ISSN:0975-9646

Genetic Algorithm Based Intrusion Detection System

Neha Rai
Department of Computer Science and Engineering
Jai Narain College of Technology
Bhopal(M.P), India

Khushbu Rai
Department of Computer science and engineering
lakshmi narain college of Technology
Bhopal(M.P),India

***Abstract*—With the increase in network based attacks in general, and the world-wide access to computer networks and systems in particular, those responsible for network and computer system security need to utilize every tool available. To achieve this objective Genetic Algorithm Based Intrusion Detection System has been Proposed. Implementation of genetic algorithm is unique as it considers both temporal and spatial information of DARPA data set Rule Set Rule Base Network Sniffer GA network connections during the encoding of the problem; therefore, it should be more helpful for identification of network anomalous behaviors**

***Key Words*— Genetic Algorithm, Intrusion Detection System.**

I. INTRODUCTION

1.1 Genetic Algorithms are utilized in various areas of data analytics and problem solving. A branch of machine learning:

Genetic algorithm is a family of computational models based on principles of evolution and natural selection. These algorithms convert the problem in a specific domain into a model by using a chromosome-like data structure and evolve the chromosomes using selection, recombination (crossover) and mutation operators.

Genetic Algorithm (GA) has been used in different ways in IDSs. The Applied Research Laboratories of the University of Texas at Austin [14] uses different machine learning techniques, such as finite state machine, decision tree, and GA, to generate artificial intelligence rules for IDS. One network connection and its related behavior can be translated to represent a rule to judge whether or not a real-time connection is considered an intrusion. These rules can be modeled as chromosomes inside the population. The population evolves until the evaluation criteria are met. The generated rule set can be used as knowledge inside the IDS for judging whether the network connection and related behaviors are potential intrusions [14]. The COAST Laboratory in Purdue University [15] implemented an IDS using autonomous agents (security sensors) and applied AI techniques to evolve genetic algorithms. Agents are modeled as chromosomes and an internal evaluator is used inside every agent [15].

1.2 Network Intrusion Detection

Intrusion Detection can be defined as "...the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a resource."1 More specifically, the goal of intrusion detection is to identify entities attempting to subvert in-place security controls [11]. Within the overall architecture of the IIDS, some open-source intrusion detection software tools are integrated for use as security sensors [13], such as Bro [19] and Snort [20]. Techniques proposed in this paper are part of the IIDS research efforts.

1.2.1 Common types of Intrusion Detection:

1.2.1.1 Network Based (Network IDS)

Network based intrusion detection attempts to identify unauthorized, illicit, and anomalous behavior based solely on network traffic. A network IDS, using either a network tap, span port, or hub collects packets that traverse a given network. Using the captured data, the IDS system processes and flags any suspicious traffic. Unlike an intrusion prevention system, an intrusion detection system does not actively block network traffic. The role of a network IDS is passive, only gathering, identifying, logging and alerting. Examples of Network IDS:

- SNORT

1.2.1.2 Host Based (HIDS)

Often referred to as HIDS, host based intrusion detection attempts to identify unauthorized, illicit, and anomalous behavior on a specific device. HIDS generally involves an agent installed on each system, monitoring and alerting on local OS and application activity. The installed agent uses a combination of signatures, rules, and heuristics to identify unauthorized activity. The role of a host IDS is passive, only gathering, identifying, logging, and alerting. Examples of HIDS:

- OSSEC - Open Source Host-based Intrusion Detection System
- Tripwire
- AIDE - Advanced Intrusion Detection Environment
- Prelude Hybrid IDS

1.2.1.3 Physical (Physical IDS)

Physical intrusion detection is the act of identifying threats to physical systems. Physical intrusion detection is most often seen as physical controls put in place to ensure CIA. In many cases physical intrusion detection systems act as prevention systems as well. Examples of Physical intrusion detections are:

- Security Guards
- Security Cameras
- Access Control Systems (Card, Biometric)
- Firewalls
- Man Traps
- Motion Sensors

II PROPOSED WORK
GENETIC ALGORITHM BASED INTRUSION DETECTION SYSTEM

2.1 System Models

2.1.1 Overall architecture

The improved model of genetic feedback algorithm based network security policy framework consists of following components:

Gene Designer- In this module the gene of every new network event will be created based on the packets involved in the network event. These properties can be source and destination IP address and port number, size of packet, in case of security breach the level of threat and damage caused, depending on the type of security breach and etc.

Genetic Operation Unit- In this component of the model genetic operations such as crossover mutation and selection are applied to the initial set of population selected by the administrator.

Gene Pool- In this component all the gene selected during genetic operation based on their fitness score are stored along with their fitness value for future references.

Gene Checker- In this component the gene generated by the gene designer is compared with the gene present in the gene pool. If the gene is present in the gene pool then the output of the component is forwarded to network report generator. If the gene is not present then the fitness value of the gene is calculated in the fitness calculator.

Fitness Calculator- Here the fitness of gene is calculated and if the score is more than the threshold value decided for the fitness function then the gene is added in the gene pool and the output is sent to network report generator.

Network Report Generator- Here a report generated of the network event behavior after a fixed time period. This time period is decided by the network administrator. And while generating final network report two time period are taken into account i.e. present and immediate previous, and a window of size equal to the time period is created and the reports are scanned to check the event behavior. This is done because it might happen that the number of particular event occurring in a fixed time interval does not pass the threshold, which is required to take a policy enforcement decision. But while checking through the windows it is passing its threshold value at a particular instance of the window.

Policy Management Point- In this point policy management is done for the policy present in policy repository i.e. policy selection, deletion and creation and storing of policy in the policy repository. The improved model has two policy management points one for the administrator and other for the system.

Policy Repository- Here all the policy defined by the administrator or by the genetic algorithm based system are stored, so that they can be used.

Policy Decision Part- Here the policy which is to be implemented is decided. If the system is running in supervised to allow the policy then it is up to administrator to allow the policy which is enforce by the system to be enforced or just enforce his own policy. And pass on the decision to policy enforcement point via feedback console.

Feedback Console- Here if any policy is selected to be enforced or any change is made to previous policy then feedback console update to the policy repository by sending command to the policy management point.

Policy Enforcement Point- Here the selected policy is enforced on the system.

2.1.2 Fitness function

There are many parameters can influence the effectiveness of the genetic algorithm. The evaluation function is one of the most important and difficult parameter in genetic algorithm. First we define a formula to calculate whether a field of the connection matches the pre-classified data set.

$$Match\_value = E\,match * weight(from\ 1\ to\ n)$$

Where 'n' is the number of genes present in each chromosome. In our case gene means that each network event is to be checked for, here each network event is equal to a chromosome.

Some of the properties which might be considered as gene for a network are as follows:

- Source IP address
- Destination IP address
- Source port number
- Destination port number
- Size of packet
- Number of hops between the source and destination
- Time to live(TTL)
- Packet type

Above are some properties, which can be used to decide the network event fitness value. With each instance of a property there will be some weight associated with it. And the fitness score of the event will be decided by the adding the weights of all the properties. If the fitness value of the event is greater than equal to the threshold value decided, then the event is considered fit.

Here THREAT is taken as the fitness function. So the THREAT value of each event is been calculated and if it is above the threshold decided, then the packet is considered as dangerous.

$$THREAT\_VALUE = E\,THREAT\_MATCH * WEIGHT(from\ i = 0\ to\ n)$$

The absolute difference between the THREAT_VALUE of the chromosome and the actual THREAT_THRESHOLD is then computed using the following equation.

$$Sigma = THREAT\_VALUE - THREAT\_THRESHOLD$$

If the value of 'Sigma' is greater than zero then the network event is considered as dangerous or else the network event is safe.

3 FLOWCHART GENETIC FEEDBACK:

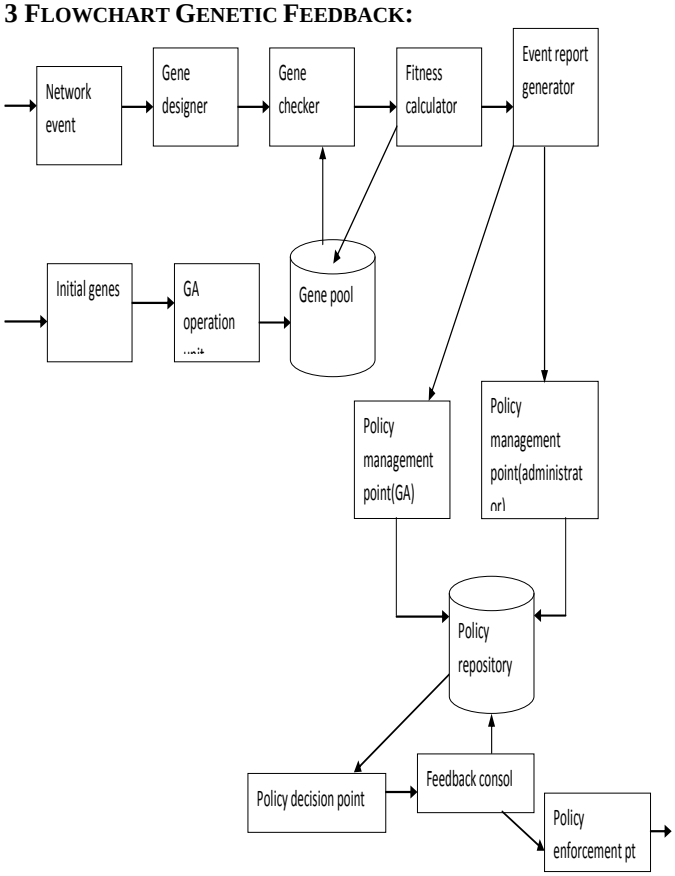

Fig 1 Genetic Networker Feedback System

4. IMPLEMENTATION DETAILS AND RESULT

4.1 Genetic Algorithm

- **[Start]** Generate random population of *n* chromosomes (suitable solutions for the problem)
- **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population
- **[New population]** Create a new population by repeating following steps until the new population is complete
    - **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
    - **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
    - **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
    - **[Accepting]** Place new offspring in a new population
- **[Replace]** Use new generated population for a further run of algorithm
- **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population
- **[Loop]** Go to step **2**

4.2 Genetic Algorithm Use in Rule Set Creation

Genetic algorithm is a family of computational models based on principles of evolution and natural selection. These algorithms convert the problem in a specific domain into a model by using a chromosome-like data structure and evolve the chromosomes using selection, recombination, and mutation operators. The range of the applications that can make use of genetic algorithm is quite broad [14, 16]. In computer security applications, it is mainly used for finding optimal solutions to a specific problem.

The process of a genetic algorithm usually begins with a randomly selected population of chromosomes. These chromosomes are representations of the problem to be solved. According to the attributes of the problem, different positions of each chromosome are encoded as bits, characters, or numbers. These positions are sometimes referred to as genes and are changed randomly within a range during evolution. The set of chromosomes during a stage of evolution are called a population. An evaluation function is used to calculate the "goodness" of each chromosome. During evaluation, two basic operators, crossover and mutation, are used to simulate the natural reproduction and mutation of species. The selection of chromosomes for survival and combination is biased towards the fittest chromosomes. GA's ability to model almost any kind of constraints in the form of penalty functions or by using various chromosome coding schemes tailored to the specific problem [27].

Figure 2 shows the structure of a simple genetic algorithm. It starts with a randomly generated population, evolves through selection, recombination (crossover), and mutation. Finally, the best individual (chromosome) is picked out as the final result once the optimization criterion is met [17].

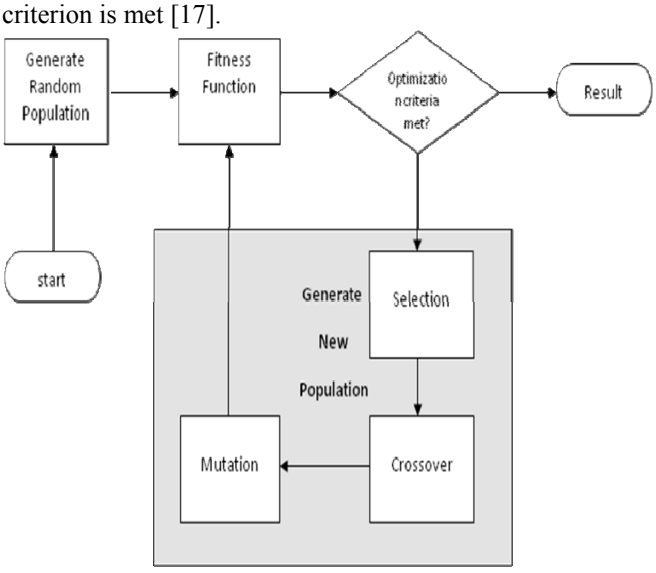

Fig 2 Structure of a simple genetic algorithm [17]

4.3 Fitness calculation

A GA Fitness Function typically has the following or similar steps. First, the general outcome is determined based on whether a gene (or allele) "matches" an existing data set of suspect log record that was obtained

from a network device such as a firewall. Then, the function multiplies the "weight" of that field to the degree that the field value "matched" the suspect record field. Typically, the "match" value is either 1 or 0 (See Figure 3).

Outcome = $\sum_{i=1}^{6} Matched * Weig.$

Weight values are applied to the different genes as historically reported by network devices. For example, if the Destination IP gene historically demonstrates to be Genetic Algorithms and Network Intrusion Detection 9 a consist predictor of a network intrusion, its weight will be more than the other genes. Moreover, all particular genes types have the same weight value so all Protocol genes have a weight of 15, regardless of their degree of being a suspect record (See Figure 3).

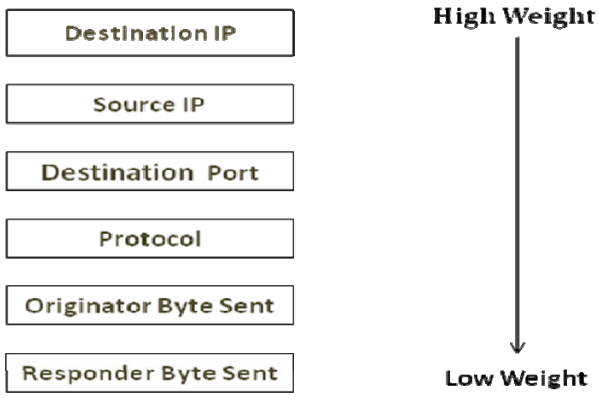

Figure 3 Gene Weight

As a clarifying example, in the case of a "gene" such as a Source IP address, let us suppose that historic data from an organization's border hardware devices such as its firewalls reveal that a Source IP address of 125.19.54.155 has attempted various intrusions targeting valuable assets such as a cluster of database systems. If the weight of a Source IP was 10, and given that the historic data supports a "match" value of 1, the outcome of the Source IP gene is 10 (10 = 1 * 10).

Next, the delta value or absolute difference between the "outcome" of the chromosome and the suspicion level is then computed using the following equation (See Figure 5).

$$\Delta = | outcome - suspicion\_level |$$

The suspicion level is a value that indicates if the historical gene value and the suspicious gene value are considered a "match" from historic log data. Continuing with our previous example, given that the Source IP of 125.19.54.155 was determined to be a suspicious IP address, the suspicion level value would be higher with a value such as 8. Therefore, the delta result is a low number of 2 (2 = |10 – 8|).

If the delta level is high enough, a penalty value is calculated using this delta (or absolute difference) (See Figure 6). The "ranking" in the equation below indicates whether or not a network intrusion is easy to establish. Historical data should determine the value of the ranking. For example, given that Destination IP addresses of certain asset systems are well known by those within an organization, this ranking would be higher.

$$penalty = \left(\frac{\Delta * ranking}{100}\right)$$

Finally, the chromosome's fitness is then computed using the above penalty. The scope of the fitness result is between 0 and 1 (See Figure 7).

$$fitness = 1 - penalty$$

Dr. Richard Fox, Associate Professor and Graduate Program Director in the Department of Computer Science at Northern Kentucky University, informed this student (by Genetic Algorithms and Network Intrusion Detection 11 personal communication, November 12, 2008) that the selection process detailed here utilizes a stochastic process which results in a random outcome allowing a range of possibilities (Stochastic Process). Dr. Fox also pointed out that there are other selection strategies that could be utilized such as those that result in the fittest chromosome in combination with the most diverse or the fittest chromosomes only. In summary, following the running of the fitness function within the GA, the fitness level is reviewed. If the desired fitness level is not obtained, the algorithm then evolves through the selection, crossover (recombination), and mutation functions.

Function to calculate fitness

```
public static int calculateFitness(int[] gene)
   {
     int fitness=0;
     try{
       ResultSet
rs=LoginPackage.LoginFrame.con.createStatement().execu
teQuery("Select * from fitness");
       if(rs.next())
       {

fitness=((gene[0]*rs.getInt(2))+(gene[1]*rs.getInt(3))+(gen
e[2]*rs.getInt(4))+(gene[3]*rs.getInt(5))+(gene[4]*rs.getIn
t(6))+(gene[5]*rs.getInt(7))+(gene[6]*rs.getInt(8)));
       System.out.println("the fitness is"+fitness);
       }
     }
     catch(Exception e)
     {
     System.out.println("Some Error Occured during
Calculating fitness: database Error");
     }

     return fitness;
  }
```

4.4 Selection in GA

Once the initial population (of chromosomes) is evaluated, the GA experiments with new generations and iteratively refines the initial outcomes so that those that are most fit are more probable to be ranked higher as results. The objective is to produce new generation of chromosomes to evaluate.

4.5 Crossover in GA

In essence, the crossover operation creates new chromosomes that share optimistic characteristics of the parent chromosomes while at the same time lowering the negative attributes in a child chromosome Figure 8 below provides an example of a crossover of chromosomes from the parents to their offspring.

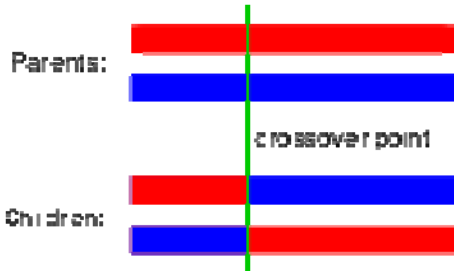

Figure 4 Crossover in GA

Although this step is typical in most genetic algorithms, in the case of this project's chromosome (see Table 4 above) the crossover operation may not be beneficial. While a Source or Destination IP may be bound by upper and lower IP settings (as demonstrated in Table 3 above), a crossover of the IP octet values would probabilistically not be advantageous. For example, the crossover of the parental values of 209.103.51.134 and 101.1.25.193 could result in child IP addresses of 209.103.25.193 and 101.1.51.134. However, the probability that this offspring will be potential suspicious Source or Destination IP addresses is low.

We use the single-point crossover and bitwise mutation for GAs and the simulated binary crossover (SBX) operator and polynomial mutation [26] for real-coded GAs.

4.6 Mutation in GA

The final step in the process of generating a new population is mutation. This phase randomly alters a gene's value to create a different one (Marakas, 2003, p. 143). Figure 9 below details how a gene's (or allele's) value is changed and thereby creating a new chromosome. Concerning the applicability of this step with the network intrusion chromosome, as was the case in the crossover step above the probability of useful outcomes is minimal.

1010010

↓

1010110

4.7 The Rule Set

Essentially, the rule set is produced from the output of the GA. For example, the input of Source IP = 1829975662 (which is an IPv4 address of 109.19.54.110)| Destination IP = 1828782356 (which is an IPv4 address of 109.1.1.20) | Destination Port = 8184 | Protocol = 5 | Originator Bytes = 10500 | Responder Bytes = 250000 could produce the following rule:

if {the connection has following information: source IP 125.19.54.155; destination IP address: 119.1.1.17 ~ 119.1.1.21; destination port number: 8184; the protocol used is FTP; the originator sent more than 10,000 bytes of data; and the responder sent more than 250,000 bytes of data } then {log the intrusion and stop the connection}

4.8 Result:

4.8.1 Scenario 1:

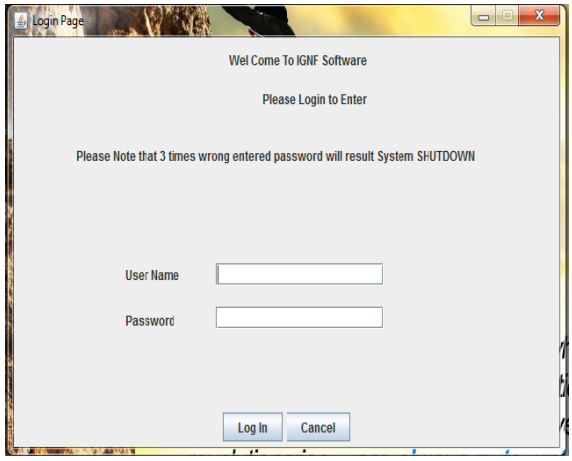

Figure 5 Admin Login form for entering in Admin area

Description:

This is the Admin login form which allows entering in the Admin area. Admin have 3 tries for entering correct username and password, after 3 tries the login window will disappear.

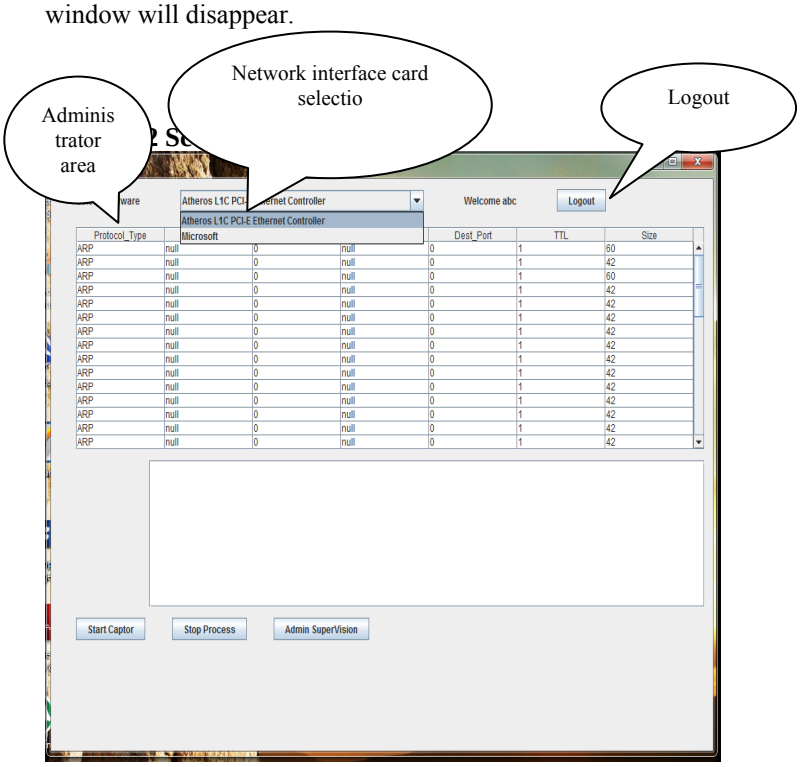

Figure 6 Main window

Description:

Main window allow Admin to choose the Network interface card to scan the packets. Admin supervision button opens Administrator window which allow Admin to change the parameter required in algorithm.

4.8.3 Scenario 3:

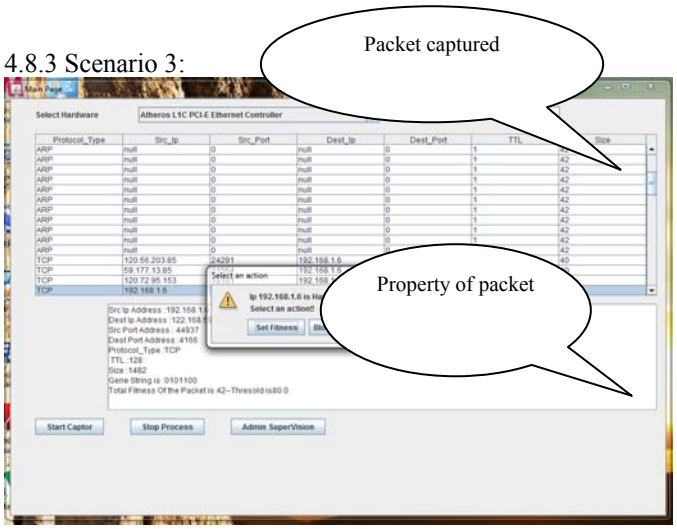

Figure 7 Action for low fitness value

Description:

If any of the packet captured is having fitness less than threshold then the admin can

- Set Fitness.
- Block IP

4.8.4 Scenario 4:

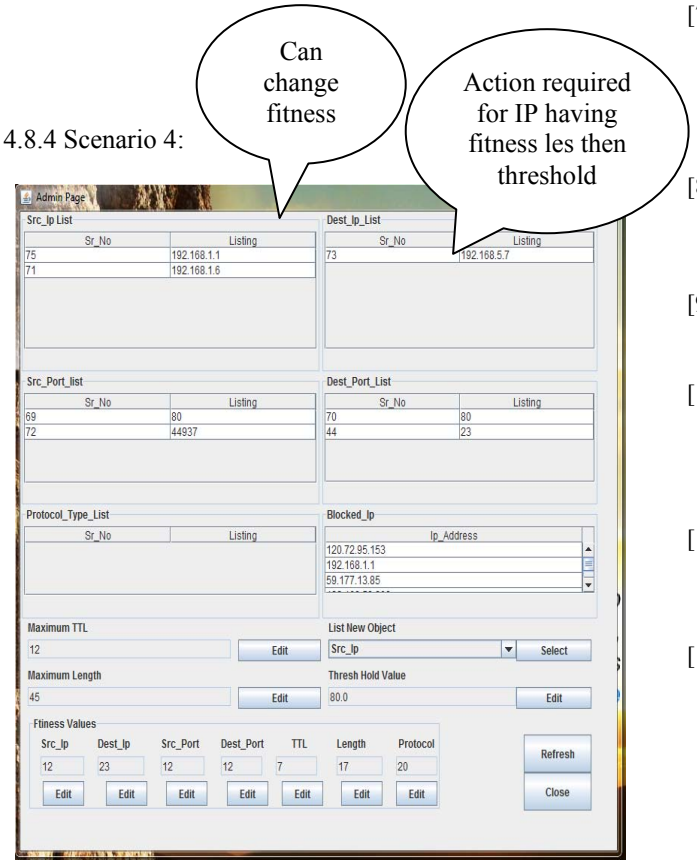

Figure 8 Admin window for managing Genetic algorithm input values

REFERENCES

[1] C. S. R. Murthy and B. S. Manoj, "Ad hoc Wireless Networks: Architectures and Protocols", Prentice Hall, 2004.

[2] Tuan Anh Le, Choong Seon Hong, Md. Abdur Razzaque, Sungwon Lee, and Heeyoung Jung, "ecMTCP: An Energy-Aware Congestion Control Algorithm for Multipath TCP" IEEE Communications Letters, VOL. 16, No. 2, pp.275 - 277, February 2012.

[3] Charles E. P, E. M. Belding-Royer and I. Chakeres, "Ad hoc On-Demand Distance Vector Routing", IETF Internet draft, 2003.

[4] M. Marina and S. Das, "On-demand Multipath Distance Vector Routing in Ad Hoc Networks", in Proceedings of the International Conference for Network Procotols (ICNP), Riverside, Nov. 2001.

[5] Sunsook Jung et. al."Energy Efficiency of Load Balancing in MANET Routing Protocols" Department of Computer Science, Georgia State University, Atlanta, Georgia 30303.

[6] Soundararajan et. al. "Adaptive Multi-Path Routing for Load Balancing in Mobile Ad-Hoc Networks" Journal of Computer Science 8 (5): 648-655, 2012, ISSN 1549-3636, Science Publications

[7] Sivakumar, P. and K. Duraiswamy, 2011. A QoS routing protocol for mobile ad hoc networks based on the load distribution. Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Dec. 28-29, IEEE Xplore Press, Coimbatore, pp: 1- 6.

[8] Bin, Z., Z. Xiao-Ping, X. Xian-Sheng, C. Qian and F. Wen-Yan, "A novel adaptive load balancing routing algorithm in ad hoc networks. J. Convergence Inform. Technol., pp. 81-85, 2010.

[9] Rajbhupinder Kaur et. al. "Load Balancing of Ant Based Algorithm in MANET" IJCST Vol. 1, Issue 2, December 2010.

[10] Sujatha.P. Terdal, Dr.V.D.Mytri, Dr. A.Damodaram "A Link Quality Based Dispersity Routing Algorithm For Mobile Ad Hoc Networks" I. J. Computer Network and Information Security, 2012, 9, 20-28 Published Online August 2012 in MECS.

[11] Yasser A. Dahab, Hesham N. Elmahdy and Imane A. Saroit " A Load Aware Routing Protocol For Mobile Ad Hoc Networks" CiiT International Journal of Wireless Communication, Vol 3, No 4, March 2011.

[12] Yahya M. Tashtoush and Omar A. Darwish "A Novel Multipath Load Balancing Approach Using Fibonacci Series For Mobile Ad Hoc Networks" International Journal of Computer Theory and Engineering Vol. 4, No. 2, April 2012 Ismail Ghazi Shayeb, AbdelRahman Hamza Hussein, Ayman Bassam Nasoura "A Survey Of Clustering Schemes For Mobile Ad-Hoc Network (MANET)" American Journal of Scientific Research ISSN 1450-223X, pp.135-151 Issue 20(2011).

[13] www.cs.cmu.se/education/examina/Rapporter/ClaesGahlin.pdf.