A Retrovirus Inspired Algorithm for Virus Detection & Optimization

Kenneth S. Edge, Gary B. Lamont, and Richard A. Raines
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Wright-Patterson AFB, Dayton, OH USA 45433
kenneth.edge@afit.edu, gary.lamont@afit.edu, and richard.raines@afit.edu

ABSTRACT

In the search for a robust and efficient algorithm to be used for computer virus detection, we have developed an artificial immune system genetic algorithm (REALGO) based on the human immune system's use of reverse transcription ribonucleic acid (RNA). The REALGO algorithm provides *memory* such that during a complex search the algorithm can revert back to and attempt to mutate in a different "direction" in order to escape local minima. In lieu of non-existing virus generic templates, validation is addressed by using an appropriate variety of function optimizations with landscapes believed to be similar to that of-virus detection. It is empirically shown that the REALGO algorithm finds "better" solutions than other evolutionary strategies in four out of eight test functions and finds equally "good" solutions in the remaining four optimization problems.

Categories and Subject Descriptors

K.6.5 [Management of Computing and Information Systems]: Security and Protection – *Invasive Software*

General Terms: Algorithms, Computer Security.

Keywords: Computer Virus, Security, Detection, Memetic, Genetic Algorithms, RNA, Function Optimization

1. INTRODUCTION

Contemporary computer systems are integral in daily life with regard to accomplishing increasing complex tasks. With this complexity we have created system vulnerabilities. There are so many interdependencies and relationships between entities in a computer system that a malicious user does not need to find a vulnerability to attack, rather they decide which one to attack.

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.

In 1995, the first year the Computer Emergency Response Team (CERT) Coordination center published statistics for vulnerabilities, they reported 171 different vulnerabilities. In 2004, the number has increased to 3,780 [2]. In 1995, there were 2,412 security incidents reported. The number has had an exponential increase to 137,529 in 2003 [2]. Viruses exploit these vulnerabilities to infect their hosts. CERT has stopped counting separate incidents in search for a more meaningful statistic of attacks since it is easy to attack a computer system. Of course, other computer security issues with increasing concern include Trojan horses, bots, backdoors, dialers, worms, adware, and spyware. The proposed AIS system can also be used to address these threats as well as viruses.

Currently most virus protection for computers is implemented using signature recognition derived after analyzing known viruses [22]. Although this method has been somewhat successful to date, we are rapidly approaching a time when such methods will be inadequate. As viruses are constantly mutated and tweaked to avoid detection, the signature list becomes larger and larger possibly approaching seven figures. Another problem is that viruses are only detected after they have been initially discovered, analyzed, and signatures have been distributed. This process can be very slow and wastes precious time while a fast replicating virus quickly renders a networked computing system useless.

What is required is a system that can combine known information from past viruses with a type of prediction for future viruses. Utilizing a computing system based on an artificial immune system may be able to achieve this. If a computer can be "immunized" initially and "learn" from viruses and other computers, and use that knowledge with evolutionary algorithms to come up with new signatures for probable viruses, we can effectively eliminate viruses before they ever have time to replicate. A variety of AIS operators have been suggested for this purpose [4,5]. This paper proposes a new type of artificial immune system (AIS) to detect computer viruses based on the REtrovirus ALGOrithm (REALGO). REALGO is based on the concept of reverse transcription RNA (see Figure 1) as found in biological systems [14]; i.e., Reverse Transcription Ribonucleic Acid (RNA).

In the following sections, generic AIS design is addressed (Section 2), the process of reverse transcription RNA is discussed and symbolically described as it applies to computer virus detection in an AIS. Next, the REALGO is mapped into the algorithm domain (Section 3), the computational domain is

presented in Section 4, and the design of experiments (Section 5) and results are detailed (Section 6). Lastly, conclusions and future work are discussed in Sections 7 and 8 respectively.

2. COMPUTER VIRUS DETECTION

AIS Design encompasses self-organization, self-regulation, primary and secondary immune system responses, clonal selection, adaptation and diversification, knowledge extraction and generalization, a network structure, metadynamics, memory, and knowledge of self and non-self [4,5,19,21,23] as evolved from biological immune system (BIS) constructs. Many of these processes are a hierarchical network integration of others and use a combination of the same operators to define their associate interactions. For example, *clonal selection* followed by *somatic hypermutation* are elements of self-organization, adaptation and diversification, and positive selection and negative selection relate to self and non-self. Note that clonal selection followed by somatic hypermutation with high selection pressure is usually defined as *affinity maturation*. The antigen-antibody BIS pattern matching is accomplished by specialized proteins attached to B-cells that bind with antigens. Of course, such proteins are generated by DNA through RNA processing (Figure 1). These proteins then, as appropriate, become gene libraries for use as initial antibodies. B-cell computational antibodies usually correspond to an AIS analyzing computer input streams for viruses, where as file structures with embedded viruses map to the combined B-cell and T-cell helper situation matching the MHC protein, the equivalent partial antigen [4, 23]. The major vehicles for the AIS adaptation process to virus protection relate to changes in the computational antibody structures through some form of mutation. In particular, the use of an affinity maturation model of antibody clones provides for new virus exploration. This brief mapping of AIS to BIS discussion is quite limited with more details found in [4, 23]. In our proposed augmented virus protection AIS, the reverse transcription RNA process is formulated as an innovative adaptive computational operator.

Most contemporary virus detectors use a scanning engine to match input streams or file structures against known virus signatures (antigens) [22]. A major AIS computational problem with this contemporary approach is that every virus signature must be stored and compared to every computer input stream or file. This implies a lot of pattern matching computation and yet it still fails to detect viruses that have not already been identified and cataloged.

The REtrovirus ALGOrithm (REALGO) operates by employing a random antibody initiation process and then, using known virus signatures (antigens), "trains" these antibodies through a genetic algorithm (GA). A distance measure is usually employed as a fitness function representing the distance (Euclidian, Hamming …) between antigen bit-string and the antibody bit string. After an initial learning phase, the antibodies are released into the computer system to recognize virus patterns. The antibodies constantly evolve using mutation (affinity maturation) in order to try to identify new viruses, adapting and diversifying the antibody memory. When a new virus is detected in a trusted network a new retrovirus signature is created and sent to other network systems in order to "immunize" them. To keep the number of antibodies to a minimum and increase performance, a temporal aspect is employed. Basically, antibodies that have not

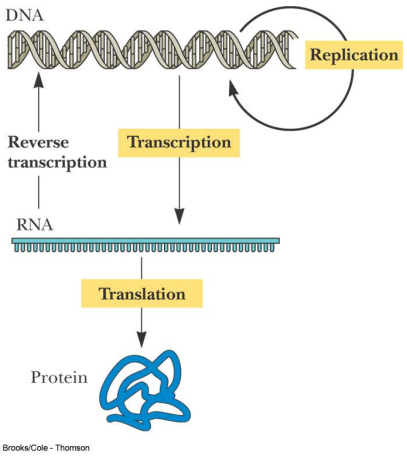

Figure 1 – Reverse Transcription in the human body [17]

been used can die after a specified time period (metadynamics) [8, 23]. If an antibody is deemed particularly important, it can be protected from elimination using a *protection flag*. On the other hand, antibodies that have been activated previously can trigger a secondary response (Figure 2), as is done in the biological immune system, as related to memory. Thus, in response to a repeated attack, the memory provides for increased efficiency (identification speed) and effectiveness.

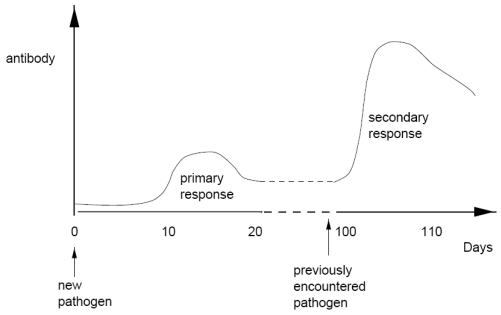

Figure 2 – Antibody Primary and Secondary Response [13]

An important concept of antibody development for detecting virus attacks is in the knowledge extraction or learning phase. This phase is further decomposed into two distinct parts; training for positive selection and negative selection. Positive selection is the ability of the antibody to detect a virus. Negative selection ensures that the antibody does not trigger on the computer's concept of "self". We do not want the antibody to identify a normal process or file (self) as a virus. The antibodies can be trained for positive selection by using a genetic algorithm on the initial population of randomly generated antibodies using a comparison operator against known virus signatures. This effectively trains the antibodies to detect known viruses. It is important to note, that these antibody to virus signature pairs are probably not exact matches. Instead they match within a specified threshold [10, 23] given a distance metric. Because they are not exact matches, we must also filter the newly trained antibodies against detecting self using negative selection techniques.

To filter the antibodies, we use a fitness function to match them against known good processes and programs in the computer (self). It is critical that the computer system has not already been

compromised by a virus since it would filter out the specific virus detection capability. If an antibody matches a good process string or program string within a certain threshold, it must be discarded. To ensure that our final population is of the desired size, enough antibodies in the positive selection phase must be generated so that some may be discarded. If we drop below the desired population during negative selection training, we must randomly generate new antibodies and train them with positive selection. If after negative selection training, an extra number of antibodies exist; simply discard the ones with the lowest positive selection fitness values.

The REALGO AIS as designed incorporates all of the salient features mentioned at the beginning of this section. It is important to observe that similar systems for detecting computer attacks were developed and implemented by Forrest, et al in [7, 12], Harmar, et al in [10, 11], and Williams, et al [24]. The main difference is our use of the RNA as a *memory* structure the search can use to revert back to a previous "good" solution if the search stagnates and fails to produce a better solution. After reverting back to this "good" solution, the search can proceed in a different "direction" to try and find a "better" solution. This operation is expected to increase the performance of the search algorithm in both efficiency and effectiveness.

3. A SYMBOLIC REALGO MODEL

In order to formally understand and apply the REALGO algorithm, a symbolic notation is discussed. The symbolic notation is provided for the reader who desires detailed comprehension of the algorithm. The following symbolic model is based on notation for genetic algorithms in [1]:

Initializing the time variable and the population of antibodies is generated randomly,

t:=0;

3.1 initialize: $P(0) \coloneqq \{\vec{a_1}(0),...,\vec{a_u}(0)\} \in I^u$

Next the population of antibodies is evaluated against the training antigens (known virus signatures). For the first iteration this evaluation is equal to zero as we have not yet imported any antigens.

3.2 evaluate: $P(0) : \{f(\vec{x_1}(0),...,\vec{x_u}(0)\}$ *;*

3.3 while not terminate (P (t)) do

The termination condition is based on a threshold of matching the antibodies to the antigens. If not terminating, then the RNA structure is imported which consists of antigens. In our application, the antigens are a string of virus signatures. For the first iteration, they are known virus signatures from previous virus knowledge bases.

3.4 import RNA:

$V(t) \coloneqq \{\vec{v_1}(t),...,\vec{v_u}(t)\} \in I^u$;

After importation, the population is compared with the imported RNA and the population member is replaced with the RNA with probability p_1 if its fitness is higher. A protection variable is checked that allows the algorithm to protect population members. We do not want to replace members that have previously had a high probability of matching to a known virus.

Select RNA: $\vec{a}_k(t) \coloneqq \vec{v}_j(t)$ with probability p_1

$$\text{if} f(\vec{v_j}(t)) > f(\vec{a}_k(t)) \forall j \in \{1,..,n\}$$

and $\rho_k \neq 1$ (Cell is not protected);

3.5 mutate:

To create new antibodies mutate members of the population to create λ children

$$\vec{a}''_k(t) \coloneqq m'_{\{\tau,\tau',\beta\}}(\vec{a}'_k(t)) \forall k \in \{1,...,\lambda\} ;$$

As part of the process of selecting the next generation of antibodies, evaluation of the children just produced

3.6 evaluate:

$$P''(t) \coloneqq \{\vec{a}''_1(t),...,\vec{a}''_\lambda(t)\} :$$

$$\{f(\vec{x}''_1(t),...,f(\vec{x}''_\lambda(t)\} ;$$

Now, check to see if the antibodies are failing to match any antigens by a specified threshold. If so, then further check if the string evolved from a RNA strand. If it did, then, revert back to the RNA string (*memory*) with probability p_2. The purpose of this check is to revert back to a known good starting point if the search for a good antibody stagnates. This operation allows the algorithm to explore from a known good starting point but in a different direction than one that produced the stagnant solution.

If $\exists f(\vec{x}''_k(t)) < \varepsilon_1$ then (If fitness falls below threshold)

If $\xi_k = 1$ (RNA based cell indicator)

$\vec{a}_k(t) \coloneqq \vec{v_k}(t)$ with Probability p_2 (Revert back to RNA)

In order to exploit an area of good solutions, the algorithm spawns new strands of RNA if a particular antibody evaluates very high based on a set threshold. If it does, the algorithm creates n copies to be used in developing the next generation.

If $\exists f(\vec{x}''_k(t)) > \varepsilon_2$ then (If fitness is above threshold value)

$V(t) \coloneqq \{\vec{v_1}(t),...,\vec{v_n}(t)\} \in I^n$ (Create n copies of RNA)

The next generation is now determined using $\mu + \lambda$ selection by selecting the best μ individuals out of the $\mu + \lambda$ population [6].

3.7 Select: $P(t+1) \coloneqq s_{(u+\lambda)}P''(t));$

The time variable is iterated and the algorithm loops back to step 6 until termination t:=t+1;

Now that positive selection is completed, the algorithm must check the antibodies against the system's concept of "self" to prevent the system from attacking good processes and programs. It is important that the system is in a known "clean" configuration at this time. If this assumption does not hold then any viruses in the system are identified as "self" and the algorithm will actually

protect them. This concept of training the antibodies against the concept of "self" is known as "negative selection".

If $\exists f(\vec{x}_k(t)) > \varepsilon_3$

then $P(t) \coloneqq P(t) - x_k(t)$

If P(t) falls below a specified number of elements then new ones must be generated using the positive selection part of the algorithm followed by negative selection until the specified number of antibodies is achieved.

Now that training is complete, the algorithm is ready to release the antibodies into the network computer system in order to patrol for viruses. As programs and/or process are started, copied, or accessed in any way, the antivirus scanner compares them to the antibodies in the system. If they match within a specified threshold, then the antivirus program checks to see if the antibody is *mature* or *immature*. If immature, then the program needs a secondary confirmation in order to take action on the suspected virus. This confirmation comes from the user. Once a virus is confirmed, the antibody is considered to be mature, sets a flag on the antibody data structure, and does not require confirmation for future viruses. Signatures that are imported can also be defined to be mature and not require confirmation by setting the mature flag. This mature flag is analogous to a *secondary response* in the human immune system [7].

Although not implemented in this system, we envision that once a virus antibody is confirmed, it is uploaded to a central repository where it is further checked by anti virus researchers. If it qualifies, the antibody could be sent to other users in order to vaccinate them on the new virus strain. The advantage of this process is that with the first variant of a virus discovered by *any* user, the inoculation could be sent out within minutes to every other user which would effectively stop the propagation of the virus almost immediately. This concept is discussed in [16].

This system does not require the researchers to actively find the virus. Instead, the virus antibody is sent to them automatically. This would drastically reduce the amount of time it takes to detect new variants of viruses by using a distributed detection system.

In order to search for antibodies that match new virus variants, the REALGO program uses a genetic algorithm with mutation to generate new antibodies. If known good building blocks exist in any of the antibodies, they can be protected and remain intact throughout the operations.

When selecting the next generation based on a fitness value, we choose between the children generated by the REALGO operations, the parents, and the antibodies received by other systems. The fitness value also takes into account the age of the antibody and the number of activations it (or its building blocks) have encountered. If there is an "old" antibody that has rarely been used, we might choose to delete it and generate a replacement based on the current threats. This operation allows the program to remain agile by maintaining a smaller signature file. We also make the assumption that at least one system finds the virus and passes the "vaccine" to the rest of the system. We are basically leveraging the power of the distributed nature of the interconnected computer systems in order to spread the burden of maintaining a large signature file.

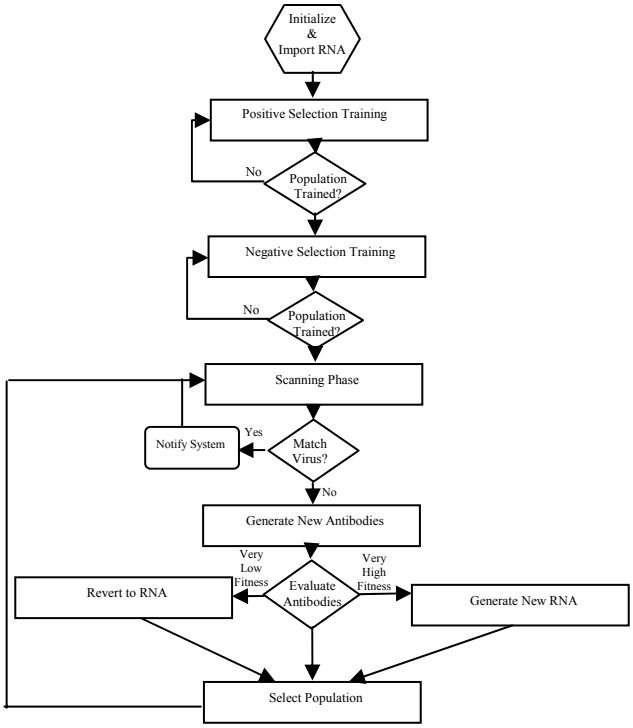

Figure 3 – Flowchart of REALGO algorithm

4. EVOLUTIONARY COMPUTATION

A genetic algorithm is the natural choice for both training the antibodies and modifying antibodies in the system once they are deployed because of the binary allele's genotype structure. A generic psuedocode for the complete anti virus scanner follows:

```
    REALGO Psuedocode
1:      Initialize population of antibodies (random)
2:      Introduce known antigens (virus signatures)
3:      for fitness of antibodies not sufficient do
4:          Compare antibodies to antigens (Genetic
            Algorithm)
5:          Evaluate antigens based on bit matching to
            antibodies
6:          Select x number with highest fitness
7:          Generate children via mutation
8:      end for
9:      if antigens based on bit matching match known
        good programs/processes then discard
10:     while (true) do
11:         if programs/processes match antibodies
            beyond specified threshold then
12:             generate warning
13:             Update data in antibody reflecting
                number of matches (fitness)
14:             if virus is confirmed then
15:                 send a copy of antibody
                    to other systems
16:                 clean system
17:         Generate new children by mutation
18:         Select parents based on fitness
19:         if fitness is high ( ε *average fitness) then
```

20: generate n copies of DNA to become RNA with prob p_1
21: **if** fitness stagnates for g generations and RNA exists **then**
22: revert back to RNA with prob p_2
23: Select population based on fitness and age
24: **end while**

The natural data structures for the virus signatures are binary strings. The strings are encoded with the signature data as well as a mature flag, a protection flag, a RNA exists flag, and a pointer to any associated RNA structure that the string initially evolved from (Figure 4). RNA structures consist of the same types of strings which means that nesting of the RNA structures can occur.

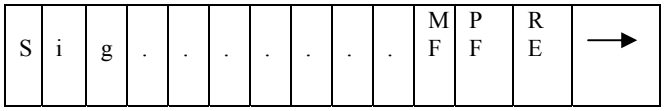

Figure 4 – Generic Signature Data Structure

For mutation, we use a Cauchy distribution as this has been shown in [25] to have the ability to make long jumps to escape local minima as compared to a Gaussian distribution. A simulated annealing time function is also used to increase exploitation after an initial period of large exploration. The combination of these two operators makes the search *memetic*.

To calculate the fitness value, as in [7], a matching rule is used. If the bits of the antigen and the antibody match for r-contiguous bits then the fitness value is computed to be r. The range of r is $0 \le r \le l$ where $l =$ signature string length

Experimental data is used to determine the ε multiplicative values for generating RNA and the number of generations of stagnation (represented as the percentage, *g*, of total number of generations in search) before reverting back to the memory RNA structure. These are basically tuning parameters that determine the rates of exploration and exploitation. It is assumed that the user has a general idea of the problem domain *landscape* and is able to initially set these parameters based on the expected search landscape.

5. DESIGN OF EXPERIMENTS

To evaluate the REALGO AIS algorithm, various procedures are defined based in part on the assessment criteria of Garrett [8]. The first step in testing the system is to ensure that the REALGO algorithm produces good results in complex search landscapes. One could consider that a computer virus search landscape would probably consist of or be close to "needles in a haystack" or delta functions on a discrete grid. But a realistic generalized virus landscape is not available since there are not any known complex models of virus selection benchmarks. Thus, other evaluation benchmarks need to be defined to evaluate the new REALGO algorithm. Function optimization was chosen to replace virus benchmarks for algorithm performance analysis. With the REALGO algorithm coded in C++, it is tested against eight well known test functions shown in Table 1 [25] that represent a variety of desired search landscape characteristics. The initial research is limited to these eight test functions as a proof of concept for the algorithm. Once the desired operation of the algorithm is confirmed based upon metrics, it would be integrated into the detection system.

Employing this functional optimization approach, results are averaged over 50 runs and results compared against those presented in [25]. Tuning parameters were developed empirically using some limited knowledge of the landscape of each problem. For example, the f6 function was graphed in a single dimension (Figure 5) to gain an understanding of the problem domain landscape. It is easy to see that there is a local minimum quite a distance away from the global minimum. This "egg carton" landscape could be thought of as a set of delta functions on a numerical grid possibly reflecting a virus signature landscape. In order to be able to jump far in this landscape, the simulated annealing time constant *t* was adjusted at a slower rate than a function such as f1 (Figure 6) which requires a much finer level of resolution in its search.

Table 1 - Test Functions Used [25].

Test function	S	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	**[-100,100]^n**	**0**
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	**[-10,10]^n**	**0**
$f_3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{n} x_j)^2$	**[-100,100]^n**	**0**
$f_4(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	**[-30,30]^n**	**0**
$f_5(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	**[-100,100]^n**	**0**
$f_6(x) = \sum_{i=1}^{n} (-x_i \sin(\sqrt{\lvert x_i \rvert}))$	**[-500,500]^n**	**-12569.5**
$f_7(x) = \sum_{i=1}^{n} [x_i^2 - \cos(2\pi x_i) + 10)]$	**[-5.12,5.12]^n**	**0**
$f_8(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}})$	**[-600,600]^n**	**0**

Once libraries of signatures and viruses are obtained, testing begins with a small sample to ensure the algorithm functions as desired in this functional optimization domain. This small sample contains 100 antibodies. After initial validation, system testing begins using the complete virus binary library available. 5-fold cross validation is used to measure detection and false positive rates [15]. The size of the libraries obtained determines the population size and the number of virus binaries tested. Based on similar work from [20], a binary library of 3000+ is anticipated to be a reasonable size for comparison purposes.

REALGO system testing is executed on a single computer platform utilizing a 2.53 GHz Pentium 4 processor with 512 MB of RAM running the Windows XP Professional operating system. Note that virus detection measurement rates are extremely non standard [9]. Once data is available from the REALGO experiments, it is compared against other published virus

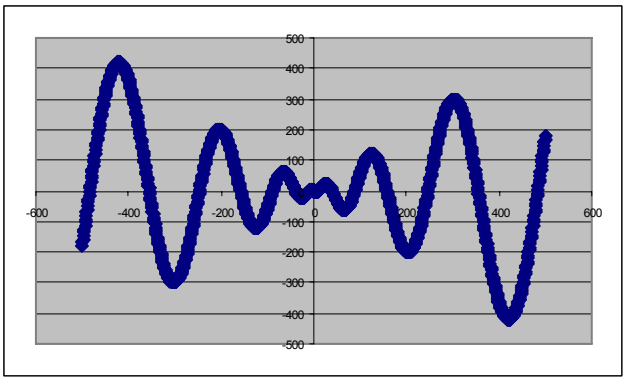

Figure 5 – The graph of f6 in one dimension. This data was used to estimate the simulated annealing constant for mutation when the search was expanded to 30 dimensions.

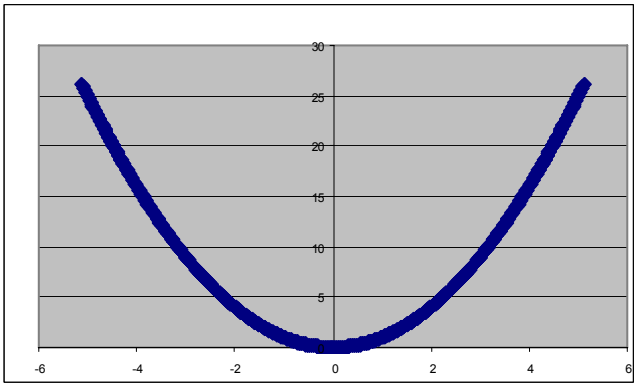

Figure 6– Graph of f1 in one dimension. This function is much smoother and requires a higher resolution search thus a smaller mutation operator which is controlled by the simulated annealing constant t.

detection algorithms such as those in [20]. The false positive rate should be close to 0 for all non virus binaries tested. Future work should include testing the artificial immune system on a distributed network. As more signatures are involved in a distributed network, the detection rate should increase. This would demonstrate the benefit of using a network of computers for REALGO virus detection. With the computers on the network cooperating, the new viruses have little chance to propagate.

6. EXPERIMENTAL RESULTS

The REALGO algorithm has been tested against eight benchmark functions as a proof of concept for the algorithm. The associated results of the REALGO algorithm are shown in Table 2. These results are compared to those from [25] (Table 3) with the resulting t-tests shown in Table 4.

Using a student t-test to compare the results, the REALGO algorithm was shown to produce better results when the RNA function was enabled on four out of the eight functions. On the remaining functions, it produced equally good results whether or not the RNA function was enabled. It never produced inferior results. When compared to the FES (Fast Evolution Strategies) algorithm from [25], REALGO outperformed FES on three of the eight functions and produced statistically equal results on the remaining five. When compared to CES (Classic Evolution Strategies), REALGO outperformed on two out of the eight functions and was shown to be statistically equal on three other functions. Two of the three that CES performed a better search (f1 and f2) were the same ones that CES outperformed FES on in [25]. CES performed better on these searches due to the simpler landscapes and the fact that CES was not burdened by the additional overhead that FES and REALGO have. On the more complex searches, the overhead was needed and was reflected in the performance measurements.

To graphically illustrate the results of REALGO we have included plots which show the means of 50 runs with a one standard deviation error bar. The other test function plots reflect similar pictorial results. The results for f1 and f8 are shown in Figures 7 and 8 respectively.

Through empirical testing, design parameters have been determined which generated the best results. These values are shown in Table 5.

Table 2 – Results of REALGO algorithm on eight test functions.

	# Gen	RNA Mean	RNA SD	No RNA Mean	No RNA SD
f1	750	9.44e-5	1.42e-5	1.63e-4	2.02e-5
f2	1000	3.37e-2	2.75e-3	4.44e-2	2.99e-3
f3	2500	3.54e-1	7.55e-2	4.41e-1	6.77e-2
f4	7500	3.27	4.06	3.14	4.09
f5	750	0	0	0	0
f6	4500	-12563.3	5.53e-1	-12577.3	109.689
f7	2500	9.44e-5	1.42e-5	6.16e-3	7.16e-4
f8	1000	1.52e-2	2.42e-3	2.46e-2	2.64e-3

Table 3- Results of FES and CES on eight test functions [25].

	FES Mean	FES SD	CES Mean	CES SD
f1	2.5e-4	6.8e-5	3.4e-5	8.6e-6
f2	6.0e-2	9.6e-3	2.1e-2	2.2e-3
f3	1.4e-3	5.3e-4	1.3e-4	8.5e-5
f4	33.28	43.13	6.69	14.45
f5	0	0	411.16	695.35
f6	-12556.4	32.53	-7549.9	631.39
f7	0.16	0.33	70.82	21.49
f8	3.7e-2	5.0e-2	0.38	0.77

Table 4 –Student t-tests for REALGO versus algorithm without RNA memory enabled, as well as REALGO versus FES and CES results. * indicates statistical significance.

	RNA-No RNA	RNA-FES	RNA-CES
f1	2.47e—3*	1.79e-2*	3.25e-3*
f2	1.63e-4*	2.18e-2*	6.26e-4*
f3	2.13e-1	1.26e-3*	1.25e-3*
f4	9.71e-1	3.15e-1	6.26e-1
f5	1	1	4.13e-1
f6	2.68e-1	7.44e-1	5.24e-3*
f7	4.43e-3*	4.90e-1	2.93e-2*
f8	1.81e-4*	5.11e-1	4.97e-1

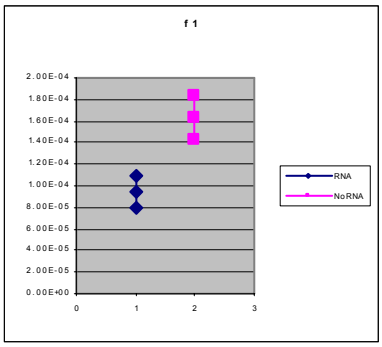

Figure 7 – Results of REALGO on function 1. The results with RNA enabled are statistically better

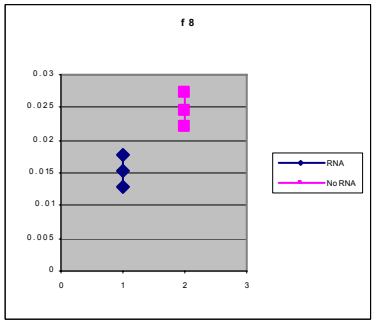

Figure 8 – Results of REALGO on function 8. The results with RNA enabled are statistically better.

7. CONCLUSIONS

The results show that the REALGO algorithm is superior for optimizing complex functions but not necessarily for easier ones. This is due to the fact that the REALGO algorithm adds complexity to the search that is not needed for simple searches. Once the complexity of the search landscape is greater than that of the algorithm, the REALGO algorithm becomes superior. For a simple search, something as basic as a hill climber suffices. If the complexity of the REALGO algorithm is used with the simple landscape, the efficiency is reduced because the complexity is not required to find the solution. Conversely, with a complex function such as f6 (Figure 5), a simple hill climber usually fails to find the best solution resulting in very poor effectiveness.

Preliminary results have shown that the REALGO algorithm does indeed provide a superior search for complex landscapes due to its ability to revert back to a previous good solution if the search stagnates. Rather than resetting to a new starting point, the search is able to attempt a search in a new direction from this previous good solution without having to waste generations for the initial convergence. The next step is to integrate it into a complete virus detector.

Table 5 – Empirically derived tuning parameters for REALGO algorithm.

Func	f1	f2	f3	f4	f5	f6	f7	f8
$\mathcal{E}$	1.03	1.03	1.03	1.03	1.03	1.03	1.03	1.03
Prob p_1	20	20	20	20	20	20	20	20
g	10%	10%	10%	10%	10%	10%	10%	10%
Prob p_2	5	5	5	5	5	5	5	5
t	500	500	0.1	500	500	0.1	500	0.1

This paper has proposed a novel artificial immune system used to detect computer viruses. The use of reverse transcription RNA as a *memory* structure for a global search has not been proposed or implemented in any literature to the authors' knowledge. The use of the reverse transcription RNA can also be generalized to other global search genetic algorithms in order to increase their performance. Additional applications of the REALGO AIS approached could include multiple robot/UAV control, fault diagnosis, and network intrusion detection (IDS).

8. FUTURE WORK

Future work involves extending the functionality of the virus detector to account for wildcards in signatures as well as signatures of variable string length. These improvements should allow the algorithm to increase its detection rates as it is able to generate the signatures using better defined building blocks. The information that is passed to future generations of antibodies only contains the most relevant parts of the signature allowing the rest of the antibody string structure to change to allow better matching to unknown virus strains. Also, benchmarks for virus search landscapes need to be develop for testing this and other AIS algorithms.

Because this is a new innovative model, advanced concepts such as encryption are ignored. Encryption is a very important part of virus detection and must be accounted for in any real world virus detection systems. Usually the encryption employed in viruses is comparatively weak and there are many methods that exist to deal with them. [22] discusses various method of generating strings to detect encrypted viruses.

An alternate approach to using byte patterns for pattern matching is that of using some type of geometric detection or shape heuristic. [20] discusses a method of using ellipsoid

detectors for negative selection. [16] also incorporates a symbolic shape-space to represent computer files. Combining a shape heuristic with the reverse transcription RNA memory function of REALGO could be promising.

Architectures for distributed virus detection system are defined in [10,11,16,18] in which REALGO could be incorporated. The REALGO algorithm can be modified from virus detection to a network IDS by changing the determination of self and non-self to represent network characteristics rather than local application characteristics.

9. REFERENCES

[1] T. Bäck, Evolutionary Algorithms in Theory and Practice. Oxford University Press. New York 1996.

[2] Cert Coordination Center Website www.cert.org as of 16 Aug 2005.

[3] D. Dasgupta, An Immunity-Based Technique to Characterize Intrusions in Computer Networks, IEEE Transactions on Evolutionary Computation, Vol 6, No. 3, June 2002.

[4] D. Dasgupta, Artificial Immune Systems and Their Applications. Springer. New York, 1999.

[5] L. de Castro and J. Timmis. Artificial Immune Systems: A New Computational Intelligence Approach, Springer-Verlag. 2002

[6] L. Eshelman, Genetic Algorithms. Evolutionary Computation 1: Basic Algorithms and Operators. Bäck, Fogel, & Michalewicz (eds.) Bristol, Institute of Physics, 2000

[7] S. Forrest and S. Hofmeyr, Immunology as Information Processing. Design Principles for Immune System & Other Distributed Autonomous Systems. Segel and Cohen, eds. Oxford University Press, 2000. pp361-387.

[8] S. M. Garrett, How Do We Evaluate Artificial Immune Systems?, Evolutionary Computation, 13(2), pp 145-178, 2005

[9] S. Gordon and R. Ford. Real world anti-virus product reviews and evaluations – the current state of affairs. In Proceedings of the 19th National Information Systems Security Conference (NISSC'96), pages 526–538, Baltimore, MD, USA, Oct. 1996. National Institute of Standards and Technology (NIST).

[10] P. K. Hammer,and G. B. Lamont, An Agent based Architecture for a Computer Virus Immune Systems, Proceedings of Artificial Immune System Workshop, Proceedings of 2000 Genetic and Evolutionary Computation Conference

[11] P. K. Harmer, P. D. Williams, C. H. Gunsch, G. B. Lamont, An Artificial Immune System Architecture for Computer Security Applications, IEEE Transactions on Evolutionary Computation, 6(3):252-280, 2002

[12] S. Hofmeyer and S. Forrest, Architecture for an Artificial Immune System,. Evolutionary Computation, Vol 8, No. 4, pp 443-473, 2000.

[13] S. Hofmeyer, An Immunological Model of Distributed Detection and Its Application to Computer Security. PhD Dissertation. University of New Mexico. 1999.

[14] S. B. Kleiboeker, Applications of Competitor RNA in Diagnostic Reverse Transcription-PCR, Journal of Clinical Microbiology, May 2003, p. 2055-2061, Vol. 41, No. 5

[15] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, IJCAI, 1995.

[16] G. B. Lamont, R. Marmelstein, D. Van Veldhuizen, A Distributed Architecture for a Self-Adaptive Computer Virus Immune System, in New Ideas in Optimization, eds. Corne, Dorigo, and Glover, McGraw Hill, 1999

[17] Lecture notes from School of Molecular and Microbial Sciences. University of Sydney, Australia. www.biochem.usyd.edu.au/MBLG2001/BMedSci/lecture%2013_Transcription.ppt as of 30 Aug 2005.

[18] T. Okamoto and Y. Ishida, A Distributed Approach to Computer Virus Detection and Neutralization by Autonomous and Heterogeneous Agents, Proceedings of the ISADS 1999 pp. 328-331.

[19] J. Percus, O. Percus, A. Perelson, Predicting the Size of the T-Cell Receptor and Antibody Combining Region from Consideration of Efficient Self-Nonself Discrimination, Proceedings of the National Academy of Sciences of the United States of America, Vol. 90, No. 5 (Mar. 1, 1993) , pp. 1691-1695

[20] M. Schultz, E. Eskin, E. Zadok, S. Stolfo. Data mining Methods for Detection of New Malicious Executables. Proceedings on the IEEE Symposium on Security and Privacy, 2001

[21] J. Shapiro, G. B. Lamont, and G.L. Peterson, An Evolutionary Algorithm to Generate Hyper-Ellipsoid Detectors for Negative Selection, Proceedings of 2005 Genetic and Evolutionary Computation Conference

[22] P. Szor. The Art of Computer Virus Research and Defense. Addison-Wesley. New Jersey 2005.

[23] A. J Timmis, T Knight, L N De Castro, and E Hart, An overview of artificial immune systems in R Paton, H Bolouri, M Holcombe, J H Parish, and R Tateson, editors, "Computation in Cells and Tissues: Perspectives and Tools for Thought", Natural Computation Series, pages 51-86. Springer, November 2004.

[24] P. D. Williams , K. Anchor, J. Bebo, G. Gunsch, G. Lamont, Warthog: Towards a Computer Immune System for Detecting "Low and Slow" Information System Attacks, Recent Advances in Intrusion Detection, RAID, 2001

[25] X. Yao and Y. Liu, Fast Evolution Strategies, Control & Cybernetics., vol. 26, no. 3, pp. 467–496, 1997