Evolving Active Defense Strategies

Sergio Caltagirone
University of Idaho
calt0563@uidaho.edu

Abstract

Active defense is a security tool that has not received much attention in research. However, it is an important topic that needs to be discussed because although not generally accepted, is still utilized informally. The largest problem facing active defense is determining what actions to take and when to take them given the risks of the actions and the risk of the threats. In this paper, competitive co-evolution is used as a technique to develop an active defense strategy by evolving an attacker and defender population in parallel and then testing them against each other. This technique has been successful and illustrates that competitive co-evolution can be a useful tool in determining security strategies.

1. Introduction

It's been no surprise to the security community that the number of attacks on computer systems has been steadily increasing over several years. [1] This pattern is more disturbing when the number of national infrastructure systems accessible over the Internet is also growing. [2] Together, this is a dangerous combination that puts the US national infrastructure at risk and threatens national security. In addition to infrastructure, medical and financial systems are equally vulnerable and accessible, placing life-critical systems in danger.

There is no question that these systems are being protected by extensive security measures such as firewalls and intrusion detection systems. However, what happens if an intruder were to bypass those defenses? It is like having a castle with very thick and tall walls, but having no security force inside – if an attacker were to tunnel under the walls, they would have full and complete access until they choose to leave. The defenses on these critical systems are good, but not sufficient to protect against the level of sophistication that intruders are reaching. Additionally, these defenses can not be maintained fast enough to keep up with the number of vulnerabilities discovered weekly; thereby providing attackers with an opportune time between when a vulnerability is known, and when a vulnerability can be protected against. Preventative defenses are no longer sufficient, especially for national infrastructure and life-critical systems.

This leaves only one choice: to implement a defense that can mitigate a threat between the time it is detected and the time that it has achieved its goal. This is the essence of active defense.

Active defense is a topic that most security researchers have shied away from. It has been portrayed as a rash tool of the vigilante. [3, 4] But in reality, active defense actions are varied in scope; from the notification of appropriate personnel, to the notification of authorities, to rewriting firewall rules, to initiating a denial of service attack against the attacker. An active defense is any set of actions taken to mitigate a threat against an asset between the time the threat is detected until the time it has completed its objectives. [5] There have been very few published instances of active defense actually being utilized. Some famous cases involve Cliff Stoll tracking German hackers in the early 80's [6] and the US Department of Defense initiating an attack against a group attempting to use a distributed denial of service attack (DDoS). [7]

However, certain companies, such as Cisco Systems, have recently begun to include 'response' technologies in their firewall and intrusion detection systems. These 'responses' limit themselves to rewriting firewall or routing rules in an attempt to block an emerging threat. [8] Yet although the technology now allows us to undertake active defense action, there is still no clear method of creating and evaluating the effectiveness of an active defense strategy. An active defense strategy is the ordered set of actions that will be taken in response to the detection of a threat.

In this paper, we will describe a novel method to derive active defense strategies using evolutionary techniques and genetic algorithms. While active defense strategies can be derived in other ways [5], an evolutionary environment can provide a unique setting where the strategies are not determined solely by calculated risk and success, but by which strategies perform best against an attacker's strategy.

More specifically, competitive evolution will be utilized so that both an attack and defend strategy are simultaneously coevolved. The strategies are then evaluated based on their performance against their evolved counterparts. In this way, we hypothesize, that using evolutionary strategies to derive active defense strategies will yield results that are reasonable based on a commonsense understanding of security and the use of active defense.

2. Background

2.1. Active Defense

There has been very little previous research directly relating to the topic of active defense. Most previous work has been applied to international policy and developing a doctrine of information warfare. In 2000, Grove, et al. in [9] attempted to determine the international legal implications of an active defense with particular focus on the UN charter and established laws of armed conflict. The paper concluded that active defense, when utilized by a nation-state, as a response to an attack by (or sponsored by) another nation-state, was an appropriate and acceptable defense with regard to established international laws and policies.

William Yurcik has also published a series of papers regarding whether the United States should pursue a policy of information warfare, and whether that policy would be (1) in the best interests of the United States, and (2) whether it would be consistent with international law. In his 2000 paper [10], Yurcik concludes that international law is vague with regard to the issue and leaves room for the United States to act if attacked via the Internet. In his 2001 paper [11], Yurcik presents a framework with which to develop an information warfare policy. This is primarily based in current rules of engagement of the United States military and the potential harm to civilian infrastructure during an attack.

Although information warfare and active defense are regularly confused because of potential offensive action, there is a significant difference between the two. Information warfare is concerned with achieving a “military advantage using tactics of destruction, denial, exploitation, and/or deception.” [11] Active defense, on the other hand, is not concerned with military advantage, and only attempts to mitigate a threat until a previous security state has been reached. This difference does not mean that information warfare research is not valuable; on the contrary, information warfare research is very valuable because of the stress on offensive action – the most questionable element of active defense.

In this experiment two topics are being combined in the hope of developing a new technique for information assurance policy and strategy creation – active defense and evolutionary computation. This is not the first time that evolutionary techniques have been brought to bear on the problems of information assurance. Most of the work combining evolution and security has been completed in the field of intrusion detection systems.

In a seminal paper on the topic [12], Crosbie and Spafford developed a prototype system, where the agents on the system were to taught to detect intrusive behavior using genetic programming techniques. To accomplish this, they developed a meta-language to examine specific aspects of the system such as network data and disk access. This language was used in the parse trees developed through the use of genetic programming, which were used as rules in the agents. If a rule was broken on the system, an agent raised the suspicion level and other agents began to look more closely at their own data by incorporating more strict rules. When a sufficient number of agents have raised the suspicion level, the level goes above a threshold and the security officer is notified of a potential system intrusion.

While Crosbie and Spafford’s work remains one of the few in the topic, some other work has continued. In [13], the authors use genetic algorithms to design an intelligent decision system for intrusion detection. They want to find a new method of limiting the number of false positives in an anomaly-based intrusion detection system (one that looks for a pattern of intrusion rather than specific behavior). Their primary objective is a classifier, which will classify and then execute an action based on the classification. They do this by evolving a population of classifiers and testing against a previously developed knowledge base. Their experiments were relatively successful in classifying the data and suggest that a system to evolve defensive security actions is possible.

Working in a similar vein was Spears and Gordon regarding the evolution of finite-state machine strategies for a defender-adversary game. [14] This work attempted to evolve strategies for a game involving two players competing for limited resources; the adversary’s strategy was fixed, while the defender evolved to beat it. Their results were promising, however limited by the existence of cycles in the strategies. Their work has potential applicability in the survivability and defense of networks.

2.2. Genetic Algorithms and Co-Evolution

Research in evolutionary computing techniques has been popular in the recent decades. Researchers in the field have created many types of evolutionary paradigms, artificial life, genetic algorithms, genetic programming, neural networks, and particle swarm among others. For our purposes, genetic algorithms and co-evolution will be focused on.

A genetic algorithm is a computation paradigm that utilizes a population of encoded chromosomes, and operations upon those chromosomes for the purpose of searching search spaces by increasing population fitness as individuals near a goal. This paradigm was first introduced by John Holland in [15]. However, although his original work describes natural competitive coevolving populations, his theories and experiments are only subject to fixed environments.

Since Holland's pioneering work, John Koza has been a leader in the field, developing and championing the technique of genetic programming. See [16] for more details about Koza's work in genetic programming. With regard to coevolution, he put forth two papers discussing his experiments in the subject [17, 18]. In these works, Koza describes a "hierarchical coevolution," which is where the environment for the first population consists of a second population and vice versa. He also describes "relative fitness," which is where an individual's fitness is determined by its performance against all of the individuals of the opposing population. He puts these into practice by attempting to evolve a game strategy using a tree structure; and succeeds in evolving the optimal strategy for each player without any direct knowledge of such strategy.

Robert Axelrod is generally regarded as the first to apply evolutionary techniques to game theory using the Prisoner's Dilemma in [19]. Prisoner's dilemma is where two players must decide whether to rat out the other person or to not talk at all. If both players rat the other person out, they receive no reward. If player one talks and player two does not, then player one gets a reward and the other player two does not. If both talk, neither gets a reward. Axelrod utilized genetic algorithms to develop a strategy for the game. Using several fixed programs submitted in a competition by others, his algorithm was able to evolve the optimal strategy of TIT-FOR-TAT.

Miller followed Axelrod's work in [20] by applying a co-evolutionary technique to the search for a Prisoner's Dilemma strategy. Instead of the tree structure used by Koza, or the linear gene structure of Axelrod, Miller utilized finite-state automata for the problem representation. His experiments showed widely varied convergence after 10 generations depending on the information about the other player available. The results of the co-evolutionary experiment showed that the top performer was just slightly less optimal than Axelrod's, but was more tolerant to short-term defections by the other player. This leads to a potential conclusion that while a co-evolutionary technique may not yield optimal strategies, the results may be very good if the game is modeled imperfectly or contains many dynamic components.

Rosin and Belew produced two papers with regard to competitive co-evolution in 1995 [21] and 1996 [22]. They used Tic-Tac-Toe, Nim, and Go as the games with which to evolve two competitive players. They developed two novel techniques, "competitive fitness sharing" and "shared sampling," which improved performance. The primary purpose of the work is to improve the "parasite" (population being tested against) population so that stronger "hosts" will be evolved. The usual method of fitness evaluation involves summing the scores during the interactions. However, "Competitive fitness sharing" is that each "parasite" is scored with the number of "hosts" that defeat it, and a host gets a fitness that is the sum of the scores of the "parasites" defeated by it. In this way a "host" is rewarded for defeating "parasites" few other "hosts" can.

The other technique developed by Rosin and Belew is "shared sampling" where a "host" is tested not against the entire "parasite" population as normally would be, but instead against a mixed set of "parasites" that tend to both defeat many "hosts," and get defeated themselves. Both of these novel techniques greatly improved the players evolved for these games.

Potter, De Jong, and Grefensttete presented [23] in 1995, which presented a solution for evolving agents with many subtasks. Their solution was to use multiple genetic algorithms, each evolving to a single subtask. When these populations had converged, they took the best of each population combining into a single agent, which was then able to effectively complete a more complex task composed of the trained subtasks. Each population was guided towards its subtask not by direct instruction, but rather by initial seeding. This is not an example of competitive co-evolution, but of a novel use of cooperative co-evolution. This work shows that complex rule-based behavior, such as active defense actions, can be successfully evolved from simpler elements.

Haynes and Sen in [24] described their (failed) attempt at co-evolving predator and prey populations. In their experiment, they used a grid with multiple predator and prey agents; where the predators could communicate together and the prey could not. They expected an arms race, where one evolutionary jump by one population is quickly matched by a counter-evolutionary jump in the other. However, the prey evolved a very simple, yet effective strategy, they all moved quickly in a straight line so that the predators were always chasing and could not surround them. This strategy fails when pitted against a greedy algorithm, and hence did not produce a novel strategy that performs better than current strategies. Possible improvements would be if the predators could predict prey action (n-look ahead), or if the predators could move quicker instead of only smarter; these improvements may have produced more complex prey strategies. The lessons from this research are that prey will always tend towards simple, effective strategies which exploit the actions, or lack thereof, available to the predators.

3. Experiment

The goal of this experiment is to discover whether it is possible to evolve an active defense strategy that is reasonable and could be considered viable enough to utilize against a threat.

3.1. Scenario

Active defense is a security tool. However, it is unlike other security tools in that it is not for general use. Active defense is a tool that must be tailored for each threat and each asset. For that reason, any attempt to experiment with active defense must provide a scenario for which the actions and risks are tailored.

For this experiment, a realistic scenario was chosen. The scenario is based on a medical patient database. This database is hosted by a medical facility that provides access via the Internet to other facilities. The data stored in the database is necessary for patient care at the facilities that use it. If the data's integrity or availability were threatened, then patient care would also be threatened. This scenario implies that the worst threats are those that compromise availability and integrity; and likewise, the riskiest active defense actions would be those that do the same.

3.2. Active Defense Modeling

The first step in developing an active defense strategy is the identification of potential actions and associated risk. On the other side, a set of attacking actions must also be defined – as well as the risks of those attacks to the asset. An important element is that both the attacker and defender also have a 'null' action available to them – which is equivalent to no action (and no risk). The null action is included because sometimes the best action is no action.

In this experiment, 12 defensive actions were defined. Those actions were given a risk amount, which is only relative to the other actions (defensive and offensive). Additionally, a flag was set if the attacker's IP address was a necessary piece of information to carry out the action; and whether the action, if successful, would permanently stop the attacker.

Table 1. Defensive Actions

Action	Risk	IP Necessary	Permanent Stop
Contact Administrator	2		
Contact CTO	5		
Shutdown Port at Firewall	1700		
Filter IP at Firewall	200	X	
Shutdown Server	2000		
Send TCP RST Packet	6	X	
Ask ISP to Shut-off Attack	10		
Contact FBI	5		X
Use Traceback	6		
Send Virus against attacker	1000	X	X
DoS Attacker	1000	X	
Hack Attacker	1500	X	X

11 offensive actions were also defined. The risks of the offensive actions are not determined by any formula, but as relative to the other actions and the value of the asset (mainly availability and integrity).

Table 2. Offensive Actions

Action	Risk
Spoof IP Address	0
Port Scan Server	0
Ping Server	0
DoS Server	800
DDoS Server	800
Poison DNS	200
Install Backdoor on Server	900
Download Records	1000
Change Records	1500
Send Virus against Server	400

A table was then created that specified which defensive actions stop which offensive actions.

Table 3. Defense Action Mitigation Matrix

Defensive Action	Stops These Actions
Contact Administrator	
Contact CTO	
Shutdown Port at Firewall	Port Scan, Ping, DoS, DDoS, Backdoor, Download Records, Change Records, Virus
Filter IP at Firewall	Port Scan, Ping, Backdoor, Download Records, Change Records
Shutdown Server	All Actions
Send TCP RST Packet	Port Scan, Ping, Backdoor, Download Records, Change Records
ISP Shut-off	Port scan, Ping, DoS, Backdoor, Download, Change Records
Contact FBI	Download Records
Use Traceback	
Send Virus Against Attacker	Port Scan, Ping, DoS, Backdoor, Download, Change Records, Send Virus against Server
DoS Attacker	Port Scan, Ping, DoS, Backdoor, Download, Change Records
Hack Attacker	Port Scan, Ping, DoS, DNS, Backdoor, Download, Change Records

The 'Use Traceback' defensive action plays an important role. It itself does not stop any actions, however if an attacker has used the 'Spoof IP' action, then traceback will find the IP of the attacker so that defensive actions that require a valid IP will then be effective. Additionally, the Contact CTO and Contact Administrator actions do not do anything very valuable, but are there because in a real active defense scenario those would are required before an active defense can be initiated (although that is not modeled here).

3.3. The Game

Active defense in this experiment is played like a game. The attacker plays one action, if that action is neither null nor 'IP Spoof', then the defender has the opportunity to counter the attack. If the defender executes the 'Use Traceback' action, then the attacker's IP address is known until the attacker executes the 'IP Spoof' action again. All of the defender's actions are iterated through until (1) there are no more defensive actions, or (2) the attack is successfully stopped. Each action that the defender takes accumulates risk, however only if the attack is not stopped is the attack risk also added. This is illustrated in the table below.

Table 4. Risk Assignment

Defender	Attacker	
	Success	*Stopped*
Success	Defender Risk	Defender Risk
Unsuccessful	Attacker + Defender Risk	Defender Risk

The algorithm for the game is as follows.

```
int risk = 0;
for-each(attackerAction){
    boolean stopped = false;
    for-each(defenderAction){
        risk += risk(dAction);
        if(permanentStop(dAction,aAction){
            return risk;
        }
        else if(stops(dAction,aAction)){
            stopped = true;
        }
    }
    if(!stopped) risk += risk(aAction);
}
return risk;
```

3.4. Genetic Algorithm

To accomplish this experiment, two populations will need to be evolved in parallel – one for attackers and one for defenders. These populations will then be placed in competition to obtain the fitness of the individuals.

Table 5. Genetic Algorithm Parameters

Paradigm	Generational
# of Populations	2
Population Size	60
# of Trials	100
Fitness Evaluations	132,000
Parent Selection	Tournament Selection
Elitism	Top 2 are kept
Mutation Type	Uniform Random Replacement
Mutation Rate	1/n
Crossover Type	2-pt crossover
Crossover Probability	100%
# Actions in Chromosome	8
# Initial Actions	4

3.4.1. Initialization
The chromosomes were randomly initialized with actions. However, they were not filled. The chromosome was first filled with null actions, then, a certain number (see Table 5. Genetic Algorithm Parameters) of random actions were chosen to be placed in random locations within the chromosome array.

3.4.2. Mutation
The mutation algorithm chosen is uniform random replacement. Each allele has probability 1/n (where n is the number of actions in the chromosome) to be randomly changed with another action. The action randomly chosen can be null; this means that the action can be 'removed.' Because no-action is a legitimate active defense strategy, this does not affect the evolution of a strategy.

3.4.3. Crossover
Two-point crossover was chosen as the method of crossover. In two-point crossover, two points are randomly chosen and the mid-section is swapped between the chromosomes.

3.4.4. Selection
Selection for crossover was done with 2-element tournament selection. In this method, two chromosomes are randomly selected, and the one with the best fitness becomes a parent in crossover.

3.4.5. Fitness Evaluation
In this experiment, the fitness evaluation follows what is normal procedure for competitive co-evolution. Each member of a population is independently tested against each member of the opposing population. The fitness of the chromosome is the sum of the risk produced in the competitions with all the elements of the opposing population.

4. Results

Using a competitive co-evolution technique, the experiment produced successful results. The results fall into two sections, population fitness and the development of a strategy. The population's fitness shows very clear competitive co-evolutionary behavior as well as an insight into the choices made by individuals in the population. The strategy developed in the experiment also shows a clear trend and supports the hypothesis that a strategy can be successfully evolved.

4.1. Population Fitness

In most genetic algorithms, the average fitness of the population over time rises or falls as it reaches a solution. However, with competitive co-evolution, this is not the case. As shown by Figure 1. Average Attack Population Fitness Over 1000 Generations and Figure 2. Average Defender Population Fitness Over 1000 Generations, the fitness of the populations quickly rise to a level then fluctuates near that level over time. More importantly is the fact that the change of the population fitness should mirror that of the opposing population.

As one population makes a change, the opposing population will make a change to counter the change. This effect has been previously shown in [21, 24]. If a close examination of the two graphs is made, the attacker population, while making more significant changes, and the defender population closely mirror each other. The reason for the larger changes in the attacker is due to the fact that the attacker is free to experiment with riskier actions, while the defender is going to usually choose the same safe actions. This conclusion about the behavior of the populations is supported by the individual choices of the chromosomes (as described in the next section).

4.2. Strategy Results

The purpose of this experiment was to determine whether competitive co-evolution was a useful technique in developing active defense strategies. After running the genetic algorithm for 1000 generations over 100 trials, a clear strategy emerged. The strategy produced is reasonable and effective.

The attackers choose a strategy that inflicts the most amount of risk while minimizing the defensive actions available to mitigate the threat. As shown by Table 7. Attack Action Frequency, the attackers normally choose to spoof their IP for the first 2-3 actions, then to poison the DNS the next 3-4 actions, and then finally either execute a DDoS attack or change the records. This strategy was developed consistently over many iterations of the experiment.

This evolved attack strategy is successful because spoofing the IP address early in the actions limits the defensive actions possible to only those not requiring the attackers IP address, or the 'Use Traceback' action. Additionally, 'Poison DNS' is an attack that is difficult to defend and therefore has the highest likelihood of succeeding. Lastly, the attacker choose to either change the records, or launch a DDoS attack because the actions necessary to mitigate the 'Poison DNS' attack do not mitigate the DDoS or changing the records – and the ones that do mitigate are very risky for the defender. Therefore, the defender is likely not to take an action but rather absorb the attack.

The defenders also choose a clear and reasonable strategy. As illustrated by Table 6. Defensive Action Frequency, the defense choose three actions very frequently; these are ‘Ask ISP to shutoff attack’, ‘Contact FBI’, and ‘Use Traceback.’ Obviously, the defender needs to execute ‘Use Traceback’ to gather the requisite IP data for other actions; and since the attackers generally use IP Spoofing early in the attack, the defenders also show heavier use of ‘Use Traceback’ in the early actions rather than the later.

The defender also used the ISP and FBI actions because those are the actions with the highest risk/benefit ratio (risk vs. stopping or permanently stopping the attack). This is also why the attackers choose to heavily use the DNS attack, because the ISP and FBI could not counteract the DNS attack.

This is a strategy also used by people who informally use active defense: get the attacker’s data (if possible), then hand the defense to another entity (either ISP or FBI) for them to stop the attack and assume the risk. This defense and attack show that competitive co-evolution can indeed derive security strategies if used with the game model.

4.2.1. Example Strategies

Here is an example attack and defend strategy developed through competitive co-evolution.

Attacker: (Spoof IP Address) (Poison DNS) (Port Scan the Server) (Port Scan the Server) (Poison DNS) (Port Scan the Server) (Hack Server, Install Backdoor) (Poison DNS)

Defender: (Ask ISP to Shut-off Attack) (Use Traceback) (Use Traceback) (Contact Administrator) (Contact FBI) (Filter IP at firewall) (Contact Chief Technology Officer) (Null Action)

5. Conclusions

Active defense is a difficult topic because uncertain risks must be weighed with benefits. However, if a method could be introduced to incorporate these issues and produce a strategy, then active defense could be used as a legitimate security tool. In this paper a new technique for developing active defense strategies was illustrated. The technique involved modeling security as a game and utilizing competitive co-evolution as the tool determining the best strategy.

The rules of the game are the actions (of both attacker and defender), the risks of the actions, the relationship of the actions, and the accumulation of risk. Then a population of defenders and attackers is initialized, and standard evolutionary techniques of mutation and crossover are applied in order to produce better strategies that will survive encounters with the opposing population.

The technique was shown to be successful. The fitness of the populations over time was shown to mirror each other, and general conclusions about the nature of the strategies emerged from the experiment. It was shown that in general, the attacking population was much more diverse in its use of actions, while (overall) the defensive population took much more conservative actions and did not overly diversify.

Additionally, the strategies that emerged were realistic and followed current active defense understandings. These strategies were that attackers spoofed their IP, then poisoned the DNS (since there are few defenses for it), and then either executed a DDoS or attempted to change the records in the server. On the other hand, the defenders took a conservative route, using the ISP and FBI to mitigate the attack because there is little risk with using third parties.

These results support the hypothesis that using evolutionary techniques, especially competitive co-evolution is successful in producing active defense solutions when coupled with a game-like paradigm. This conclusion can support extending competitive co-evolution to other security strategy production and further work into evolutionary computation as applied to computer security.

6. References

[1] Symantec, "Symantec Internet Security Threat Report," M. Higgins, Ed., 3 ed. Cupertino, CA: http://www.securitystats.com/reports/Symantec-Internet_Security_Threat_Report_vIII.20030201.pdf, 2003.

[2] CERT, "Testimony of Richard D. Pethia before the House Select Committee on Homeland Security: Cyber Security - Growing Risk from Growing Vulnerability," **http://www.cert.org/congressional_testimony/Pethia_testimony_06-25-03.html**, 2003.

[3] Reuters, "Computer Under Attack Can hack Back, Expert Says," **http://www.usatoday.com/tech/news/computersecurity/2002-08-05-hack-back_x.htm**, 2002.

[4] C. Loomis, "Appropriate Response: More Questions Than Answers," http://www.securityfocus.com/infocus/1516, 2001.

[5] S. Caltagirone, "ADAM: Active Defense Algorithm and Model," University of Idaho, 2004.

[6] C. Stoll, "Stalking the Wily Hacker," in *Communications of the ACM*, vol. 31, 1988, pp. 484-497.

[7] B. Stalbaum, "The Zapatista Tactical FloodNet," vol. 2003: http://www.thing.net/~rdom/ecd/ZapTact.html.

[8] J. Larson and J. Haile, "Understanding IDS Active Response Mechanisms," http://www.securityfocus.com/infocus/1540, 2002.

[9] G. D. Grove, S. E. Goodman, and S. J. Lukasik, "Cyber-attacks and International Law," *Survival*, vol. 42, pp. 89-104, 2000.

[10] W. Yurcik, "Information Warfare Survivability: Is the Best Defense a Good Offense?" presented at 5th Annual Ethics and Technology Conference, Loyola University, Chicago, IL, 2000.

[11] W. Yurcik and D. Doss, "Internet Attacks: A Policy Framework for Rules of Engagement," presented at 29th Research Conference on Communcation, Information, and Internet Policy, Alexandria, VA, 2001.

[12] M. Crosbie and E. Spafford, "Applying Genetic Programming to Intrusion Detection," presented at 1995 AAAI Fall Symposium on Genetic Programming, Cambridge, Massachusetts, 1995.

[13] D. Dasgupta and F. A. Gonzalez, "An Intelligent Decision Support System for Intrusion Detection and Response," presented at International Workshop on Mathematical Methods, Models and Architectures for Computer Network Security, St. Petersburg, Russia, 2001.

[14] W. M. Spears and D. F. Gordon, "Evolving Finite-State Machine Strategies for Protecting Resources," presented at the 12th International Symposium on Foundations of Intelligent Systems, 2000.

[15] J. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, Michigan: University of Michigan Press, 1975.

[16] J. R. Koza, "Genetic Programming," in *Encyclopedia of Computer Science and Technology*, vol. 39, J. G. Williams and A. Kent, Eds.: Marcel-Dekker, 1998, pp. 29-43.

[17] J. R. Koza, "Genetic Evolution and Co-Evolution of Computer Programs," in *Artificial Life*, vol. X, C. Taylor, C. Langston, J. D. Farmer, and S. Rasmussen, Eds. Santa Fe, New Mexico: Addison-Wesley, 1991, pp. 603-629.

[18] J. R. Koza, "Genetic Evolution and Co-Evolution of Game Strategies," presented at The International Conference on Game Theory and Its Applications, Stony Brook, New York, 1992.

[19] R. Axelrod, "The Evolution of Strategies in the Iterated Prisoner's Dilemma," in *Genetic Algorithms and Simulated Annealing*, L. Davis, Ed. London: Morgan Kaufman, 1987, pp. 32-41.

[20] J. H. Miller, "The Coevolution of Automata in the Repeated Prisoner's Dilemma," *Journal of Economic Behavior and Organization*, vol. 29, pp. 87-112, 1996.

[21] C. Rosin and R. Belew, "Methods for Competitive Co-evolution: Finding Opponents Worth Beating," presented at Sixth International Conference on Genetic Algorithms, San Francisco, CA, 1995.

[22] C. Rosin and R. Belew, "New Methods for Competitive Co-Evolution," *Evolutionary Computation*, vol. 5, pp. 1-29, 1996.

[23] M. A. Potter, K. A. De Jong, and J. J. Grefenstette, "A Coevolutionary Approach to Learning Sequential Decision Rules," presented at Sixth International Conference on Genetic Algorithms, San Francisco, CA, 1995.

[24] T. Haynes and S. Sen, "Evolving Behavioral Strategies in Predators and Prey," presented at IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems, Montreal, Quebec, Canada, 1996.

Appendix A: Experiment Data

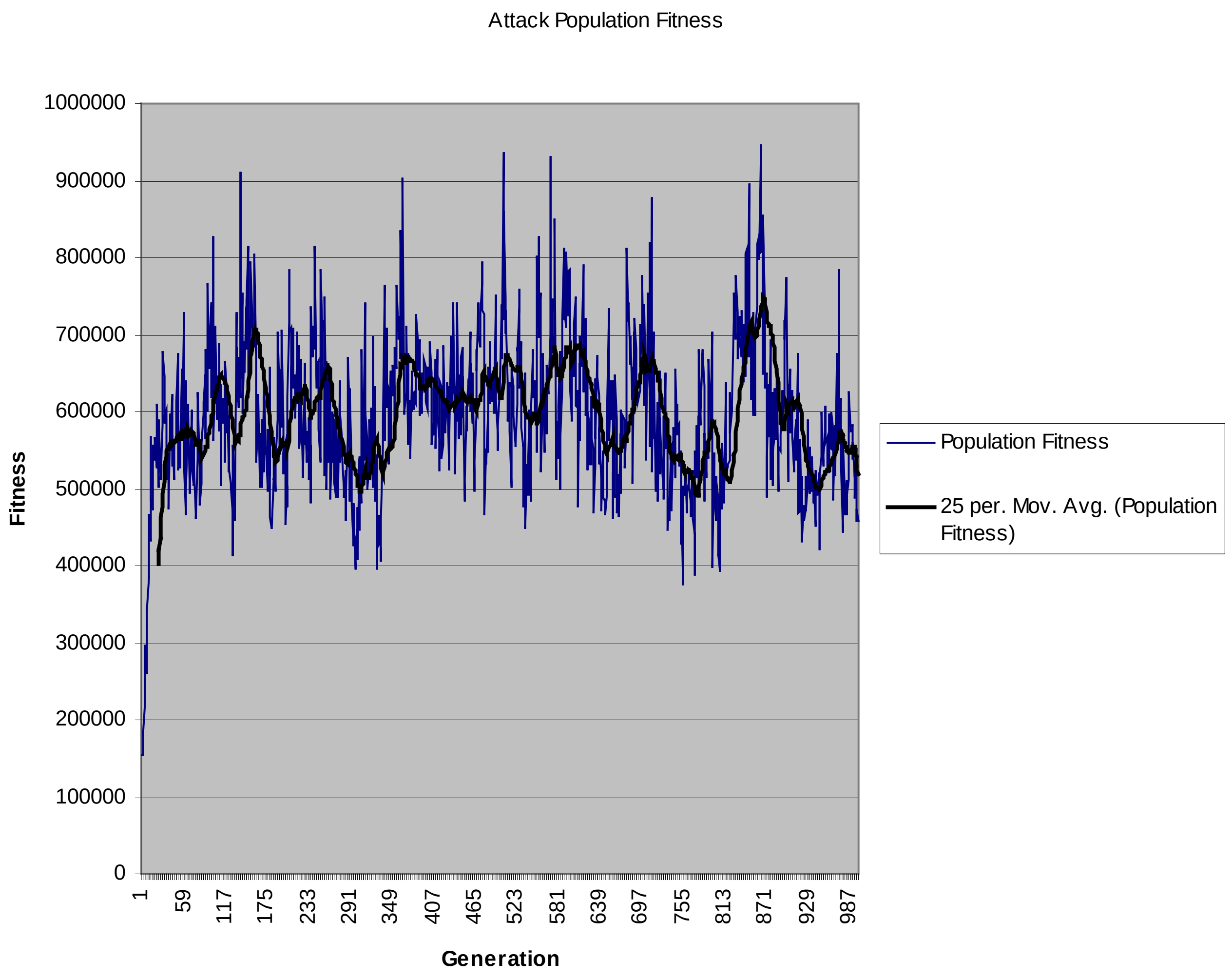

Figure 1. Average Attack Population Fitness Over 1000 Generations

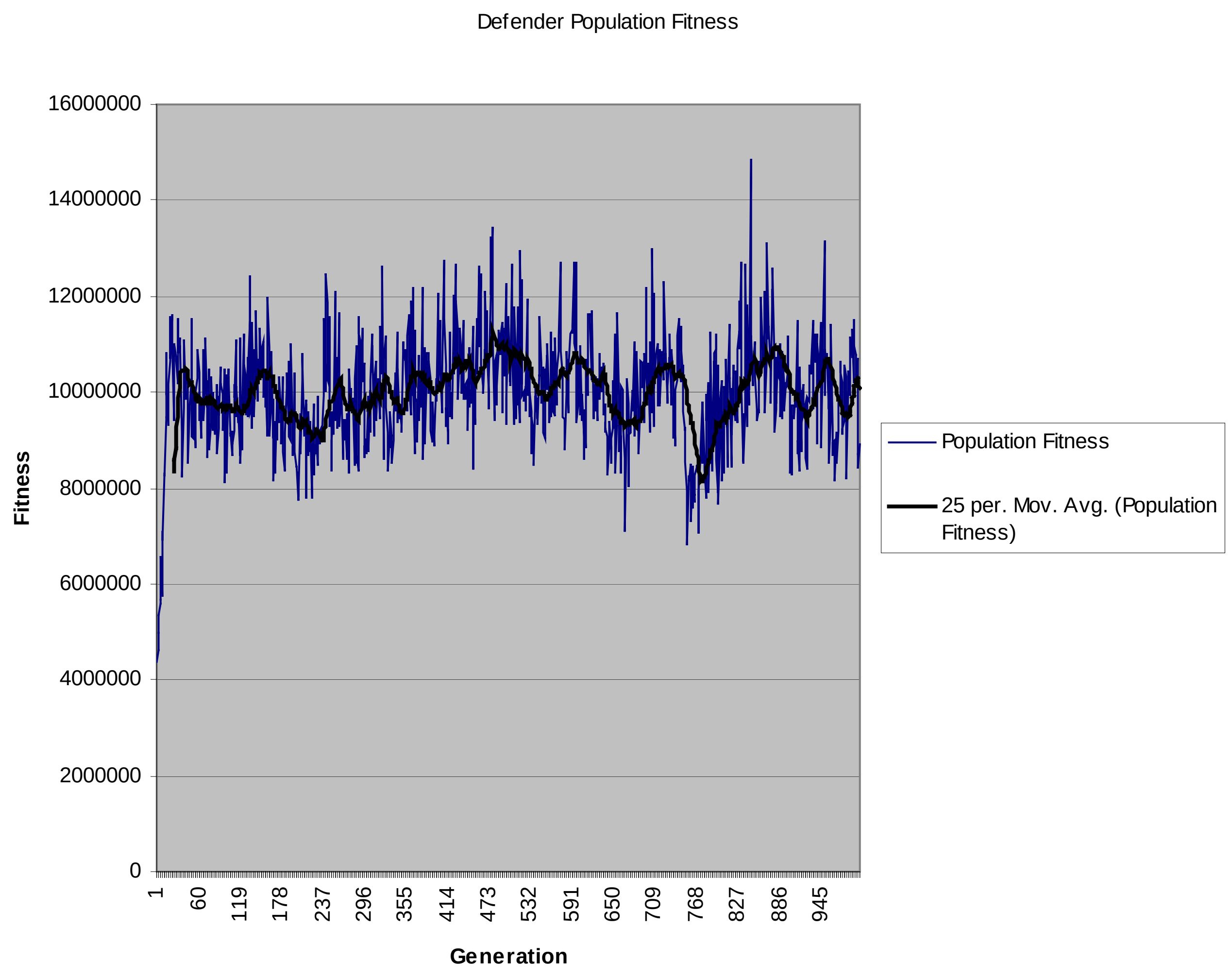

Figure 2. Average Defender Population Fitness Over 1000 Generations

Table 6. Defensive Action Frequency

DEFENSE ACTION	DEFENSE POSITION							
	0	1	2	3	4	5	6	7
Null Action	58	58	57	48	57	53	50	52
Contact Administrator	8	2	5	6	6	10	5	5
Contact Chief Technology Officer	3	2	2	6	9	5	7	9
Shutdown port at firewall	0	0	0	0	0	0	0	0
Filter IP at firewall	0	1	1	2	2	1	0	2
Shutdown Server	0	0	0	0	0	0	0	0
Send TCP RST Packet	3	4	6	5	6	5	7	5
Ask ISP to Shut-off Attack	7	15	7	10	9	7	18	11
Contact FBI	4	2	5	4	1	5	3	7
Use Traceback	17	16	17	19	10	14	10	9
Send Virus Against IP	0	0	0	0	0	0	0	0
Initiate DoS Against IP	0	0	0	0	0	0	0	0
Attempt to Hack Attacker	0	0	0	0	0	0	0	0

Table 7. Attack Action Frequency

ATTACK ACTION	ATTACK POSITION							
	0	1	2	3	4	5	6	7
Null Action	54	51	56	48	56	43	46	49
Spoof IP Address	39	24	19	7	4	2	0	3
Port Scan the Server	0	4	6	7	6	5	6	1
Ping the Server	0	1	0	2	3	2	5	1
DoS the Server	0	0	0	0	0	2	2	4
DDoS the Server w/ Zombies	0	1	0	2	2	6	6	5
Poison DNS	7	12	8	17	10	12	8	11
Hack Server, Install Backdoor	0	1	2	2	1	7	4	3
Hack Server, Download Records	0	0	1	0	2	4	2	4
Hack Server, Change Records	0	2	7	8	10	10	13	12
Send Virus Against Server	0	4	1	7	6	7	8	7