEEE Press.

[22] R. Alshammari, S. Sonamthiang, M. Teimouri, and D. Riordan. Using neuro-fuzzy approach to reduce false positive alerts. In *Fifth Annual Conference on Communication Networks and Services Research (CNSR '07)*, pages 345–349. IEEE Computer Society, May 2007.

[23] M. Amini and R. Jalili. Network-based intrusion detection using unsupervised adaptive resonance theory. In *Proceedings of the 4th Conference on Engineering of Intelligent Systems (EIS '04)*, Madeira, Portugal, 2004.

[24] M. Amini, R. Jalili, and H. R. Shahriari. RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks. *Computers & Security*, 25(6):459–468, 2006.

[25] J. An, G. Yue, F. Yu, and R. Li. Intrusion detection based on fuzzy neural networks. In J. Wang, ZhangYi, J. M. Zurada, B.-L. Lu, and H. Yin, editors, *Advances in Neural Networks - Third International Symposium on Neural Networks (ISNN '06)*, volume 3973 of *Lecture Notes in Computer Science*, pages 231–239. Springer Berlin / Heidelberg, 2006.

[26] K. P. Anchor, P. Williams, G. Gunsch, and G. Lamont. The computer defense immune system: current and future research in intrusion detection. In D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '02)*, volume 2, pages 1027–1032, Honolulu, HI, USA, 12-17 May 2002. IEEE Press.

[27] M. Ayara, J. Timmis, R. de Lemos, L. N. de Castro, and R. Duncan. Negative selection: How to generate detectors. In J. Timmis and P. J. Bentley, editors, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS '02)*, pages 89–98, Canterbury, UK, 9-11 September 2002. Unversity of Kent at Canterbury Printing Unit.

[28] S. Balachandran. Multi-shaped detector generation using real-valued representation for anomaly detection. Master's thesis, The University of Memphis, Memphis, Tennessee, December 2005.

[29] S. Balachandran, D. Dasgupta, F. Niño, and D. Garrett. A general framework for evolving multi-shaped detectors in negative selection. In *IEEE Symposium on Foundations of Computational Intelligence (FOCI '07)*, pages 401–408, Honolulu, HI, USA, 1-5 April 2007. IEEE Computer Society.

[30] S. Balachandran, D. Dasgupta, and L. Wang. A hybrid approach for misbehavior detection in wireless ad-hoc networks. In *Symposium on Information Assurance*, New York, USA, 14-15 June 2006.

[31] B. Balajinath and S. V. Raghavan. Intrusion detection through learning behavior model. *Computer Communications*, 24(12):1202–1212, 2001.

[32] J. Balthrop, F. Esponda, S. Forrest, and M. Glickman. Coverage and generalization in an artificial immune system. In W. B. L. et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02)*, pages 3–10, New York, USA, 9-13 July 2002. Morgan Kaufmann.

[33] J. Balthrop, S. Forrest, and M. R. Glickman. Revisiting LISYS: Parameters and normal behavior. In D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '02)*, volume 2, pages 1045–1050, Honolulu, HI, USA, 12-17 May 2002. IEEE Press.

[34] S. Banerjee, C. Grosan, and A. Abraham. IDEAS: intrusion detection based on emotional ants for sensors. In *Proceedings of 5th International Conference on Intelligent Systems Design and Applications (ISDA '05)*, pages 344–349, Wroclaw, Poland, 8-10 September 2005. IEEE Computer Society, Washington, DC, USA.

[35] S. Banerjee, C. Grosan, A. Abraham, and P. Mahanti. Intrusion detection on sensor networks using emotional ants. *International Journal of Applied Science and Computations*, 12(3):152–173, 2005.

[36] Z. Bankovic, D. Stepanovic, S. Bojanica, and O. Nieto-Taladriz. Improving network security using genetic algorithm approach. *Computers & Electrical Engineering*, 33(5-6):438–451, 2007. Security of Computers & Networks.

[37] W. Banzhaf, P. Nordin, R. Keller, and F. Francone. *Genetic Programming - An Introduction.* Academic Press / Morgan Kaufmann, San Francisco, CA, 1998.

[38] P. J. Bentley, J. Greensmith, and S. Ujjin. Two ways to grow tissue for artificial immune systems. In C. Jacob, M. L. Pilat, P. J. Bentley, and J. Timmis, editors, *Artificial Immune Systems*, volume 3627 of *Lecture Notes in Computer Science*, pages 139–152. Springer Berlin/Heidelberg, 2005.

[39] J. C. Bezdek. What is computational intelligence? *Computational Intelligence Imitating Life*, pages 1–12, 1994. IEEE Press, New York.

[40] A. Bivens, C. Palagiri, R. Smith, B. Szymanski, and M. Embrechts. Networkbased intrusion detection using neural networks. *Intel-*

ligent Engineering Systems through Artificial Neural Networks, 12(1):579–584, 2002.
[41] M. Brameier and W. Banzhaf. *Linear Genetic Programming*. Springer, New York, 2007.
[42] S. M. Bridges and R. B. Vaughn. Fuzzy data mining and genetic algorithms applied to intrusion detection. In *Proceedings of the 23rd National Information Systems Security Conference*, pages 13–31, Baltimore, MA, USA, 16-19 October 2000.
[43] S. M. Bridges and R. B. Vaughn. Intrusion detection via fuzzy data mining. In *Proceedings of the 12 thAnnual Canadian Information Technology Security Symposium*, pages 111–121, 2000.
[44] S. T. Brugger. The quantitative comparison of ip networks. Technical report, University of California, Davis, 2007. Retrieved January 26, 2008, from `http://bruggerink.com/~zow/GradSchool/brugger_netcompare_thesis.pdf`.
[45] T. Brugger. KDD cup '99 dataset (network intrusion) considered harmful, 15 Sep 2007. Retrieved January 26, 2008, from `http://www.kdnuggets.com/news/2007/n18/4i.html`.
[46] J. Cannady. Artificial neural networks for misuse detection. In *Proceedings of the 21st National Information Systems Security Conference*, pages 368–381, Arlington, VA, USA, 5-8 October 1998.
[47] J. Cannady. Applying CMAC-based on-line learning to intrusion detection. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '00)*, volume 5, pages 405–410, Como, Italy, 24-27 July 2000. IEEE Press.
[48] J. Cannady. Next generation intrusion detection: Autonomous reinforcement learning of network attacks. In *Proceedings of the 23rd National Information Systems Security Conference*, pages 1–12, Baltimore, MA, USA, 16-19 October 2000.
[49] J. Cannady and J. Mahaffey. The application of artificial neural networks to misuse detection: Initial results. In *Proceedings of the 1st International Workshop on Recent Advances in Intrusion Detection (RAID 98)*, Louvain-la-Neuve, Belgium, 14-16 September 1998.
[50] S. Cayzer and J. Smith. Gene libraries: Coverage, efficiency and diversity. In H. Bersini and J. Carneiro, editors, *Artificial Immune Systems*, volume 4163 of *Lecture Notes in Computer Science*, pages 136–149. Springer Berlin/Heidelberg, 2006.
[51] S. Cayzer, J. Smith, J. A. Marshall, and T. Kovacs. What have gene libraries done for AIS? In C. Jacob, M. L. Pilat, P. J. Bentley, and J. Timmis, editors, *Artificial Immune Systems*, volume 3627 of *Lecture Notes in Computer Science*, pages 86–99. Springer Berlin/Heidelberg, 2005.
[52] A. P. F. Chan, W. W. Y. Ng, D. S. Yeung, and E. C. C. Tsang. Comparison of different fusion approaches for network intrusion detection using ensemble of RBFNN. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3846–3851. IEEE Press, 18-21 Aug. 2005.
[53] S. Chavan, K. Shah, N. Dave, S. Mukherjee, A. Abraham, and S. Sanyal. Adaptive neuro-fuzzy intrusion detection systems. In *IEEE International Conference on Information Technology: Coding and Computing (ITCC'04)*, volume 1, pages 70–74. IEEE Computer Society, 2004.
[54] S. Chebrolu, A. Abraham, and J. P. Thomas. Feature deduction and ensemble design of intrusion detection systems. *Computers & Security*, 24(4):295–307, 2005.
[55] G. Chen, Q. Chen, and W. Guo. A PSO-based approach to rule learning in network intrusion detection. In B.-Y. Cao, editor, *Fuzzy Information and Engineering*, volume 40 of *Advances in Soft Computing*, pages 666–673. Springer Berlin / Heidelberg, 2007.
[56] Y. Chen, J. Zhou, and A. Abraham. Estimation of distribution algorithm for optimization of neural networks for intrusion detection system. In L. Rutkowski, R. Tadeusiewicz, L. A. Zadeh, and J. Zurada, editors, *The 8th International Conference on Artificial Intelligence and Soft Computing (ICAISC '06)*, volume 4029 of *Lecture Notes in Computer Science*, pages 9–18. Springer Berlin / Heidelberg, 2006.
[57] E. Cheng, H. Jin, Z. Han, and J. Sun. Network-based anomaly detection using an elman network. In X. Lu and W. Zhao, editors, *Networking and Mobile Computing*, volume 3619 of *Lecture Notes in Computer Science*, pages 471–480. Springer Berlin / Heidelberg, 2005.
[58] W. Chimphlee, A. H. Abdullah, M. N. M. Sap, S. Chimphlee, and S. Srinoy. Unsupervised clustering methods for identifying rare events in anomaly detection. In *6 th Internation Enformatika Conference (IEC '05)*, 26-28 October 2005.
[59] W. Chimphlee, A. H. Abdullah, M. N. M. Sap, S. Srinoy, and S. Chimphlee. Anomaly-based intrusion detection using fuzzy rough clustering. In *International Conference on Hybrid Information Technology (ICHIT '06)*, volume 1, pages 329–334, 2006.
[60] W. Chimphlee, M. N. M. Sap, A. H. Abdullah, S. Chimphlee, and S. Srinoy. To identify suspicious activity in anomaly detection based on soft computing. In *Proceedings of the 24th IASTED international conference on Artificial intelligence and applications*, pages 359–364, Innsbruck, Austria, 2006.
[61] A. Chittur. Model generation for an intrusion detection system using genetic algorithms. Technical report, High School Honors Thesis, Ossining High School. In cooperation with Columbia Univ., 2002.
[62] S.-B. Cho. Incorporating soft computing techniques into a probabilistic intrusion detection system. *IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews*, 32(2):154–160, 2002.
[63] B. Craenen and A. Eiben. Computational intelligence. Encyclopedia of Life Support Sciences, EOLSS; EOLSS Co. Ltd., 2002.
[64] M. Crosbie and E. H. Spafford. Applying genetic programming to intrusion detection. In E. V. Siegel and J. R. Koza, editors, *Working Notes for the AAAI Symposium on Genetic Programming*, pages 1–8, MIT, Cambridge, MA, USA, 10-12 Nov. 1995. AAAI.
[65] H. H. Dam, K. Shafi, and H. A. Abbass. Can evolutionary computation handle large dataset? In S. Zhang and R. Jarvis, editors, *AI 2005: Advances in Artificial Intelligence- 18th Australian Joint Conference on Artificial Intelligence, Sydney, Australia, 5-9 December, 2005*, volume 3809 of *Lecture notes in computer science*, pages 1092–1095. Springer Berlin / Heidelberg, 2005.
[66] D. Dasgupta. Immunity-based intrusion detection system: A general framework. In *Proceedings of the 22nd National Information Systems Security Conference*, pages 147–160, Arlington, VA, USA, 18-21 October 1999.
[67] D. Dasgupta. Advances in artificial immune systems. *IEEE Computational Intelligence Magazine*, 1(4):40–49, 2006.
[68] D. Dasgupta and F. Gonzalez. An immunity-based technique to characterize intrusions in computer networks. *IEEE Transactions on Evolutionary Computation*, 6(3):281–291, 2002.
[69] D. Dasgupta, S. Yu, and N. Majumdar. MILA-multilevel immune learning algorithm and its application to anomaly detection. *Soft Computing Journal*, 9(3):172–184, 2005.
[70] M. Dass. LIDS: A learning intrusion detection system. Master of science, The University of Georgia, Athens, Georgia, 2003.
[71] L. N. de Castro. Immune, swarm, and evolutionary algorithms, part II: Philosophical comparisons. In L. Wang, J. C. Rajapakse, K. Fukushima, S.-Y. Lee, and X. Yao, editors, *Proceedings of the International Conference on Neural Information Processing (ICONIP '02), Workshop on Artificial Immune Systems*, volume 3, pages 1469–1473. IEEE Press, 18-22 Nov. 2002.
[72] L. N. de Castro and J. I. Timmis. An artificial immune network for multimodal function optimization. In D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '02)*, volume 1, pages 699–674, Honolulu, HI, USA, 12-17 May 2002. IEEE Press.
[73] L. N. de Castro and J. I. Timmis. Artificial immune systems as a novel soft computing paradigm. *Soft Computing*, 7(8):526–544, 2003.
[74] L. N. de Castro and F. J. V. Zuben. Artificial immune systems: Part I - basic theory and applications. Technical Report TR - DCA 01/99, The Catholic University of Santos, Brazil, December 1999.
[75] L. N. de Castro and F. J. V. Zuben. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems*, 6(3):239–251, 2002.
[76] H. Debar, M. Becker, and D. Siboni. A neural network component for an intrusion detection system. In *Proceedings of 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 240–250, Oakland, CA, USA, 4-6 May 1992. IEEE Press.
[77] H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822, 1999.
[78] H. Debar and B. Dorizzi. An application of a recurrent network to an intrusion detection system. In *Proceeding of the International Joint Conference on Neural Networks (IJCNN 92)*, volume 2, pages 478–483. IEEE Computer Society, 7-11 June 1992.
[79] J. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien. The dynamics of collective sorting: Robot-like ants and ant-like robots. In J. A. Meyer and S. Wilson, editors, *Proceedings of the First International Conference on Simulation of Adaptive Behaviour: From Animals to Animats*, volume 1, pages 356–365. MIT Press, Cambridge, MA, USA, 1991.
[80] D. E. Denning. An intrusion detection model. *IEEE Transactions on Software Engineering, Special issue on computer security and privacy*, 13(2):222–232, 1987.
[81] P. Dhaeseleer, S. Forrest, and P. Helman. An immunological approach to change detection: Algorithms, analysis and implications. In *Proceedings of 1996 IEEE Symposium onSecurity and Privacy*, pages 110–119, Oakland, CA, USA, 6-8 May 1996. IEEE Computer Society.
[82] P. A. Diaz-Gomez and D. F. Hougen. Analysis and mathematical justification of a fitness function used in an intrusion detection system. In H.-G. Beyer and U.-M. O'Reilly, editors, *Proceedings of*

the Genetic and Evolutionary Computation Conference (GECCO '05), pages 1591–1592, Washington, D.C., USA, 25-29 June 2005. ACM.

[83] P. A. Diaz-Gomez and D. F. Hougen. Analysis of an off-line intrusion detection system:a case study in multi-objective genetic algorithms. In I. Russell and Z. Markov, editors, *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference*, pages 822–823, Clearwater Beach, FL, USA, 2005. AAAI Press.

[84] P. A. Diaz-Gomez and D. F. Hougen. Improved off-line intrusion detection using a genetic algorithm. In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 66–73, 2005.

[85] P. A. Diaz-Gomez and D. F. Hougen. A genetic algorithm approach for doing misuse detection in audit trail files. In *The 15th International Conference on Computing (CIC '06)*, pages 329–338. IEEE Computer Society, Nov. 2006.

[86] J. E. Dickerson and J. A. Dickerson. Fuzzy network profiling for intrusion detection. In *Proceedings of the 19th International Conference of the North American Fuzzy Information Society (NAFIPS '00)*, pages 301–306, Atlanta, GA, USA, 13-15 July 2000. IEEE Press.

[87] J. E. Dickerson, J. Juslin, O. Koukousoula, and J. A. Dickerson. Fuzzy intrusion detection. In *Proceedings of the 20th International Conference of the North American Fuzzy Information Society (NAFIPS '01) and Joint the 9th IFSA World Congress*, volume 3, pages 1506–1510, Vancouver, Canada, 25-28 July 2001. IEEE Press.

[88] M. Dorigo. *Optimization, learning and natural algorithms (in italian)*. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

[89] W. Duch. What is computational intelligence and where is it going? In W. Duch and J. Mańdziuk, editors, *Challenges for Computational Intelligence*, volume 63 of *Studies in Computational Intelligence*, pages 1–13. Springer Berlin / Heidelberg, 2007.

[90] N. A. Durgin and P. Zhang. Profile-based adaptive anomaly detection for network security. Technical report, Sandia National Laboratories, 2005.

[91] A. El-Semary, J. Edmonds, J. Gonzalez, and M. Papa. A framework for hybrid fuzzy logic intrusion detection systems. In *The 14th IEEE International Conference on Fuzzy Systems (FUZZ '05)*, pages 325–330, Reno, NV, USA, 25-25 May 2005. IEEE Press.

[92] C. Elkan. Results of the KDD '99 classifier learning. *ACM SIGKDD Explorations Newsletter*, 1:63–64, 2000.

[93] F. Esponda, S. Forrest, and P. Helman. The crossover closure and partial match detection. In J. Timmis, P. J. Bentley, and E. Hart, editors, *Artificial Immune Systems*, volume 2787 of *Lecture Notes in Computer Science*, pages 249–260. Springer Berlin / Heidelberg, 2003.

[94] F. Esponda, S. Forrest, and P. Helman. A formal framework for positive and negative detection schemes. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, 34(1):357–373, 2004.

[95] W. Fan, M. Miller, S. Stolfo, W. Lee, and P. Chan. Using artificial anomalies to detect unknown and known network intrusions. *Knowledge and Information Systems*, 6(5):507–527, 2004.

[96] K. Faraoun and A. Boukelif. Genetic programming approach for multi-category pattern classification applied to network intrusions detection. *International Journal Of Computational Intelligence And Applications*, 3(1):77–90, 2006.

[97] Y. Feng, Z. Wu, K. Wu, Z. Xiong, and Y. Zhou. An unsupervised anomaly intrusion detection algorithm based on swarm intelligence. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, volume 7, pages 3965–3969. IEEE Computer Society, 18-21 Aug. 2005.

[98] Y. Feng, J. Zhong, Z. Xiong, C. xiao Ye, and K. gui Wu. Network anomaly detection based on dsom and aco clustering. In D. Liu, S. Fei, Z. Hou, H. Zhang, and C. Sun, editors, *Advances in Neural Networks (ISNN 2007)*, volume 4492 of *Lecture Notes in Computer Science*, pages 947–955. Springer Berlin / Heidelberg, 2007.

[99] Y. Feng, J. Zhong, C. Ye, and Z. Wu. Clustering based on self-organizing ant colony networks with application to intrusion detection. In S. Ceballos, editor, *Proceedings of 6th International Conference on Intelligent Systems Design and Applications (ISDA '06)*, volume 6, pages 3871– 3875, Jinan, China, 16-18 October 2006. IEEE Computer Society, Washington, D.C., USA.

[100] C. Ferreira. Gene expression programming: A new adaptive algorithm for solving problems. *Complex Systems*, 13(2):87–129, 2001.

[101] G. Florez, S. M. Bridges, and R. B. Vaughn. An improved algorithm for fuzzy data mining for intrusion detection. In *Proceedings of the 21st International Conference of the North American Fuzzy Information Society (NAFIPS '02)*, pages 457–462, New Orleans, LA, USA, June 27-29 2002. IEEE Press.

[102] D. B. Fogel. What is evolutionary computation? *IEEE Spectrum*, 37(2):26, 28–32, 2000.

[103] G. Folino, C. Pizzuti, and G. Spezzano. An evolutionary ensemble approach for distributed intrusion detection. In *International Conference on Artificial Evolution (EA '05)*, University of Lille, France., 26-28 October 2005.

[104] G. Folino, C. Pizzuti, and G. Spezzano. GP ensemble for distributed intrusion detection systems. In S. Singh, M. Singh, C. Apté, and P. Perner, editors, *Pattern Recognition and Data Mining, Third International Conference on Advances in Pattern Recognition (ICAPR '05), Bath, UK, August 22-25, 2005, Proceedings, Part I*, volume 3686 of *Lecture Notes in Computer Science*, pages 54–62. Springer Berlin / Heidelberg, 2005.

[105] S. Forrest and C. Beauchemin. Computer immunology. *Immunological Reviews*, 216(1):176–197, 2007.

[106] S. Forrest, S. Hofmeyr, and A. Somayaji. Computer immunology. *Communications of the ACM*, 40(10):88–96, 1997.

[107] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff. A sense of self for Unix processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 120–128, Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[108] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *Proceedings of 1994 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 202–212, Oakland, CA, USA, 16-18 May 1994. IEEE Press.

[109] S. Forrest, R. Smith, B. Javornik, and A. Perelson. Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1(3):191–211, 1993. MIT Press Cambridge, MA, USA.

[110] K. Fox, R. Henning, and J. Reed. A neural network approach toward intrusion detection. In *Proceedings of the 13th National Computer Security Conference*, volume 1, pages 124–134, Washington, D.C., USA, 1-4 Oct. 1990.

[111] A. A. Freitas and J. Timmis. Revisiting the foundations of artificial immune systems: A problem-oriented perspective. In J. Timmis, P. J. Bentley, and E. Hart, editors, *Artificial Immune Systems*, volume 2787 of *Lecture Notes in Computer Science*, pages 229–241. Springer Berlin / Heidelberg, 2003.

[112] J. C. Galeano, A. Veloza-Suan, and F. A. González. A comparative analysis of artificial immune network models. In H.-G. Beyer and U.-M. O'Reilly, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pages 361–368, Washington, D.C., USA, 25-29 June 2005. ACM.

[113] S. M. Garrett. How do we evaluate artificial immune systems? *Evolutionary Computation*, 13(2):145–177, 2005.

[114] A. K. Ghosh, C. Michael, and M. Schatz. A real-time intrusion detection system based on learning program behavior. In H. Debar, L. Mé, and S. F. Wu, editors, *Proceedings of the 3rd International Workshop on Recent Advances in Intrusion Detection (RAID '00), Toulouse, France, 2-4 October, 2000*, volume 1907 of *Lecture Notes in Computer Science*, pages 93–109. Springer Berlin / Heidelberg, 2000.

[115] A. K. Ghosh and A. Schwartzbard. A study in using neural networks for anomaly and misuse detection. In *Proceedings of the 8th USENIX Security Symposium*, volume 8, pages 141–152, Washington, D.C., USA, 23-36 August 1999.

[116] A. K. Ghosh, J. Wanken, and F. Charron. Detecting anomalous and unknown intrusions against programs. In *Proceedings of the 14th Annual Computer Security Applications Conference (ACSAC '98)*, pages 259–267, Phoenix, AZ, USA, 7-11 December 1998. IEEE Computer Society.

[117] A. Giordana, F. Neri, and L. Saitta. Search-intensive concept induction. *Evolutionary Computation*, 3(4):375–416, 1995.

[118] L. Girardin. An eye on network intruder-administrator shootouts. In *Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring*, pages 19–28, Santa Clara, CA, USA, 9-12 April 1999. USENIX Association, Berkeley, CA, USA.

[119] M. Glickman, J. Balthrop, and S. Forrest. A machine learning evaluation of an artificial immune system. *Evolutionary Computation*, 13(2):179–212, 2005.

[120] J. Gómez and D. Dasgupta. Complete expression trees for evolving fuzzy classifier systems with genetic algorithms and application to network intrusion detection. In *Proceedings of the 21st International Conference of the North American Fuzzy Information Society (NAFIPS '02)*, pages 469–474, New Orleans, LA, USA, June 27-29 2002. IEEE Press.

[121] J. Gómez and D. Dasgupta. Evolving fuzzy classifiers for intrusion detection. In *Proceedings of the 2002 IEEE Workshop on Information Assurance*, United States Military Academy, West Point, NY, USA, June 2002. IEEE Press.

[122] J. Gómez, F. GonzAlez, and D. Dasgupta. An immuno-fuzzy approach to anomaly detection. In *The 12th IEEE International Conference on Fuzzy Systems (FUZZ '03)*, volume 2, pages 1219–1224, St.Louis, MO, USA, 25-28 May 2003. IEEE Press.

[123] M. Gong, H. Du, L. Jiao, and L. Wang. Immune clonal selection algorithm for multiuser detection in DS-CDMA systems. In G. I.Webb and XinghuoYu, editors, *AI 2004: Advances in Artifi-*

cial Intelligence, volume 3339 of *Lecture Notes in Computer Science*, pages 1219–1225. Springer Berlin / Heidelberg, 2004.

[124] R. H. Gong, M. Zulkernine, and P. Abolmaesumi. A software implementation of a genetic algorithm based approach to network intrusion detection. In *The sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005 and the First ACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN '05)*, pages 246 – 253. IEEE Computer Society Washington, D.C., USA, 2005.

[125] F. González. *A Study of Artificial Immune Systems Applied to Anomaly Detection.* PhD thesis, The University of Memphis, 2003.

[126] F. González and D. Dasgupta. Anomaly detection using real-valued negative selection. *Genetic Programming and Evolvable Machines*, 4(4):383–403, 2003.

[127] F. González, D. Dasgupta, and J. Gomez. The effect of binary matching rules in negative selection. In E. C.-P. et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '03), Part I, Chicago, IL, USA, 12-16 July, 2003*, volume 2723 of *Lecture Notes in Computer Science*, pages 195–206. Springer Berlin / Heidelberg, 2003.

[128] F. González, D. Dasgupta, and R. Kozma. Combining negative selection and classification techniques for anomaly detection. In D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '02)*, volume 1, pages 705–710, Honolulu, HI, USA, 12-17 May 2002. IEEE Press.

[129] F. González, D. Dasgupta, and L. F. Niño. A randomized real-valued negative selection algorithm. In J. Timmis, P. J. Bentley, and E. Hart, editors, *Proceedings of the 2nd International Conference on Artificial Immune Systems (ICARIS '03), Edinburgh, UK, 1-3 September, 2003*, volume 2787 of *Lecture Notes in Computer Science*, pages 261–272. Springer Berlin / Heidelberg, 2003.

[130] F. González, J. Gómez, M. Kaniganti, and D. Dasgupta. An evolutionary approach to generate fuzzy anomaly signatures. In *Proceedings of the 4th Annual IEEE Systems, Man and Cybernetics Society Information Assurance Workshop*, pages 251–259, West Point, NY, USA, 18-20 June 2003. IEEE Press.

[131] L. J. González and J. Cannady. A self-adaptive negative selection approach for anomaly detection. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '04)*, volume 2, pages 1561–1568, Portland, OR, USA, 19-23 June 2004. IEEE Press.

[132] J. Greensmith and U. Aickelin. Dendritic cells for real-time anomaly detection. In *Proceedings of the Workshop on Artificial Immune Systems and Immune System Modelling (AISB '06)*, pages 7–8, Bristol, UK, 2006.

[133] J. Greensmith and U. Aickelin. Dendritic cells for syn scan detection. In H. Lipson, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '07)*, pages 49–56, London, England, UK, 7-11July 2007. ACM.

[134] J. Greensmith, U. Aickelin, and S. Cayzer. Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In C. Jacob, M. L. Pilat, P. J. Bentley, and J. Timmis, editors, *Proceedings of the 4th International Conference on Artificial Immune Systems (ICARIS '05), Banff, Alberta, CA, 14-17 August, 2005*, volume 3627 of *Lecture Notes in Computer Science*, pages 153–167. Springer Berlin / Heidelberg, 2005.

[135] J. Greensmith, U. Aickelin, and G. Tedesco. Information fusion for anomaly detection with the dendritic cell algorithm. Information Fusion, in print. Retrieved January 26, 2008, from `http://eprints.nottingham.ac.uk/570/`, 2007.

[136] J. Greensmith, U. Aickelin, and J. Twycross. Detecting danger: Applying a novel immunological concept to intrusion detection systems. In *6th International Conference in Adaptive Computing in Design and Manufacture (ACDM '04)*, Bristol, UK, 2004.

[137] J. Greensmith, J. Twycross, and U. Aickelin. Dendritic cells for anomaly detection. In G. G. Y. et al., editor, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pages 664–671, Vancouver, Canada, 16-21 July 2006. IEEE Press.

[138] C. Grosan, A. Abraham, and S. Y. Han. Mepids: Multi-expression programming for intrusion detection system. In J. Mira and J. Alvarez, editors, *International Work-conference on the Interplay between Natural and Artificial Computation (IWINAC'05)*, volume 3562 of *Lecture Notes in Computer Science*, pages 163–172. Springer Verlag, Germany, Spain, 2005.

[139] C. R. Haag, G. B. Lamont, P. D. Williams, and G. L. Peterson. An artificial immune system-inspired multiobjective evolutionary algorithm with application to the detection of distributed computer network intrusions. In D. Thierens, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '07)*, pages 2717–2724, London, England, UK, 7-11July 2007. ACM.

[140] S. J. Han and S. B. Cho. Evolutionary neural networks for anomaly detection based on the behavior of a program. *IEEE Transactions On Systems, Man, And Cybernetics, part B*, 36(3):559–570, 2006.

[141] J. Handl, J. Knowles, and M. Dorigo. Strategies for the increased robustness of ant-based clustering. In G. D. M. Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, editors, *Engineering Self-Organising Systems*, volume 2977 of *Lecture Notes in Computer Science*, pages 90–104. Springer Berlin / Heidelberg, 2004.

[142] X. Hang and H. Dai. Constructing detectors in schema complementary space for anomaly detection. In K. D. et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '04), Part I, Seattle, WA, USA, 26-30 June, 2004*, volume 3102 of *Lecture Notes in Computer Science*, pages 275–286. Springer Berlin / Heidelberg, 2004.

[143] X. Hang and H. Dai. An extended negative selection algorithm for anomaly detection. In H. Dai, R. Srikant, C. Zhang, and N. Cercone, editors, *Advances in Knowledge Discovery and Data Mining*, volume 3056 of *Lecture Notes in Computer Science*, pages 245–254. Springer Berlin / Heidelberg, 2004.

[144] X. Hang and H. Dai. Applying both positive and negative selection to supervised learning for anomaly detection. In H.-G. Beyer and U.-M. O'Reilly, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECOO '05)*, pages 345–352, Washington, D.C., USA, 25-29 June 2005. ACM.

[145] J. V. Hansen, P. B. Lowry, R. D. Meservy, and D. M. McDonald. Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection. *Decision Support System*, 43(4):1362–1374, 2007.

[146] P. K. Harmer. A distributed agent architecture of a computer virus immune system. Master's thesis, Air Force Institute of Technology, Air University, March 2000.

[147] P. K. Harmer, P. D. Williams, G. H. Gunsch, and G. B. Lamont. An artificial immune system architecture for computer security applications. *IEEE Transactions on Evolutionary Computation*, 6(3):252–280, 2002.

[148] H. He, X. Luo, and B. Liu. Detecting anomalous network traffic with combined fuzzy-based approaches. In D.-S. Huang, X.-P. Zhang, and G.-B. Huang, editors, *Advances in Intelligent Computing*, volume 3645 of *Lecture Notes in Computer Science*, pages 433–442. Springer Berlin / Heidelberg, 2005.

[149] J. He, D. Long, and C. Chen. An improved ant-based classifier for intrusion detection. In *The 3nd International Conference on Natural Computation (ICNC '07)*, volume 4, pages 819–823. IEEE Press, 24-27 Aug. 2007.

[150] A. Hofmann, C. Schmitz, and B. Sick. Intrusion detection in computer networks with neural and fuzzy classifiers. In O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, editors, *Artificial Neural Networks and Neural Information Processing (ICANN/ICONIP '03)*, volume 2714 of *Lecture Notes in Computer Science*, pages 316–324. Springer Berlin / Heidelberg, 2003.

[151] A. Hofmann, C. Schmitz, and B. Sick. Rule extraction from neural networks for intrusion detection in computer networks. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1259–1265. IEEE Press, 5-8 Oct. 2003.

[152] S. Hofmeyr and S. Forrest. Immunity by design: An artificial immune system. In W.Banzhaf, J.Daida, A.E.Eiben, M.H.Garzon, V.Honavar, M.Jakiela, and R.E.Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 99)*, pages 1289–1296, Orlando, FL, USA, 13-17 July 1999. Morgan Kaufmann.

[153] S. A. Hofmeyr. *An Immunological Model of Distributed Detection and Its Application to Computer Security.* PhD thesis, The University of New Mexico, 1999.

[154] A. J. Hoglund, K. Hatonen, and A. S. Sorvari. A computer host-based user anomaly detction system using the self-organizing map. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '00)*, volume 5, pages 411–416, Como, Italy, 24-27 July 2000. IEEE Press.

[155] J. Holland and J. Reitman. Cognitive systems based on adaptive algorithms. In D. Waterman and F. Hayes-Roth, editors, *Pattern-Directed Inference Systems.* Academic Press, New York, 1978.

[156] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michican Press, Cambridge, MA, USA, 1975. ISBN-10: 0262581116.

[157] J. Horn and D. E. Goldberg. Natural niching for evolving cooperative classifiers. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Proceedings of the 1st Annual Conference on Genetic Programming*, pages 553–564, Cambrige, MA, USA, 1996. The Mit Press.

[158] N. Jerne. Towards a network theory of the immune system. *Ann Immunol (Paris)*, 125(1-2):373–389, 1974.

[159] Z. Ji. A boundary-aware negative selection algorithm. In A. del Pobil, editor, *In Proceedings of the 9th IASTED International Conference on Artificial Intelligence and Soft Computing*, pages 481–146, Benidorm, Spain, 12-14 September 2005. ACTA Press.

[160] Z. Ji. *Negative Selection Algorithms: from the Thymus to V-detector.* PhD thesis, Computer Science, The University of Memphis, August 2006.

[161] Z. Ji and D. Dasgupta. Artificial immune system (AIS) research in the last five years. In T. Gedeon, editor, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '03)*, volume 1, pages 123– 130, Canberra, Australia, 8-12 December 2003. IEEE Press.

[162] Z. Ji and D. Dasgupta. Augmented negative selection algorithm with variable-coverage detectors. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '04)*, volume 1, pages 1081–1088, Portland, OR, USA, 19-23 June 2004. IEEE Press.

[163] Z. Ji and D. Dasgupta. Real-valued negative selection using variable-sized detectors. In K. D. et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '04), Part I, Seattle, WA, USA, 26-30 June, 2004*, volume 3102 of *Lecture Notes in Computer Science*, pages 287–298. Springer Berlin / Heidelberg, 2004.

[164] Z. Ji and D. Dasgupta. Estimating the detector coverage in a negative selection algorithm. In H.-G. Beyer and U.-M. O'Reilly, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pages 281–288, Washington, D.C., USA, 25-29 June 2005. ACM.

[165] Z. Ji and D. Dasgupta. Applicability issues of the real-valued negative selection algorithms. In M. Cattolico, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '06)*, pages 111–118, Seattle, WA, USA, 8-12 July 2006. ACM.

[166] Z. Ji and D. Dasgupta. Revisiting negative selection algorithm. *Evolutionary Computation Journal*, 15(2):223–251, 2007.

[167] G. Jian, L. Da-xin, and C. in ge. An induction learning approach for building intrusion detection models using genetic algorithms. In *The 5th World Congress on Intelligent Control and Automation (WCICA 2004)*, volume 5, pages 4339–4342, Hangzhou, China, 5-19 June 2004. IEEE Press.

[168] J. Jiang, C. Zhang, and M. Kame. RBF-based real-time hierarchical intrusion detection systems. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN '03)*, volume 2, pages 1512–1516, Portland, OR, USA, 20-24 July 2003. IEEE Press.

[169] C. Jirapummin, N. Wattanapongsakorn, and P. Kanthamanon. Hybrid neural networks for intrusion detection system. In *The 2002 International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC '02)*, volume 7, pages 928–931, Phuket, Thailand, 2002.

[170] H. G. Kayacik. Hierarchical self organizing map based ids on kdd benchmark. Master's thesis, Dalhousie University, 2003.

[171] H. G. Kayacik, A. N. Zincir-Heywood, and M. Heywood. Evolving successful stack overflow attacks for vulnerability testing. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC '05)*, pages 8–15. IEEE Press, 5-9 Dec. 2005.

[172] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood. On the capability of an SOM based intrusion detection system. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN '03)*, volume 3, pages 1808–1813, Portland, OR, USA, 20-24 July 2003. IEEE Press.

[173] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood. A hierarchical SOM-based intrusion detection system. *Engineering Applications of Artificial Intelligence*, 20(4):439–451, 2007.

[174] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE Press, Nov/Dec 1995.

[175] J. Kim. *Integrating Artificial Immune Algorithms for Intrusion Detection*. PhD thesis, Department of Computer Science, University College London, 2003.

[176] J. Kim and P. Bentley. Negative selection and niching by an artificial immune system for network intrusion detection. In W.Banzhaf, J.Daida, A.E.Eiben, M.H.Garzon, V.Honavar, M.Jakiela, and R.E.Smith, editors, *Late Breaking Papers in the Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 99)*, pages 149–158, Orlando, FL, USA, 13-17 July 1999. Morgan Kaufmann.

[177] J. Kim and P. Bentley. Towards an artificial immune system for network intrusion detection: An investigation of dynamic clonal selection. In D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '02)*, volume 2, pages 1015–1020, Honolulu, HI, USA, 12-17 May 2002. IEEE Press.

[178] J. Kim, P. Bentley, U. Aickelin, J. Greensmith, G. Tedesco, and J. Twycross. Immune system approaches to intrusion detection-a review. *Natural Computing: an international journal*, 6(4):413–466, 2007.

[179] J. Kim, P. Bentley, C. Wallenta, M. Ahmed, and S. Hailes. Danger is ubiquitous: Detecting malicious activities in sensor networks using the dendritic cell algorithm. In H. Bersini and J. Carneiro, editors, *Artificial Immune Systems*, volume 4163 of *Lecture Notes in Computer Science*, pages 390–403. Springer Berlin / Heidelberg, 2006.

[180] J. Kim and P. J. Bentley. Towards an artificial immune system for network intrusion detection: An investigation of clonal selection with a negative selection operator. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '01)*, volume 2, pages 1244–1252, Seoul, South Korea, 27-30 May 2001. IEEE Press.

[181] J. Kim and P. J. Bentley. Immune memory in the dynamic clonal selection algorithm. In J. Timmis and P. J. Bentley, editors, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS '02)*, pages 57–65, Canterbury, UK, 9-11 September 2002. Unversity of Kent at Canterbury Printing Unit.

[182] J. Kim and P. J. Bentley. A model of gene library evolution in the dynamic clonal selection algorithm. In J. Timmis and P. J. Bentley, editors, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS '02)*, pages 175–182, Canterbury, UK, 9-11 September 2002. Unversity of Kent at Canterbury Printing Unit.

[183] J. Kim and P. J. Bentley. Immune memory and gene library evolution in the dynamical clonal selection algorithm. *Journal of Genetic Programming and Evolvable Machines*, 5(4):361–391, 2004.

[184] J. Kim, J. Greensmith, J. Twycross, and U. Aickelin. Malicious code execution detection and response immune system inpired by the danger theory. In *Adaptive and Resilient Computing Security Workshop (ARCS 2005)*, Santa Fe, NM, USA, 2005.

[185] J. Kim, W. Wilson, U. Aickelin, and J. McLeod. Cooperative automated worm response and detection immune algorithm (CARDINAL) inspired by t-cell immunity and tolerance. In C. Jacob, M. L. Pilat, P. J. Bentley, and J. Timmis, editors, *Proceedings of the 4th International Conference on Artificial Immune Systems (ICARIS '05), Banff, Alberta, CA, 14-17 August, 2005*, volume 3627 of *Lecture Notes in Computer Science*, pages 168–181. Springer Berlin / Heidelberg, 2005.

[186] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, 3rd edition, 2001.

[187] B. Kosko. Fuzzy cognitive maps. *International Journal of Man-Machine Studies*, 24(1):65–75, 1986.

[188] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992. ISBN-10: 0262111705.

[189] C. Kuok, A. Fu, and M. Wong. Mining fuzzy association rules in databases. *The ACM SIGMOD Record*, 27(1):41–46, 1998.

[190] K. Labib and R. Vemuri. NSOM: A real-time network-based intrusion detection system using self-organizing maps. Technical report, Dept. of Applied Science, University of California, Davis, 2002.

[191] P. LaRoche and A. N. Zincir-Heywood. 802.11 network intrusion detection using genetic programming. In F. Rothlauf, editor, *Workshop Proceedings of the Genetic and Evolutionary Computation Conference*, pages 170 – 171, Washington, D.C, USA, 25-26 June 2005. ACM.

[192] P. LaRoche and A. N. Zincir-Heywood. Genetic programming based WiFi data link layer attack detection. In *Proceedings of the 4th Annual Communication Networks and Services Research Conference (CNSR 2006)*, pages 8–15. IEEE Press, 24-25 May 2006.

[193] K.-C. Lee and L. Mikhailov. Intelligent intrusion detection system. In *Proceedings of the 2nd IEEE International Conference on Intelligence Systems*, volume 2, pages 497–502. IEEE Press, 22-24 June 2004.

[194] S. C. Lee and D. V. Heinbuch. Training a neural-network based intrusion detector to recognize novel attacks. *IEEE Transactions on Systems, Man and Cybernetics, Part A,*, 31(4):294–299, 2001.

[195] E. Leon, O. Nasraoui, and J. Gomez. Anomaly detection based on unsupervised niche clustering with application to network intrusion detection. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '04)*, volume 1, pages 502–508, Portland, OR, USA, 19-23 June 2004. IEEE Press.

[196] K. S. Leung, Y. Leung, L. So, and K. F. Yam. Rule learning in expert systems using genetic algorithms: 1, concepts. In *Proceeding of the 2nd International Conference on Fuzzy Logic and Neural Networks*, volume 1, pages 201–204, 1992.

[197] W. Li. A genetic algorithm approach to network intrusion detection. Technical report, SANS Institute 2004, 2004.

[198] W. Li. Using genetic algorithm for network intrusion detection. In *Proceedings of United States Department of Energy Cyber Security Group 2004 Training Conference*, Kansas City, KS, USA, 24-27 May 2004.

[199] Y. Liao, V. R. Vemuri, and A. Pasos. Adaptive anomaly detection with evolving connectionist systems. *Journal of Network and Computer Applications*, 30(1):60–80, 2007. Special Issue on Network and Information Security: A Computational Intelligence Approach.

[200] P. Lichodzijewski and M. I. Heywood. Pareto-coevolutionary genetic programming for problem decomposition in multi-class classification. In H. Lipson, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '07)*, pages 464–471, London, England, UK, 7-11July 2007. ACM.

[201] P. Lichodzijewski, A. Zincir-Heywood, and M. I. Heywood. Dynamic intrusion detection using self-organizing maps. In *In The*

14th Annual Canadian Information Technology Security Symposium, Ottawa, Canada, May 2002.

[202] P. Lichodzijewski, A. Zincir-Heywood, and M. I. Heywood. Host-based intrusion detection using self-organizing maps. In *The IEEE World Congress on Computational Intelligence, International Joint Conference on Neural Networks (IJCNN '02)*, volume 2, pages 1714–1719, Honolulu, HI, USA, 12-17 May 2002. IEEE Press.

[203] F. Liu and L. Lin. Unsupervised anomaly detection based on an evolutionary artificial immune network. In F. R. et al., editor, *Applications on Evolutionary Computing-EvoWorkkshops 2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC, Lausanne, Switzerland, March 30 - April 1, 2005*, volume 3449 of *Lecture Notes in Computer Science*, pages 166–174. Springer Berlin / Heidelberg, 2005.

[204] F. Liu and L. Luo. Immune clonal selection wavelet network based intrusion detection. In W. Duch, J. Kacprzyk, E. Oja, and S. Zadrożny, editors, *Artificial Neural Networks: Biological Inspirations-ICANN 2005*, volume 3696 of *Lecture Notes in Computer Science*, pages 331–336. Springer Berlin / Heidelberg, 2005.

[205] F. Liu, B. Qu, and R. Chen. Intrusion detection based on immune clonal selection algorithms. In G. I.Webb and XinghuoYu, editors, *AI 2004: Advances in Artificial Intelligence*, volume 3339 of *Lecture Notes in Computer Science*, pages 1226–1232. Springer Berlin / Heidelberg, 2004.

[206] Z. Liu, G. Florez, and S. M. Bridges. A comparison of input representations in neural networks: A case study in intrusion detection. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN '02)*, volume 2, pages 1708–1713, Honolulu, HI, USA, 12-17 May 2002. IEEE Press.

[207] W. Lu. *An unsupervised anomaly detection framework for multiple-connection based network intrusions.* PhD thesis, Department of Electrical and Computer Engineering, University of Victoria, 2005.

[208] W. Lu and I. Traore. Detecting new forms of network intrusion using genetic programming. *Computational Intelligence*, 20(3):475–494, 2004. Blackwell Publishing, Boston MA & Oxford UK.

[209] W. Lu and I. Traore. An unsupervised anomaly detection framework for network intrusions. Technical report, Information Security and Object Technology (ISOT) Group, University of Victoria, October 2005.

[210] J. Luo and S. M. Bridges. Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection. *International Journal of Intelligent Systems*, 15(8):687–703, 2001.

[211] J. Luo, S. M. Bridges, and R. B. Vaughn. Fuzzy frequent episodes for real-time intrusion detection. In *The 10th IEEE International Conference on Fuzzy Systems (FUZZ '01)*, volume 1, pages 368–371, Melbourne, Vic., Australia, 2001. IEEE Press.

[212] W. Luo, X. Wang, and X. Wang. A novel fast negative selection algorithm enhanced by state graphs. In L. N. de Castro and F. J. Z. H. Knidel, editors, *Artificial Immune Systems*, volume 4628 of *Lecture Notes in Computer Science*, pages 168–181. Springer Berlin / Heidelberg, 2007.

[213] K. Luther, R. Bye, T. Alpcan, A. Muller, and S. Albayrak. A cooperative ais framework for intrusion detection. In *IEEE International Conference on Communications (ICC '07)*, pages 1409–1416, Glasgow, Scotland, 4-28 June 2007.

[214] S. W. Mahfoud. Crossover interactions among niches. In *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, volume 1, pages 188–193, Orlando, FL, USA, June 1994.

[215] M. V. Mahoney and P. K. Chan. An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection. Technical Report TR CS-2003-02, Computer Science Department, Florida Institute of Technology, 2003.

[216] H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 146–151, Portland, OR, USA, August 1996. AAAI Press.

[217] P. Matzinger. Tolerance, danger and the extended family. *Annual Review in Immunology*, 12:991–1045, 1994.

[218] P. Matzinger. The danger model in its historical context. *Scandinavian Journal of Immunology*, 54(1-2):4–9, 2001.

[219] J. McHugh. Testing intrusion detection systems a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security*, 3(4):262–294, 2000.

[220] L. Mé. GASSATA, a genetic algorithm as an alternative tool for security audit trails analysis. In *Proceedings of the 1st International Workshop on the Recent Advances in Intrusion Detection (RAID 98)*, Louvain-la-Neuve, Belgium, 14-16 September 1998.

[221] M. Mischiatti and F. Neri. Applying local search and genetic evolution in concept learning systems to detect intrusion in computer networks. In R. L. de Mántaras and E. Plaza, editors, *Proceedings of the 11th European Conference on Machine Learning (ECML '00), Barcelona, Spain, 31 May - 2 June, 2000*, volume 1810 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2000.

[222] A. Mitrokotsa and C. Douligeris. Detecting denial of service attacks using emergent self-organizing maps. In *Proceedings of the 5th IEEE International Symposium on Signal Processing and Information Technology*, pages 375–380. IEEE Press, 18-21 Dec. 2005.

[223] A. Mitrokotsa and C. Douligeris. Intrusion detection using emergent self-organizing maps advances in artificial intelligence. In G. Antoniou, G. Potamias, C. Spyropoulos, and D. Plexousakis, editors, *Advances in Artificial Intelligence*, volume 3955 of *Lecture Notes in Computer Science*, pages 559–562. Springer Berlin / Heidelberg, SETN, 2006.

[224] A. Mitrokotsa, N. Komninos, and C. Douligeris. Towards an effective intrusion response engine combined with intrusion detection in ad hoc networks. In *The Sixth Annual Mediterranean Ad Hoc Networking WorkShop*, Corfu, Greece, June 12-15 2007.

[225] M. Mohajerani, A. Moeini, and M. Kianie. NFIDS: a neuro-fuzzy intrusion detection system. In *Proceedings of the 2003 10th IEEE International Conference on Electronics, Circuits and Systems (ICECS '03)*, volume 1, pages 348–351, 14-17 Dec. 2003.

[226] M. Moradi and M. Zulkernine. A neural network based system for intrusion detection and classification of attacks. In *Proceedings of the 2004 IEEE International Conference on Advances in Intelligent Systems-Theory and Applications*, Luxembourg-Kirchberg, Luxembourg, 15-18 November 2004. IEEE Press.

[227] S. Mukkamala and A. H. Sung. A comparative study of techniques for intrusion detection. In *Proceedings of 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 570– 577. IEEE Press, 3-5 Nov. 2003.

[228] S. Mukkamala, A. H. Sung, and A. Abraham. Modeling intrusion detection systems using linear genetic programming approach. In R. Orchard, C. Yang, and M. Ali, editors, *The 17th International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems, Innovations in Applied Artificial Intelligence,*, volume 3029 of *Lecture Notes in Computer Science*, pages 633–642. Springer Verlag, Germany, 2004.

[229] S. Mukkamala, A. H. Sung, and A. Abraham. Intrusion detection using an ensemble of intelligent paradigms. *Journal of Network and Computer Applications*, 28(2):167–182, 2005.

[230] F. Neri. Mining TCP/IP traffic for network intrusion detection by using a distributed genetic algorithm. In R. L. de Mántaras and E. Plaza, editors, *Proceedings of the 11th European Conference on Machine Learning (ECML '00), Barcelona, Spain, 31 May - 2 June, 2000*, volume 1810 of *Lecture Notes in Computer Science*, pages 313–322. Springer Berlin / Heidelberg, 2000.

[231] S. Olariu and A. Y. Zomaya, editors. *Handbook Of Bioinspired Algorithms And Applications.* Chapman & Hall/CRC, 2006. ISBN-10:1584884754.

[232] M. Oltean. Multi expression programming. Technical report, Department of Computer Science, Babes-Bolyai University, June 4 2006.

[233] M. Ostaszewski, F. Seredynski, and P. Bouvry. Immune anomaly detection enhanced with evolutionary paradigms. In M. Cattolico, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '06)*, pages 119–126, Seattle, WA, USA, 8-12 July 2006. ACM.

[234] M. Ostaszewski, F. Seredynski, and P. Bouvry. A nonself space approach to network anomaly detection. In *20th International Parallel and Distributed Processing Symposium (IPDPS '06)*, pages 8–16. IEEE Press, 25-29 April 2006.

[235] M. Ostaszewski, F. Seredynski, and P. Bouvry. Coevolutionary-based mechanisms for network anomaly detection. *Journal of Mathematical Modelling and Algorithms*, 6(3):411–431, 2007.

[236] T. Özyer, R. Alhajj, and K. Barker. Intrusion detection by integrating boosting genetic fuzzy classifier and data mining criteria for rule pre-screening. *Journal of Network and Computer Applications*, 30(1):99–113, 2007.

[237] R. Parpinelli, H. Lopes, and A. Freitas. Data mining with an ant colony optimization algorithm. *Evolutionary Computation, IEEE Transactions on*, 6(4):321–332, 2002.

[238] S. Peddabachigari, A. Abraham, C. Grosan, and J. Thomas. Modeling intrusion detection system using hybrid intelligent systems. *Journal of Network and Computer Applications*, 30(1):114–132, 2007.

[239] A. Perelson, R. Hightower, and S. Forrest. Evolution and somatic learning in V-region genes. *Research in Immunology*, 147(4):202–208, 1996.

[240] M. M. Pillai, J. H. Eloff, and H. S. Venter. An approach to implement a network intrusion detection system using genetic algorithms. In *Proceedings of the 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*, volume 75 of *ACM International Conference Proceeding Series*, pages 221–221, Stellenbosch, Western Cape, South Africa, 2004. South African In-

stitute for Computer Scientists and Information Technologist.

[241] D. Poole, A. Mackworth, and R. Goebel. *Computational Intelligence - A Logical Approach.* Oxford University Press, Oxford, UK, 1998. ISBN-10: 195102703.

[242] V. Ramos and A. Abraham. ANTIDS: Self-organized ant-based clustering model for intrusion detection system. In *The 4th IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST'05)*, Japan, 2005. IEEE Press.

[243] A. Rapaka, A. Novokhodko, and D. Wunsch. Intrusion detection using radial basis function network on sequence of system calls. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN '03)*, volume 3, pages 1820–1825, Portland, OR, USA, 20-24 July 2003. IEEE Press.

[244] B. C. Rhodes, J. A. Mahaffey, and J. D. Cannady. Multiple self-organizing maps for intrusion detection. In *Proceedings of the 23rd National Information Systems Security Conference*, pages 16–19, Baltimore, MA, USA, 16-19 October 2000.

[245] J. Ryan, M. J. Lin, and R. Miikkulainen. Intrusion detection with neural networks. *Advances in Neural Information Processing Systems*, 10:943–949, 1998.

[246] M. Sabhnani and G. Serpen. Why machine learning algorithms fail in misuse detection on KDD intrusion detection data set. *Intelligent Data Analysis*, 8(4):403–415, 2004.

[247] S. T. Sarasamma, Q. A. Zhu, and J. Huff. Hierarchical kohonenen net for anomaly detection in network security. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 35(2):302–312, 2005.

[248] H. Seo, T. Kim, and H. Kim. Modeling of distributed intrusion detection using fuzzy system. In D.-S. Huang, K. Li, and G. W. Irwin, editors, *Computational Intelligence*, volume 4114 of *Lecture Notes in Computer Science*, pages 165–170. Springer Berlin / Heidelberg, 2006.

[249] K. Shafi, H. Abbass, and W. Zhu. Real time signature extraction during adaptive rule discovery using ucs. In D. Srinivasan and L. Wang, editors, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pages 2509–2516, Singapore, 25-28 September 2007. IEEE Press.

[250] K. Shafi, H.A.Abbass, and W. Zhu. An adaptive rule-based intrusion detection architecture. In *The Security Technology Conference, the 5th Homeland Security Summit*, pages 345–355, Canberra, Australia, 19-21 September 2006.

[251] K. Shafi, Kamran, H. A. Abbass, and W. Zhu. The role of early stopping and population size in xcs for intrusion detection. In T.-D. Wang, X. Li, S.-H. Chen, X. Wang, H. Abbass, H. Iba, G. Chen, and XinYao, editors, *Simulated Evolution and Learning*, volume 4247 of *Lecture Notes in Computer Science*, pages 50–57. Springer Berlin / Heidelberg, 2006.

[252] K. Shafi, T. Kovacs, H. A. Abbass, and W. Zhu. Intrusion detection with evolutionary learning classifier systems. *Natural Computing*, 2007. In print. Springer Netherlands. Retrieved January 26, 2008, from `http://www.springerlink.com/content/u71x164277408137/`.

[253] H. Shah, J. Undercoffer, and A. Joshi. Fuzzy clustering for intrusion detection. In *The 12th IEEE International Conference on Fuzzy Systems (FUZZ '03)*, volume 2, pages 1274–1278, St.Louis, MO, USA, 25-28 May 2003. IEEE Press.

[254] J. M. Shapiro, G. B. Lamont, and G. L. Peterson. An evolutionary algorithm to generate hyper-ellipsoid detectors for negative selection. In H.-G. Beyer and U.-M. O'Reilly, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pages 337–344, Washington, D.C., USA, 25-29 June 2005. ACM.

[255] C. Sinclair, L. Pierce, and S. Matzner. An application of machine learning to network intrusion detection. In *Proceedings of 15th Annual Computer Security Applications Conference (ACSAC '99)*, pages 371–377, Phoenix, AZ, USA, 6-10 December 1999. IEEE Computer Society.

[256] A. Siraj, S. M. Bridges, and R. B. Vaughn. Fuzzy cognitive maps for decision support in an intelligent intrusion detection system. In *Proceedings of the 20th International Conference of the North American Fuzzy Information Society (NAFIPS '01) and Joint the 9th IFSA World Congress*, volume 4, pages 2165–2170, Vancouver, Canada, 25-28 July 2001. IEEE Press.

[257] A. Siraj, R. B. Vaughn, and S. M. Bridges. Intrusion sensor data fusion in an intelligent intrusion detection system architecture. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS '04)*, volume 9, pages 10–20. IEEE Press, 5-8 Jan. 2004.

[258] A. Somayaji, S. A. Hofmeyr, and S. Forrest. Principles of a computer immune system. In *Proceedings of the 1997 workshop on New security paradigms*, pages 75 – 82, Langdale, Cumbria, UK, 1997. ACM.

[259] D. Song. A linear genetic programming approach to intrusion detection. Master's thesis, Dalhousie University, March 2003.

[260] D. Song, M. I. Heywood, and A. N. Zincir-Heywood. A linear genetic programming approach to intrusion detection. In E. C.-P. et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '03), Part II, Chicago, IL, USA, 12-16 July, 2003*, volume 2724 of *Lecture Notes in Computer Science*, pages 2325–2336. Springer Berlin / Heidelberg, 2003.

[261] D. Song, M. I. Heywood, and A. N. Zincir-Heywood. Training genetic programming on half a million patterns: An example from anomaly detection. *IEEE Transactions on Evolutionary Computation*, 9(3):225–239, 2005.

[262] T. Stibor, P. Mohr, and J. Timmis. Is negative selection appropriate for anomaly detection? In H.-G. Beyer and U.-M. O'Reilly, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pages 321–328, Washington, D.C., USA, 25-29 June 2005. ACM.

[263] T. Stibor, J. Timmis, and C. Eckert. A comparative study of real-valued negative selection to statistical anomaly detection techniques. In C. Jacob, M. L. Pilat, P. J. Bentley, and J. Timmis, editors, *Artificial Immune Systems*, volume 3627 of *Lecture Notes in Computer Science*, pages 262–275. Springer Berlin / Heidelberg, 2005.

[264] K. Tan. The application of neural networks to unix computer security. In *Proceedings of IEEE International Conference on Neural Networks*, volume 1, pages 476–481, Perth, WA, Australia, Nov/Dec 1995. IEEE Press.

[265] G. Tedesco, J. Twycross, and U. Aickelin. Integrating innate and adaptive immunity for intrusion detection. In H. Bersini and J. Carneiro, editors, *Proceedings of the 5th International Conference on Artificial Immune Systems (ICARIS '06), Oeiras, Portugal, 4-6 September, 2006*, volume 4163 of *Lecture Notes in Computer Science*, pages 193–202. Springer Berlin / Heidelberg, 2006.

[266] J. Tian, Y. Fu, Y. Xu, and J. ling Wang. Intrusion detection combining multiple decision trees by fuzzy logic. In *Proceedings of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT '05)*, pages 256–258. IEEE Press, 05-08 Dec. 2005.

[267] J. Timmis. Artificial immune systems-today and tomorrow. *Natural Computing*, 6(1):1–18, 2007.

[268] A. N. Toosi and M. Kahani. A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers. *Computer Communications*, 30(10):2201–2212, 2007.

[269] C.-H. Tsang and S. Kwong. Multi-agent intrusion detection system in industrial network using ant colony clustering approach and unsupervised feature extraction. In *IEEE International Conference on Industrial Technology (ICIT '05)*, pages 51– 56. IEEE Press, 14-17 Dec. 2005.

[270] C.-H. Tsang and S. Kwong. Ant colony clustering and feature extraction for anomaly intrusion detection. In A. Abraham, C. Grosan, and V. Ramos, editors, *Swarm Intelligence in Data Mining*, volume 34 of *Studies in Computational Intelligence*, pages 101–123. Springer Berlin / Heidelberg, 2006.

[271] C.-H. Tsang, S. Kwong, and H. Wang. Anomaly intrusion detection using multi-objective genetic fuzzy system and agent-based evolutionary computation framework. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM '05)*, pages 4–7. IEEE Press, 27-30 Nov. 2005.

[272] C.-H. Tsang, S. Kwong, and H. Wang. Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. *Pattern Recognition*, 40(9):2373–2391, 2007.

[273] J. Twycross and U. Aickelin. libtissue-implementing innate immunity. In G. G. Y. et al., editor, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pages 499–506, Vancouver, Canada, 16-21 July 2006. IEEE Press.

[274] J. Twycross and U. Aickelin. Detecting anomalous process behaviour using second generation artificial immune systems. Retrieved January 26, 2008, from `http://www.cpib.ac.uk/~jpt/papers/raid-2007.pdf`, 2007.

[275] J. Twycross and U. Aickelin. An immune-inspired approach to anomaly detection. In *Handbook of Research on Information Assurance and Security.* Idea Publishing Group, 2007.

[276] J. P. Twycross. *Integrated Innate and Adaptive Artificial Immune Systems applied to Process Anomaly Detection.* PhD thesis, the University of Nottingham, January 2007.

[277] W. Wang, X. Guan, X. Zhang, and L. Yang. Profiling program behavior for anomaly intrusion detection based on the transition and frequency property of computer audit data. *Computers & Security*, 25(7):539–550, 2006.

[278] A. Watkins, J. Timmis, and L. Boggess. Artificial immune recognition system (airs): An immune-inspired supervised learning algorithm. *Genetic Programming and Evolvable Machines*, 5(3):291–317, 2004.

[279] S. Wierzchon. Generating optimal repertoire of antibody strings in an artificial immune system. In *Proceedings of the IIS'2000 Symposium on Intelligent Information Systems*, pages 119–133. Physica-Verlag, 2000.

[280] P. D. Williams, K. P. Anchor, J. L. Bebo, G. H. Gunsch, and G. D. Lamont. CDIS: Towards a computer immune system for detecting network intrusions. In W. Lee, L. Mé, and A. Wespi, editors, *Pro-*

ceedings of the 4th International Workshop on Recent Advances in Intrusion Detection (RAID '01), Davis, CA, USA, 10-12 October, volume 2212 of *Lecture Notes in Computer Science*, pages 117–133. Springer Berlin / Heidelberg, 2001.

[281] D. Wilson and D. Kaur. Using grammatical evolution for evolving intrusion detection rules. In *Proceedings of the 5th WSEAS Int. Conf. on Circuits, Systems, Electronics, Control & Signal Processing*, pages 42–47, Dallas, TX, USA, 1-3 November 2006.

[282] T. Xia, G. Qu, S. Hariri, and M. Yousif. An efficient network intrusion detection method based on information theory and genetic algorithm. In *The 24th IEEE International Conference on Performance, Computing, and Communications (IPCCC 2005)*, pages 11–17, Phoenix, AZ, USA, 7-9 April 2005. IEEE Press.

[283] J. Xian, F. Lang, and X. Tang. A novel intrusion detection method based on clonal selection clustering algorithm. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3905– 3910, 18-21 Aug. 2005.

[284] J. Xin, J. E. Dickerson, and J. A. Dickerson. Fuzzy feature extraction and visualization for intrusion detection. In *The 12th IEEE International Conference on Fuzzy Systems (FUZZ '03)*, volume 2, pages 1249–1254, St.Louis, MO, USA, 25-28 May 2003. IEEE Press.

[285] Q. Xu, W. Pei, L. Yang, and Q. Zhao. An intrusion detection approach based on understandable neural network trees. *International Journal of Computer Science and Network Security*, 6(11):229–234, 2006.

[286] J. Yao, S. Zhao, and L. V. Saxton. A study on fuzzy intrusion detection. In *Proceedings of SPIE: Data Mining, Intrusion Detection, Information Assurance, And Data Networks Security*, volume 5812, pages 23–30, 2005.

[287] C. Yin, S. Tian, H. Huang, and J. He. Applying genetic programming to evolve learned rules for network anomaly detection. In L. Wang, K. Chen, and Y. S. Ong, editors, *Advances in Natural Computation*, volume 3612 of *Lecture Notes in Computer Science*, pages 323–331. Springer Berlin / Heidelberg, 2005.

[288] Y. Yu, F. Gao, and Y. Ge. Hybrid BP/CNN neural network for intrusion detection. In *Proceedings of the 3rd international conference on Information security*, volume 85 of *ACM International Conference Proceeding Series*, pages 226–228, 2004.

[289] L. Zadeh. Role of soft computing and fuzzy logic in the conception, design and development of information/intelligent systems. In O. Kaynak, L. Zadeh, B. Turksen, and I. Rudas, editors, *Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications; Proceedings of the NATO Advanced Study Institute on Soft Computing and Its Applications held at Manavgat, Antalya, Turkey, August 21-31, 1996*, volume 162 of *NATO ASI Series*, pages 1–9. Springer Berlin / Heidelberg, 1998.

[290] S. Zanero. Analyzing TCP traffic patterns using self organizing maps. In F. Roli and S. Vitulano, editors, *International Conference on Image Analysis and Processing (ICIAP '05), Cagliari, Italy, 6-8 September, 2005*, volume 3617 of *Lecture Notes in Computer Science*, pages 83–90. Springer Berlin / Heidelberg, 2005.

[291] S. Zanero. Improving self organizing map performance for network intrusion detection. In *International Workshop on Clustering High-Dimensional data and its applications, in Conjunction with the 5th SIAM International Conference on Data Mining (SDM '05)*, Newport Beach, CA, USA, April 2005.

[292] S. Zanero and S. M. Savaresi. Unsupervised learning techniques for an intrusion detection system. In *Proceedings of the ACM Symposium on Applied Computing (ACM SAC '04)*, Computer security, pages 412–419, Nicosia, Cyprus, 14-17 Mar 2004. ACM.

[293] J. Zeng, T. Li, X. Liu, C. Liu, L. Peng, and F. Sun. A feedback negative selection algorithm to anomaly detection. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 3, pages 604–608. IEEE Press, 24-27 Aug 2007.

[294] B. Zhang. Internet intrusion detection by autoassociative neural network. In *Proceedings of International Symposium on Information & Communications Technologies*, Malaysia, Dec. 2005.

[295] C. Zhang, J. Jiang, and M. Kamel. Comparison of BPL and RBF network in intrusion detection system. In G. Wang, Q. Liu, Y. Yao, and A. Skowron, editors, *Proceedings of the 9th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing (RSFDGrC '03), 26-29 May, Chongqing, China*, volume 2639 of *Lecture Notes in Computer Science*, chapter Proceedings of the 9th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing (RSFDGrC '03), pages 466–470. Springer Berlin / Heidelberg, 2003.

[296] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, and J. Ucles. HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In *Proceedings of the 2001 IEEE Workshop Information Assurance and Security*, pages 85–90, West Point, NY, USA, 2001. IEEE Press.

[297] J. Zhao, J. Zhao, and J. Li. Intrusion detection based on clustering genetic algorithm. In *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, volume 6, pages 3911–3914, Guangzhou, China, 18-21 August 2005. IEEE Press.

[298] C. Zheng and L. Chen. FCBI-an efficient user-friendly classifier using fuzzy implication table. In L. Kalinichenko, R. Manthey, B. Thalheim, and U. Wloka, editors, *Advances in Databases and Information Systems*, volume 2798 of *Lecture Notes in Computer Science*, pages 266–277. Springer Berlin / Heidelberg, 2003.