Practical Comparison between Genetic Algorithm and Clonal Selection Theory on KDD data set

Najlaa Badie Aldabagh
Assistant prof.
College of computer science and Mathematics/ computer science

Mafaz Muhsin Khalil
Assistant teacher
College of science/ Biology science

Abstract

This paper compares between two models: Common Genetic algorithm and the new Clonal selection theory in the field of Intrusion Detection. Genetic algorithms (GA) which is a model of genetic evolution, while Clonal selection theory (CST) is from models of the natural immune system NIS, the two models are from two different fields of Artificial Intelligence AI but they have portion of shared operations and objectives. The comparison to be done by applying the two models on some records of Knowledge Discovery and Data mining tools which is known by the name KDD data sets (its records the data of the interring packets to the computer system from the internet), to produce population (in case of GA) or antibodies (in case of CST) can recognize these abnormal records.

مقارنة عملية بين الخوارزمية الجينية ونظرية إنتقاء السلالة على مجموعة بيانات KDD

أ.د.م نجلاء بديع الدباغ
أستاذ مساعد
كلية علوم الحاسبات والرياضيات/
علوم حاسبات

مفاز محسن خليل
مدرس مساعد
كلية العلوم/علوم حياة

الملخص

يقارن هذا البحث مابين نموذجان: الخوارزمية الجينية المعروفة ونـظرية انتقـاء الـسلالة الجديدة في مجال كشف التطفل. حيث تعتبر الخوارزمية الجينية (GA) نموذج للتطور الجيني، بينما تعتبر نظرية انتقاء السلالة (CST) من نماذج نظام المناعة الطبيعي، أي أنهما من حقلين مختلفـين للذكاء الاصطناعي ولكن لديهما بعض العمليات والاهداف المشتركة. تمت المقارنة بتطبيق النموذجان على بعض سجلات مجموعات بيانات الـ KDD (التي تسجل بيانات الحزم الداخلـة إلـى نظـام الحاسوب من خلال الإنترنت)، لإنتاج جيل (في حالة الخوارزمية الجينية) أو مضادات حيـة Abs (في حالة نظرية انتقاء السلالة) يمكنهما تميز السجلات الغير طبيعية أو المهاجمة.

1. Introduction

Internet has given users a need for security components to protect themselves. Certain techniques are used to secure important data, such as firewall and encryption etc. . Firewall acts as a defense to protect sensitive data, but it merely reduces exposure rather than monitors or eliminates vulnerabilities in computer systems. Any encrypted message can be decrypted in theory, and encryption adds extra burden on hosts or application. Moreover, any new security techniques themselves might have design flaws. Obviously, it is important to have a detecting and monitoring system to protect important data. For this reason the detection methods of intruders in the computer networks have drawn attention to many researchers in recent years.[1]

An Intrusion Detection System (IDS) is an important component of the computer and information security framework. Its main goal is to differentiate between normal activities of the system and behaviors that can be classified as suspicious or intrusive.

There are two main approaches to design of IDSs: misuse and anomaly detection techniques. In a misuse detection based IDS, intrusions are detected by looking for activities that correspond to known signatures of intrusion and vulnerabilities. On the other hand, the anomaly detection based IDSs detect attacks by observing deviations from behavior of the system. Its works by comparing network traffic, system call sequences, or other features of known attack patterns.

Clonal selection algorithms, however, are very similar to a kind of evolutionary algorithm; namely, evolutionary strategies, although they have a different biological inspiration. Clonal selection algorithms are also population-based search and optimization algorithms generating a memory pool of suitable antibodies for solving a particular problem.

2. Input data (the KDD Cup 99 Data)

This is the data set of The Third International Knowledge Discovery and Data mining tools competition, which was held in conjunction with KDD cup 99 the Fifth International Conference on Knowledge Discovery and Data mining. The KDD cup 1999 is dataset used for benchmarking intrusion detection problems. The dataset was a collection over a period of nine weeks on local area network. The types are grouped into five categories (Normal, Probing, Denial of Service (DoS), User to Root (U2R), and Remote to Local (R2L)).

KDD Cup 99 dataset is divided into training and testing record sets. Total number of connection records in the training dataset is about 5 million records. This too large for our purpose, only concise training dataset of KDD Cup 99, known as 10% KDD Cup 99, and test dataset which called (correct) data set was employed here [1].

Each record contained values of 41 independent variables (fields) describing the different features of the connection, and the value of the dependent variable labeled as either normal, or as an attack, with exactly one specific attack type, the sample of four connection record corresponding to the attack types, and the list of 41 features corresponding to their types is showing below [KDD data set].

0,tcp,http,SF,181,5450,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0.00,0.00,0.00,0.00,1.00,0.00,0.00,9,9,1.00,0.00,0.11,0.00,0.00,0.00,0.00,0.00,normal.

0,tcp,telnet,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,5,0.83,1.00,0.00,0.00,0.83,0.33,0.00,5,6,1.00,0.00,0.20,0.33,1.00,0.83,0.00,0.00,neptune.

0,icmp,ecr_i,SF,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,316,316,0.00,0.00,0.00,0.00,1.00,0.00,0.00,148,3,0.02,0.02,0.02,0.00,0.00,0.00,0.00,0.00,smurf.

0,udp,private,SF,28,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,73,1,0.01,0.05,0.01,0.00,0.00,0.00,0.00,0.00,teardrop.

Four Samples of Connection Records Corresponding to the Attack Types.

3. Evolutionary Computation

Evolution is an optimization process where the aim is to improve the ability of an organism (or system) to survive in dynamically changing and competitive environments [2][3].

Evolutionary computation (EC) refers to computer-based problem solving systems that use computational models of evolutionary processes, such as natural selection, survival of the fittest and reproduction, as the fundamental components of such computational systems.

Evolution via natural selection of a randomly chosen population of individuals can be thought of as a search through the space of possible chromosome values. In that sense, an evolutionary algorithm (EA) is a stochastic search for an optimal solution to a given problem. The evolutionary search process is influenced by the following main components of an EA:

- an **encoding** of solutions to the problem as a chromosome;
- a **function** to evaluate the **fitness**, or survival strength of individuals;
- **initialization** of the initial population;
- **selection** operators; and
- **reproduction** operators.

Algorithm (1) shows how these components are combined to form a generic EA [2].

Algorithm 1. Generic Evolutionary Algorithm

Let $t = 0$ be the generation counter;
Create and initialize an n_x-dimensional population, $C(0)$, to consist of n_s individuals;
while *stopping condition(s) not true* **do**
    Evaluate the fitness, $f(\mathbf{x}_i(t))$, of each individual, $\mathbf{x}_i(t)$;
    Perform reproduction to create offspring;
    Select the new population, $C(t + 1)$;
    Advance to the new generation, i.e. $t = t + 1$;
end

The steps of an EA are applied iteratively until some stopping condition is satisfied. Each iteration of an EA is referred to as a generation. The different ways in which the EA components are implemented result in different EC paradigms:[2]

- **Genetic algorithms (GAs)**, which model genetic evolution.
- **Genetic programming (GP)**, which is based on genetic algorithms, but individuals are programs (represented as trees).
- **Evolutionary programming (EP)**, which is derived from the simulation of adaptive behavior in evolution (i.e. phenotypic evolution).
- **Evolution strategies (ESs)**, which are geared toward modeling the strategic parameters that control variation in evolution, i.e. the evolution of evolution.
- **Differential evolution (DE)**, which is similar to genetic algorithms, differing in the reproduction mechanism used.
- **Cultural evolution (CE)**, which models the evolution of culture of a population and how the culture influences the genetic and phenotypic evolution of individuals.
- **Co-evolution (CoE)**, where initially "dumb" individuals evolve through cooperation, or in competition with one another, acquiring the necessary characteristics to survive.

3.2 Genetic Algorithms

Genetic algorithms (GA) are possibly the first algorithmic models developed to simulate genetic systems. GAs model genetic evolution, where the characteristics of individuals are expressed using genotypes. The main driving operators of a GA is selection (to model survival of the fittest) and recombination through application of a crossover operator (to model reproduction). This section discusses in detail GA used in this research and their evolution operators, which is follows the general algorithm as given in Algorithm (1), but with different components are combined to form GA particularity to solve intrusion detection problem in KDD data set [2].

- A real value representation was used.
- Stochastic Universal sampling selection was used to select parents for recombination.

- Uniform crossover was used as the primary method to produce offspring.
- Somatic Mutation for real-value.
- Fitness evaluation, see section 8 .
- Positive Selection, see section 10.2.
- Replace worst.
- Stopping Condition, see section 9.
- Negative Selection, see section 10.1, this step performed on detectors one time after the generation cycles complete.

The two steps Positive Selection and Negative Selection are from AIS and we added them here because they are necessary for intrusion detection applications.

1. Real value representation

Since our data consist of fields have different types characters and numbers. To unite them we convert characters to numbers, and then applied normalization process on them to obtain values in range [0 – 1].

The benefit of data transformation such as normalization may improve the accuracy and efficiency of artificial algorithms. Such methods provide better results if data to be analyzed has been normalized, that is, scaled to specific range as [0 – 1]. [2]

Min-Max Normalization: The min-Max normalization performs a linear transformation on the original data values. Suppose that minX and maxX are the minimum and maximum of feature X. In order to map interval [*minX* - *maxX*] into new interval [new *minX* – new *maxX*]. Consequently, every value *v* from the original interval will be mapped into value new_v using the following formula [1]:

$$new_v = \frac{v - minX}{maxX - minX}$$

2. Proportional Selection (Stochastic Universal sampling)

Selection operators are characterized by their selective pressure, also referred to as the takeover time, which relates to the time it requires to produce a uniform population. It is defined as the speed at which the best solution will occupy the entire population by repeated application of the selection operator alone. An operator with a high selective pressure decreases diversity in the population more rapidly than operators with a low selective pressure, which may lead to premature convergence to suboptimal solutions. A high selective pressure limits the exploration abilities of the population [2].

Two popular sampling methods used in proportional selection is roulette wheel sampling and stochastic universal sampling.

In roulette wheel selection it may happen that the best individual is not selected to produce offspring during a given generation. To prevent this problem, stochastic universal sampling (refer to Algorithm 2), used to determine for each individual the number of offspring, λ_i, to be produced by the individual with only one call to the algorithm.

Because selection is directly proportional to fitness, it is possible that strong individuals may dominate in producing offspring, thereby limiting the diversity of the new population. In other words, proportional selection has a high selective pressure [2][4].

Algorithm 2. Stochastic Universal Sampling.

for i = 1, . . . , n_s **do** , where n_s is the population size
    $\lambda_i(t) = 0$; , no of offspring for each individual
end
r ~ U(0, 1/λ), where λ is the total number of offspring, r is random no. in range [0, 1/λ];
sum = 0.0;
for i = 1, . . . , ns **do**
    sum = sum + $\gamma_s(x_i(t))$; where $\gamma_s(x_i(t))$ is the probability that x_i will be selected
    while r < sum **do**
        λ_i ++;
        r = r + 1/λ;
    end
end
return λ = (λ_1, . . . , λ_{ns});

3. Crossover (Uniform crossover)

Crossover operators can be divided into three main categories based on the amity (i.e. the number of parents used) of the operator. This results in three main classes of crossover operators:

- **asexual**, where an offspring is generated from one parent.
- **sexual**, where two parents are used to produce one or two offspring.
- **multi-recombination**, where more than two parents are used to produce one or more offspring.

Crossover operators are further categorized based on the representation scheme used. For example, binary-specific operators have been developed for binary string representations, and operators specific to floating-point representations.

Parents are selected using the selection scheme discussed in previous section. But here binary crossover applied on parent's features instead of 0 and 1. Recombination is applied probabilistically, Each pair (or group) of parents have a probability, pc, of producing offspring. Usually, a high crossover probability (also referred to as the crossover rate) is used.

Most of the crossover operators for binary representations are sexual, being applied to two selected parents. If $x_1(t)$ and $x_2(t)$ denote the two selected parents, then the recombination process is summarized in Algorithm (3). In this algorithm, m(t) is a mask that specifies which bits of the parents should be swapped to generate the offspring, $\tilde{x}_1(t)$ and $\tilde{x}_2(t)$. Several crossover operators have been developed to compute the mask: One-point crossover, Two-point crossover, Uniform crossover [2].

Uniform crossover: The n_x-dimensional mask is created randomly as summarized in Algorithm (3). Here, p_x is the bit-swapping probability. If $p_x = 0.5$, then each bit has an equal chance to be swapped. Uniform crossover is illustrated in Figure (1).

Algorithm 3. Uniform Crossover Mask Calculation

Initialize the mask: $m_j(t) = 0$, for all $j = 1, \ldots, n_x$;
for $j = 1$ to n_x do
    if $U(0, 1) \leq p_x$ then
        $m_j(t) = 1$;
    end
end

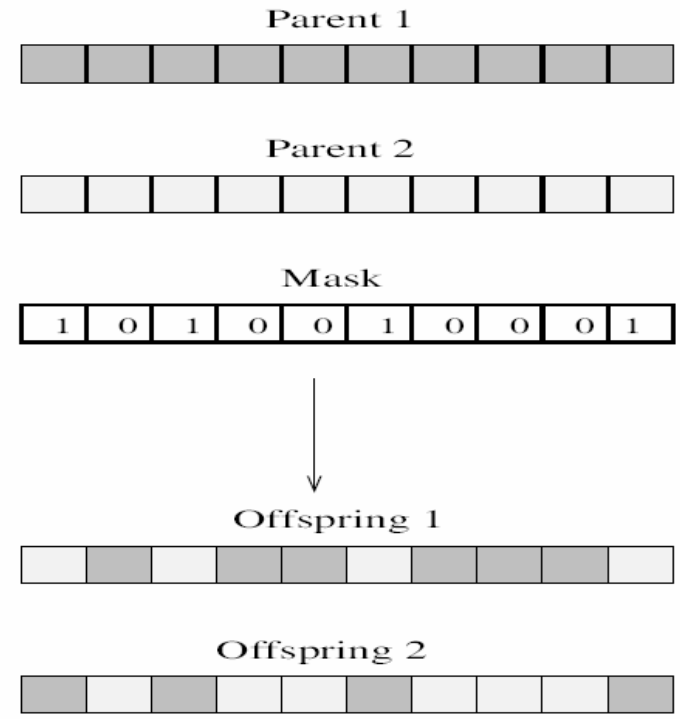


Figure 1. Uniform Crossover.

4. Somatic Mutation for real-value.

Mutation real-value attribute strings (vectors) has the same essence as mutating the other types of strings, i.e., a change is made in one or more of the attributes, but it has to respect the upper and lower limits of each attribute (vector coordinate).

In *inductive mutation*, a random number to be added to a given attribute is generated. A common mutation operator for real-valued vectors in evolutionary algorithms is *Gaussian mutation*. The Gaussian mutation alters all The attributes of a string according to the following expression:

$$m` = m + \alpha(D)\, N(0,\sigma) \quad \ldots\ldots\ldots (1)$$

where $m = (m_1, m_2, \ldots, m_L)$ is attribute string , $m`$ its mutated version, $\alpha(D)$ is a function that accounts for affinity (AIS) proportional mutation therefore is canceled here in evolutionary computation, and $N(0,\sigma)$ is a vector of independent Gaussian random variables of zero mean and standard deviation σ [5].

5. Replacement Strategy

A replacement strategy that decides if offspring will replace parents, and which parents to replace.

Two main classes of GAs are identified based on the replacement strategy used, namely generational genetic algorithms (GGA) and steady state genetic algorithms (SSGA), also referred to as incremental GAs. For GGAs the replacement strategy replaces all parents with their offspring after all offpsring have been created and mutated. This results in no overlap between the current population and the new population (assuming that elitism is not used). For SSGAs, a decision is made immediately after an offspring is created and mutated as to whether the parent or the offspring survives to the next generation. Thus, there exists an overlap between the current and new populations.

The amount of overlap between the current and new populations is referred to as the generation gap. GGAs have a zero generation gap, while SSGAs generally have large generation gaps [2][4].

A number of replacement strategies have been developed for SSGAs: Replace worst, Replace random, Kill tournament, Replace oldest, Conservative selection, Elitist, Parent-offspring.

• **Replace worst**, was used here where the offspring replaces the worst individual of the current population.

The following flowchart (see figure 2) display in summary way the preceding steps in our applying GA to solve intrusion detection problem in KDD data set.

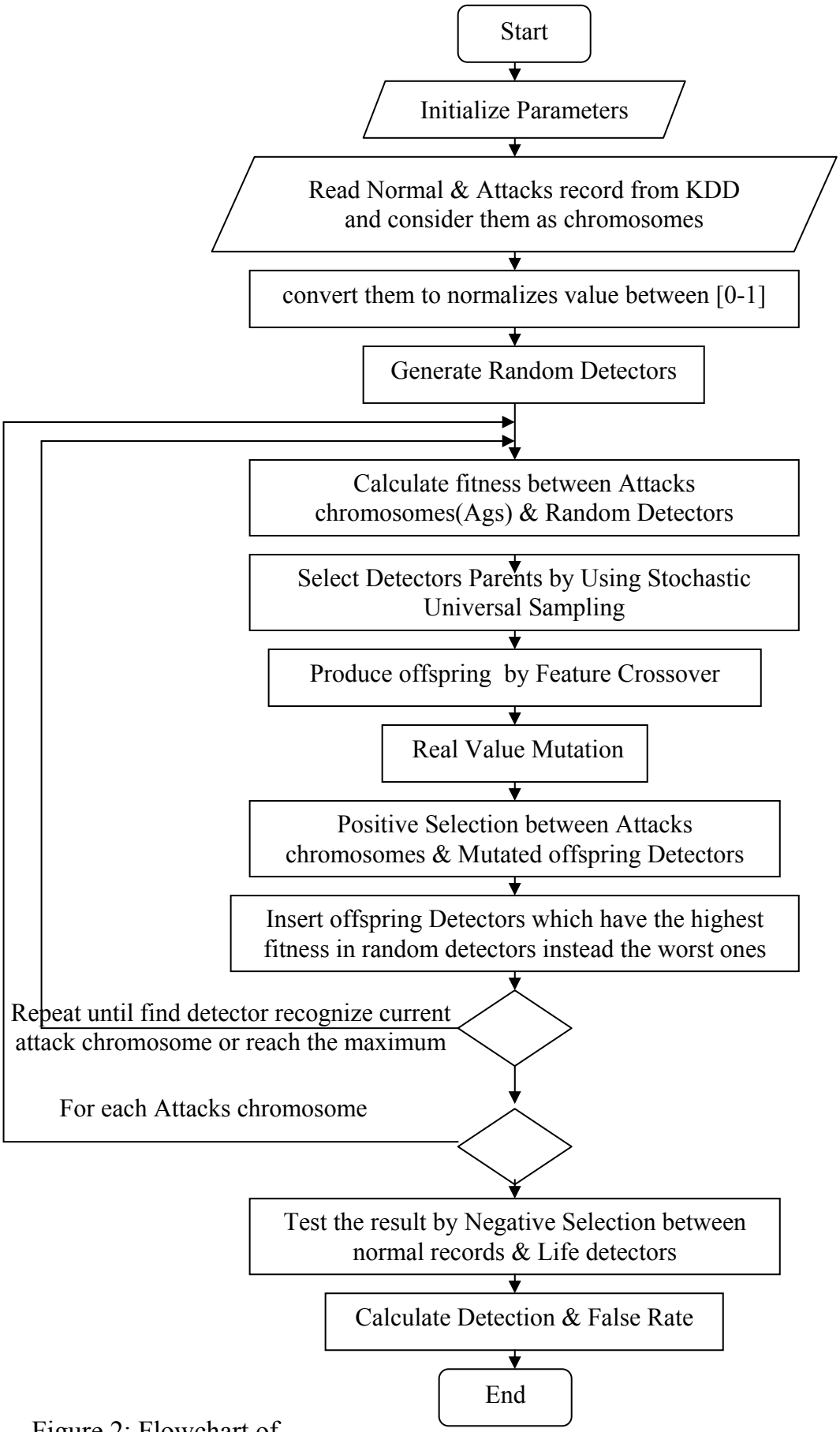


Figure 2: Flowchart of implementation GA to solve ID problem in KDD data set.

4. AIS - Learning the Antigen Structure

Learning in the immune system is based on increasing the population size of those lymphocytes that frequently recognize antigens. Learning by the immune system is done by a process known as affinity maturation. Affinity maturation can be broken down into two smaller processes namely, a cloning process and a somatic hyper-mutation process. The cloning process is more generally known as *clonal selection*, which is the proliferation of the lymphocytes that recognize the antigens.

The interaction of the lymphocyte with an antigen leads to an activation of the lymphocyte where upon the cell is proliferated and grown into a clone. When an antigen stimulates a lymphocyte, the lymphocyte not only secretes antibodies to bind to the antigen but also generates mutated clones of itself in an attempt to have a higher binding affinity with the detected antigen. The latter process is known as somatic hyper-mutation. Thus, through repetitive exposure to the antigen, the immune system learns and adapts to the shape of the frequently encountered antigen and moves from a random receptor creation to a repertoire that represents the antigens more precisely. Lymphocytes in a clone produce antibodies if it is a B-Cell and secrete growth factors (lymphokines) in the case of an HTC [2].

Since antigens determine or select the lymphocytes that need to be cloned, the process is called *clonal selection*. The fittest clones are those which produce antibodies that bind to antigen best (with highest affinity). Since the total number of lymphocytes in the immune system is regulated, the increase in size of some clones decreases the size of other clones. This leads to the immune system forgetting previously learned antigens. When a familiar antigen is detected, the immune system responds with larger cloning sizes. This response is referred to as the secondary immune response. Learning is also based on decreasing the population size of those lymphocytes that seldom or never detect any antigens. These lymphocytes are removed from the immune system. For the affinity maturation process to be successful, the receptor molecule repository needs to be as complete and diverse as possible to recognize any foreign shape [2][3].

5. Clonal Selection Theory Models

The process of *clonal selection* in the natural immune system was discussed in the previous Section. *Clonal selection* in AIS is the selection of a set of Artificial LymphoCytes (ALCs) with the highest calculated affinity with a *non-self* pattern. The selected ALCs are then cloned and mutated in an attempt to have a higher binding affinity with the presented *non-self* pattern. The mutated clones compete with the existing set of ALCs, based on the calculated affinity between the mutated clones and the *non-self* pattern, for survival to be exposed to the next *non-self* pattern [2].

In Clonal selection algorithms, each antibody and antigen is represented by a set of attributes $\{x_1, x_2, \ldots, x_n\}$. Thus, antibodies and antigens may be

represented as either n-dimensional points in a metric space such as Euclidean space or use binary encoding of the attributes; however, other representations are also used. The antigenic affinity of each antibody is typically defined based on a metric, usually, the Euclidean distance. Also, some operators are defined to introduce genetic variation to the antibodies based on their antigenic affinities. First, a cloning operator is defined to make exact copies (clones) of those antibodies having higher antigenic affinities; the higher the antigenic affinity, the higher the number of clones an antibody can generate. Then some genetic variation is introduced to these antibodies (through a mutation operator) to allow them for better matching with the antigens [3].

Clonal selection algorithms are developed based on the Clonal selection theory proposed nearly 50 years ago. The main immunological elements used are:

- Maintenance of a specific memory set.
- Selection and cloning of most stimulated antibodies.
- Removal of poorly stimulated or nonstimulated antibodies.
- Affinity maturation (hypermutation) of activated immune cells.
- Generation and maintenance of a diverse set of antibodies.

5.1 CLONALG

The *selection* of a lymphocyte by a detected antigen for Clonal proliferation, inspired the modeling of CLONALG. CLONALG is an algorithm that performs machine-learning and pattern recognition tasks. All patterns are presented as binary strings [2].

The affinity between an ALC and a non-self pattern is measured as the Hamming distance between the ALC and the non-self pattern. The Hamming distance gives an indication of the similarity between two patterns, i.e. a lower Hamming distance between an ALC and a non-self pattern implies a stronger affinity.

All patterns in the training set are seen as non-self patterns. Algorithm (4) summarizes CLONALG for pattern recognition tasks. The different parts of the algorithm are explained next [2].

The set of ALCs, C, is initialized with n_a randomly generated ALCs. The ALC set is split into a memory set of ALCs, M, and the remaining set of ALCs, R, which are not in M. Thus, $C = M \cup R$ and $|C| = |M| + |R|$ (i.e. $n_a = n_m + n_r$). The assumption in CLONALG is that there is one memory ALC for each of the patterns that needs to be recognized in DT .

Each training pattern, z_p, at random position, p, in DT , is presented to C. The affinity between z_p and each ALC in C is calculated. A subset of the n_h highest affinity ALCs is selected from C as subset H. The n_h selected ALCs are then sorted in ascending order of affinity with z_p. Each ALC in the sorted H are cloned proportional to the calculated affinity with z_p and added to set W. The number of clones, n_{ci}, generated for an ALC, x_i, at position i in the sorted set H, is defined in as

$$n_{ci} = \text{round}\ \frac{(\beta \times n_h)}{i}$$

where β is a multiplying factor and *round* returns the closest integer.

The ALCs in the cloned set, W, are mutated with a mutation rate that is inversely proportional to the calculated affinity, i.e. a higher affinity implies a lower rate of mutation. The mutated clones in W are added to a set of mutated clones, $W`$. The affinity between the mutated clones in $W`$ and the selected training pattern, z_p, is calculated.

The ALC with the highest calculated affinity in $W`$ , x`, replaces the ALC at position, p, in set M, if the affinity of x` is higher than the affinity of the ALC in set M. Randomly generated ALCs replace n_l of the lowest affinity ALCs in R. The learning process repeats, until the maximum number of generations, t_{max}, has been reached. A modified version of CLONALG has been applied to multi-modal function optimization .

Algorithm 4. CLONALG Algorithm for Pattern Recognition

$t = t_{max}$;
Determine the antigen patterns as training set D_T ;
Initialize a set of n_a randomly generated ALCs as population C;
Select a subset of $n_m = |DT|$ memory ALCs, as population $M \subset C$;
Select a subset of $n_a - n_m$ ALCs, as population $R \subset C$;
while $t > 0$ **do**
    for *each antigen pattern* $z_p \in DT$ **do**
        Calculate the affinity between z_p and each of the ALCs in C;
        Select n_h of the highest affinity ALCs with z_p from C as subset H;
        Sort the ALCs of H in ascending order, according to the ALCs affinity;
        Generate W as the set of clones for each ALC in H;
        Generate $W`$ as the set of mutated clones for each ALC in W;
        Calculate the affinity between z_p and each of the ALCs in $W`$;
        Select the ALC with the highest affinity in $W`$ as x`;
        Insert x` in M at position p;
        Replace n_l of the lowest affinity ALCs in R with randomly generated ALCs;
    end
$t = t - 1$;
end

6. Affinity Proportional Mutation rates

Here in CLONALG we also applied **Somatic Mutation** for real-value discussed in evolutionary sections, but from the viewpoint of evolution, a remarkable characteristic of the affinity maturation process is its controlled

nature. That is to say the hypermutation rate to be applied to every immune cell receptor is proportional to its antigenic affinity. By computationally simulating this process, one can produce powerful algorithms that perform a search akin to local search around each candidate solution. In equation (1) mutations borrowed from evolutionary algorithms do not account for this important aspect of the mutation in the immune system: it is inversely proportional to the antigenic affinity [5].

In this case, one can evaluate the relative affinity of each candidate solution by scaling (normalizing) their affinities. The inverse of an exponential function can be used to establish a relationship between the hypermutation rate σ(.) and normalized affinity D*, as described in equation (2). In some cases it might be interesting to re-scale α to an interval such as [0 – 1].

$$\alpha(D^*) = \exp(-\rho D^*) \quad \ldots\ldots\ldots\ldots\ldots\ldots(2)$$

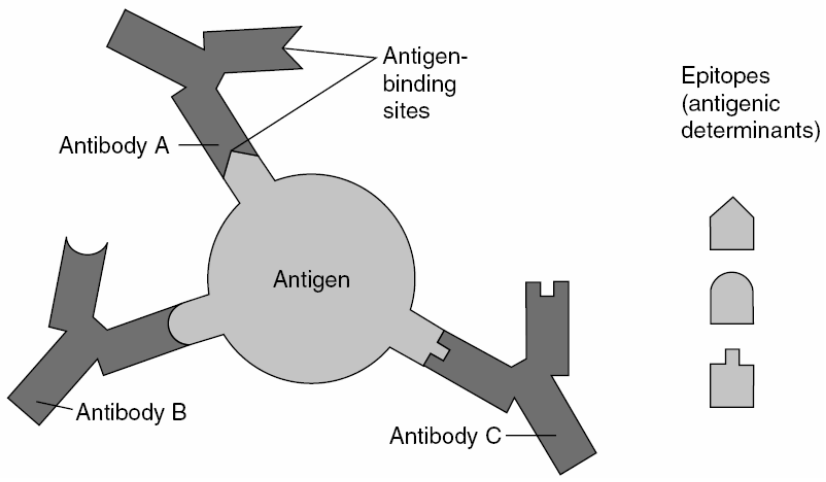
where ρ is a parameter that controls the smoothness of the inverse exponential, and D* is the normalized affinity, that can be determined by $D^* = D/D_{max}$.

7. Shape–Space and Affinity

A shape–space (or representation space) concept to represent antibody or antigen binding (see Figure 2). Accordingly, antigens and antibodies are characterized by their physicochemical binding properties, which are represented as coordinate points in such space, typically, a Euclidean space (Figure 4). Binding properties include geometric shape, hydrophobicity, charge, etc. In computational models, the notion of affinity between antibodies and antigens is defined based on a distance measure between points in the shape–space. Specifically, a small distance between an antibody and an antigen represents high affinity between them. It should be noticed that in some cases, coordinates are not given explicitly but the distance between antibodies and antigens is provided.[3]

In Figure 5, the big outer circle V, crosses (X), and small inner circles V_ε represent the shape–space, antigens, and affinity (coverage) of antibodies, respectively.[3]

Thus, ε specifies a recognition threshold; if the affinity between an antibody and an antigen (X) is less than ε (i.e., the antigen lies inside the affinity region of an antibody), then the antigen is said to match (bind) the antibody.



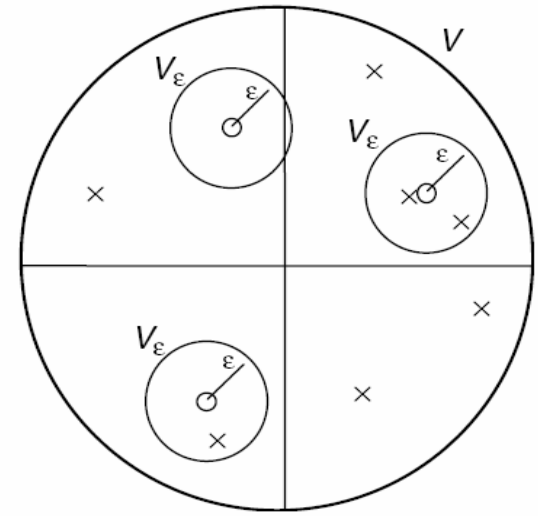


Figure 4: Antigens and antibodies are represented as points in an *N*- dimensional (Euclidean) space.

8. Real-Valued Vector Matching Rules (GA & CLON ALG)

Some distance measures that have been used to define matching rules in real-valued vector representation are explained as the amount of difference between two objects [3].

Euclidean Distance

A Euclidean distance is defined as

$$d(x, y) = \sum_i (x_i - y_i)^2 = || x - y|| \quad \ldots\ldots\ldots (3)$$

Euclidean distance can be modified when all the dimensions do not have equal weights by multiplying each component of the vectors by specific weights. Other distance measures can be used to define real-valued matching rule in a similar way to Euclidean distance. The choice of distance measures mainly relies on the type of data and domain knowledge of the specific application [3].

9. Stopping Conditions (GA & CLON ALG)

The evolutionary operators are iteratively applied in an EA until a stopping condition is satisfied. The simplest stopping condition is to limit the number of generations that the EA is allowed to execute, or alternatively, a limit is placed on the number of fitness function evaluations. This limit should not be too small, otherwise the EA will not
have sufficient time to explore the search space [2].

In addition to a limit on execution time, a convergence criterion is usually used to detect if the population has converged. Convergence is loosely

defined as the event when the population becomes stagnant. In other words, when there is no genotypic or phenotypic change in the population. The following convergence criteria can be used:

- Terminate when no improvement is observed over a number of consecutive generations.
- Terminate when there is no change in the population.
- Terminate when an acceptable solution has been found.
- Terminate when the objective function slope is approximately zero.[2]

In this paper we use Termination when an acceptable solution has been found, but if not found continue until maximum number of generations.

10. AIS - Self/Nonself Discrimination

An important mechanism of the adaptive immune system is the "self/nonself recognition". The immune system is able to recognize which cells are its own (self) and which are foreign (nonself); thus, it is able to build its defense against the attacker instead of self-destructing. T cells of enormous diversity are first assembled with a "pseudorandom genetic rearrangement process" and those that recognize self-cells are eliminated before the rest are deployed into the immune system to recognize and attack foreign pathogens. Therefore, T cells go through a process of selection that ensures that they are able to recognize nonself peptides presented by major histocompatibility complex (MHC). This process has two main phases: positive selection (PS) and NS. During the PS phase, T cells are tested for recognition of MHC molecules expressed on the cortical epithelial cells. If a T cell fails to recognize any of the MHC molecules, it is discarded; otherwise, it is kept [2][3][4].

The purpose of NS is to test for tolerance of self-cells. T cells that recognize the combination of MHC and self-peptides fail this test. This process can be viewed as a filtering of a big diversity of T cells; only those T cells that do not recognize self-peptides are kept. *In the next two section we described NS and PS which are two idea from immune system but we also use them in GA to evaluate the matching between affinity or fitness of the mutated detectors (or ALCs) with Ags or self.*

10.1 Negative Selection Algorithms

This algorithm models the T cell maturation process that occurs in the thymus. Several variations of NSAs have been proposed after the original version was introduced; however, the main features of the original algorithm still remain. Particularly, the goal of NS is to cover the nonself space with an appropriate set of detectors (shown in Figure 5).

Two important aspects of an NSA are as follows:

1. The target concept of the algorithm is the complement of a self-set.
2. The goal is to discriminate between self and nonself patterns, while only selfsamples are available.

There are two steps in NSAs as follows: "detector generation" and "nonself detection." In the first step, a set of detectors is generated by some randomized process that uses a collection of self as the input. Candidate detectors that match any of the self-samples are eliminated, whereas unmatched ones are kept [2].

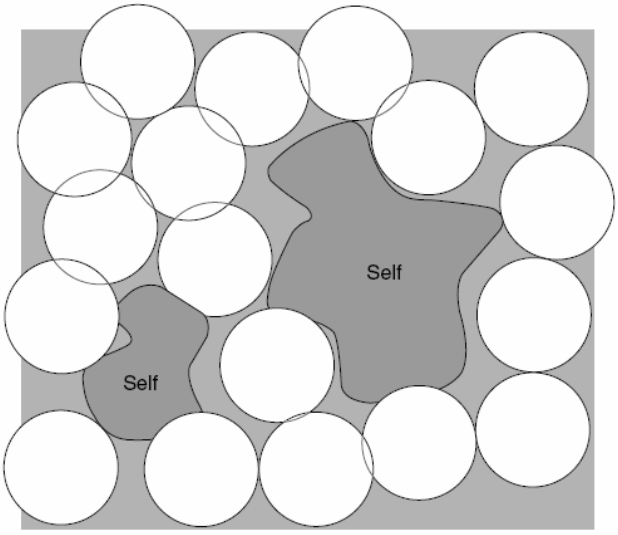


Figure 5: Illustration of the self and nonself regions.

The first step is canceled here in this paper and replaced with GA or CLON ALG, but In the detection stage, the stored detectors or ALCs (generated in the first stage) are used to check whether new incoming samples correspond to self or nonself instances. If an input sample matches a detector, then it is identified as part of nonself, which in most applications, means that an anomaly/change has occurred (see Figure 6).

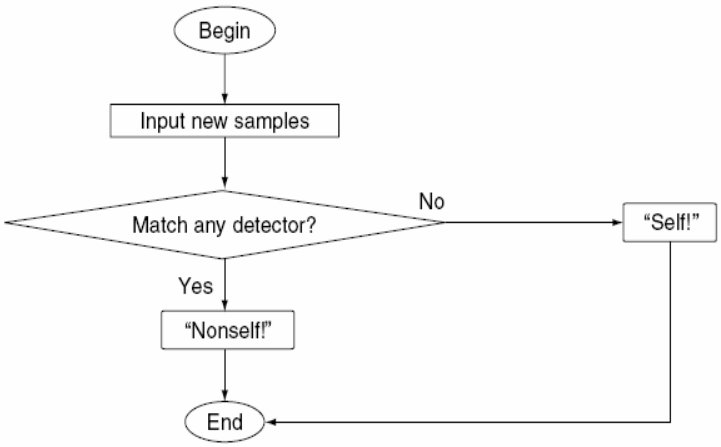


Figure 6: Monitoring phase of an NSA.

10.2 Positive Selection Algorithms

In contrast to NS, "positive detection techniques" are widely used in pattern recognition, clustering, and other domains, where they generate a set of detectors that match self-points (instead of nonself points). In this case, a model of the self-set (training data) is used to classify a sample as part of either self or nonself. A simple model of a positive detection could be built using a nearest neighbor approach. If a point lies in a neighborhood of a sample self-point, then it will be labeled as belonging to the self-set (Figure 7) [2].

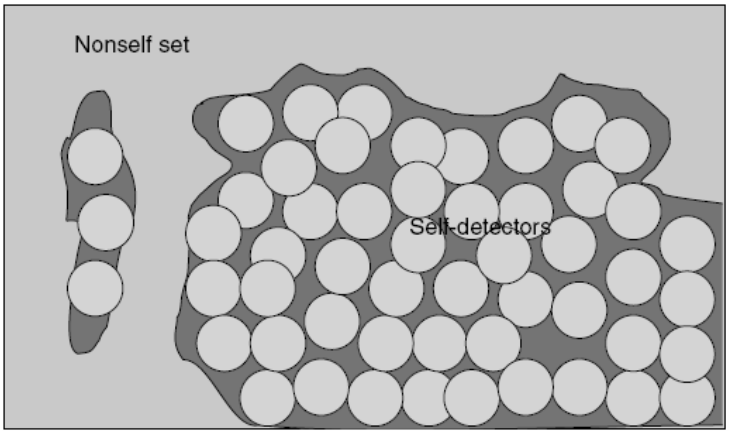


Figure 7: PS approaches. The goal of PS is to cover the self set with an appropriate set of detectors.

Generally, a positive detector defines the neighborhood by assuming a hypersphere with a certain radius centered on each of the self-points. Moreover, detectors can be defined in a more sophisticated way by using some clustering algorithm on the self-sample points. Therefore, a sample point can be classified as belonging to a cluster by measuring its distance to it. A measure of the distance from a sample to a cluster may be defined in terms of the Euclidean distance to the "cluster centroid" [5]. As shown in figure 8 if the detector match any Ag is selected and put in memory, else its rejected.

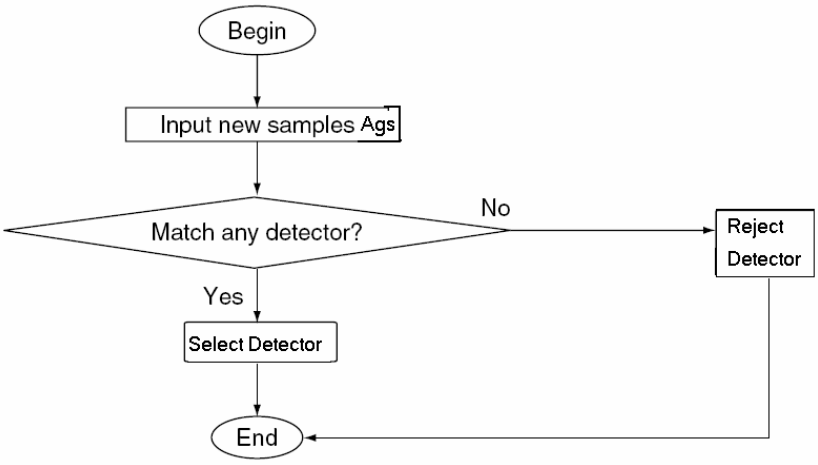
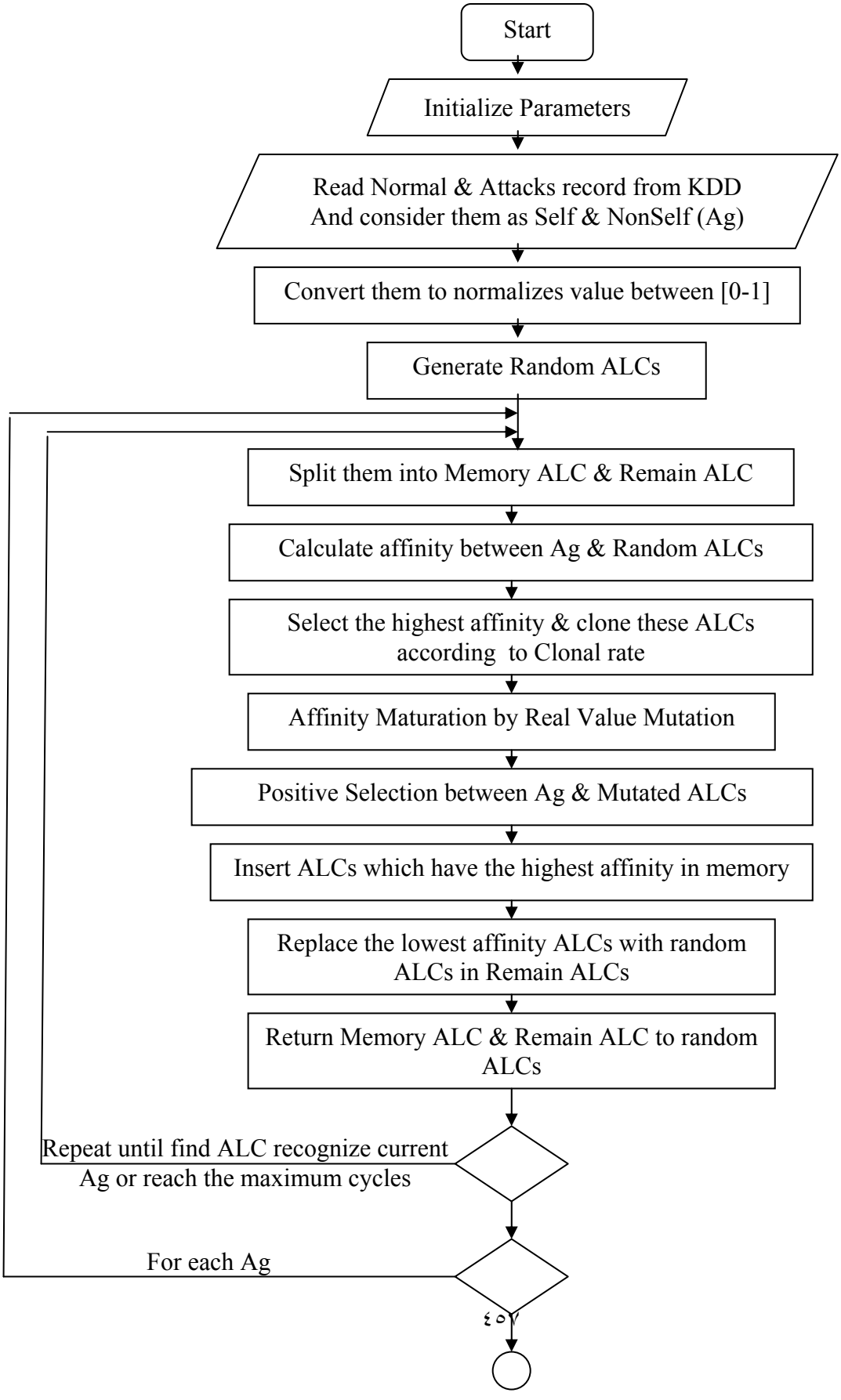


Figure 8: Monitoring phase of an PSA.

The following flowchart (see figure 9) display in summary way the preceding steps in our applying CLONALG to solve intrusion detection problem in KDD data set.



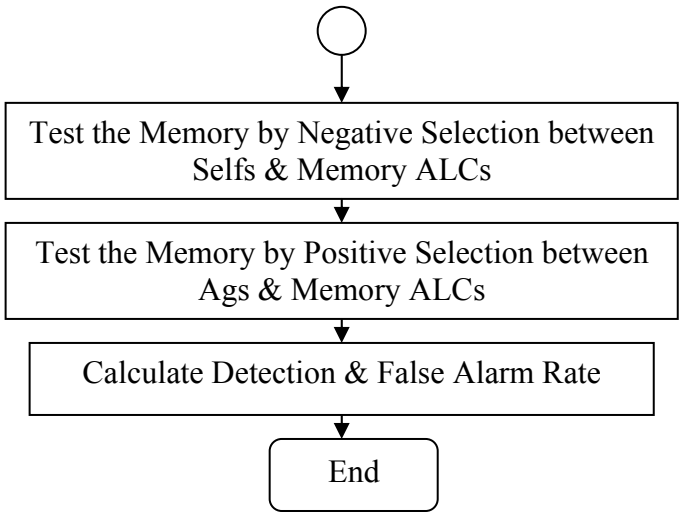


Figure 9: Flowchart of implementation
CLONALG to solve ID problem in KDD data set.

11. Evaluation Criteria (GA & CLON ALG)

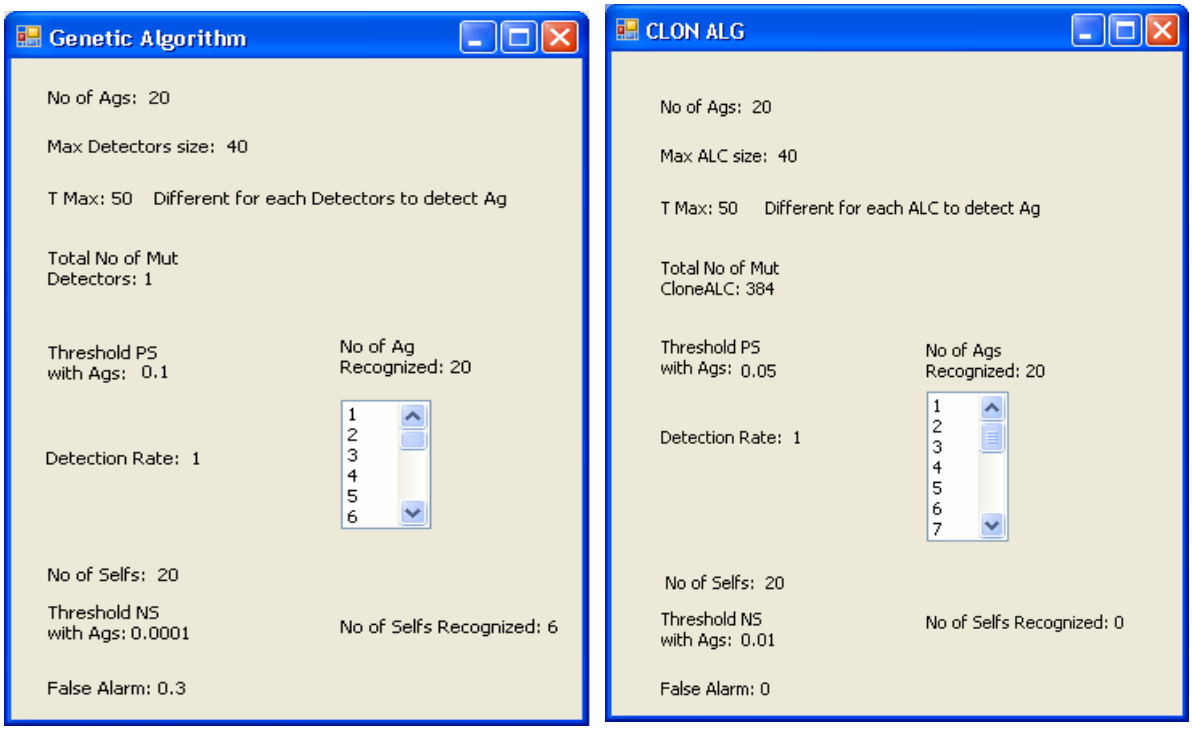
To rank the different results, there are standard metrics that have been developed for evaluating intrusion detections. Detection Rate (DR) and False Alarm rate are two most famous metrics that have already been used. DR is computed as the ratio between the number of correctly detected attacks and the total number of attacks, while False Alarm (false positive) rate is computed as the ratio between the number of normal connections that is incorrectly misclassified as attacks and the total number of normal connections [1].

12. Experimental Results

For this paper we make two separate programs: Genetic Algorithm and CLON ALG. See the result in figure 10, and the details as follow:

- **Input data**: 20 normal KDD record as Self, and 20 Abnormal KDD record as Nonself or Antigens (Ags) .
- **Max (Clonal ALC/Detectors): size**: is the size of population, its first initiate randomly and then enter the other operations.
- **Total No of Mutation (Clonal ALC/Detectors):** is the final total number of mutation Clonal ALC which can recognize Ags.
- **Positive Selection Threshold**: is a threshold used to compare with the (affinity/fitness)between Mutation (Clonal ALC / Detectors) and Ags, and must the (affinity/fitness)smaller than threshold to consider this Ag is recognized.
- **No of Ags Recognized** and their sequence number.
- **Detection Rate**: is the ratio between the number of correctly detected Ags and the total number of Ags, the desire value is one.

- **Negative Selection Threshold**: is a threshold used to compare with the (affinity/fitness) between Mutation (Clonal ALC / Detectors) and Selfs, and must the (affinity/fitness)smaller than threshold to consider this Self is recognized.
- **No of Selfs Recognized**: which is no. of normal KDD records.
- **False Alarm Rate**: is as the ratio between the number of Selfs that is incorrectly misclassified as Ags and the total number of Selfs, , the desire value is zero.



(a) Genetic Algorithm. (b) CLON ALG

Figure 10: Show results.

11. Conclusions

From applying The two models Genetic Algorithm (GA) and Clonal Selection Theory (CST) in order to detect intrusion in KDD data set, and based on the result we have obtained from them, we access to the following:

1. After studying the algorithms in fields Artificial Immune system we select Clonal Selection Theory , Negative Selection and Positive Selection to solve Intrusion Detection problem because their capabilities in detect intrusion in the input space.
2. We find some similarities between Clonal Selection Theory and Genetic Algorithm, so we decide to compare between them.
3. From the good results we observe the suitability of two models GA and CST to solve Intrusion Detection problem, but CST has more flexibility in its steps operations and has almost times the optimal

result; Detection Rate equal (1) and False alarm equal (0), but some times GA also give the optimal results.

4. The thresholds used in the two models different, and we select these value under too many experiments.
5. CLON ALG has more mutated ALC because it based on Clone the beast, while GA based on Selection method.
6. After observing the result of this research, it become from suitability to trend to the new artificial subject which is Artificial Immune System.

References

1. Adel Sabry Issa, A Comparative Study among several modified Intrusion Detection system Techniques, 2009.
2. Andries P. Engelbrecht, Computational Intelligence An Introduction, 2007.
3. Dispanker Dasgupta and Luis Fernando Nino, Immunological Computation Theory and Applications, 2009.
4. Edward Keedwell and Ajit Narayanan, Intelligent Bioinformatics The application of artificial intelligence techniques to bioinformatics problems, 2005.
5. Leandro N. de Castro and Jonathan Timmis, Artificial Immune Systems: A New Computational Intelligence Approach, 2002.