Self Organized Multi-Agent Entangled Hierarchies for Network Security

Eric M. Holloway and Gary B. Lamont
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Wright-Patterson AFB, Dayton, OH 45433
eric.holloway@afit.edu, gary.lamont@afit.edu

ABSTRACT

Cyber-security management is of central importance in financial, business, and military environments. Thus, we propose an effective and efficient self organized entangled hierarchial architecture of multiple agents that decentralizes network security control and communication. Considering the Cyberspace container model, the self organized multi-agent swarms are evolved based on partially observable Markov decision process formal models. Desired network security swarm behaviors are defined and formalized to interact with these models. The "optimal" policy (agent rules and parameters) for a given behavior is evolved using a multi-objective evolutionary algorithm. Swarm effectiveness is compared in various network security scenarios using statistical testing techniques and visualization.

Categories and Subject Descriptors

I.2.11.d [**General Methodologies**]: Artificial Intelligence—*Distributed artificial intelligence, Multiagent systems*; I.2.m.d [**General Methodologies**]: Artificial Intelligence—*Miscellaneous, Evolutionary computing and genetic algorithms*; K.6.5.c [**Computing Milieux**]: Management of Computing and Information Systems—*Miscellaneous, Security*

General Terms

Algorithm, Design, Security

Keywords

Multi-agents, Multiobjective Optimization, Cybercraft

1. INTRODUCTION

Observing the increasing number of computer network threats, intrusions and internal anomalies, network security is a serious concern in a large variety of applications. To address this situation, the use of outside intrusion detection systems, insider covert network detection, and system anomaly detection techniques have become important security tools in cyberspace. Such pattern recognition elements use bio-inspired approaches from artificial immune system constructs [9] to particle swarm approaches have been incorporated [8]. Nevertheless, such network security systems are generally fairly slow and limited when automatically operated.

Therefore to provide better effective and efficient real-time performance for cyberspace security management, a swarm of autonomous self organized agents is proposed that evolve a *non-hierarchical* entangled. This paper develops a self-organized multi-agent swarm (SOMAS) system in order to evolve behaviors for a variety of network security scenarios [8]. This multi-agent system uses the formal structure of a Markov decision process (MDP)as the design foundation [11] which leads to the partially observable MDP, the POMDP. Note that when the multiple agents cannot obtain a full observation of the network security environment, they can be represented by the POMDP model. Desired swarm security behaviors or objectives are formalized to interact with various POMDP models. The "optimal" policy (agent rules and parameters) for a given behavior is evolved using a multi-objective evolutionary algorithm. The objective is to test and evaluate this SOMAS architecture for a number of desired network security behaviors through statistical and visualization techniques.

The paper discusses current computer network defense in Section 2. Section 3 briefly describes self organization and a quantification metric. Formalizing a model for the multi-agent security domain is developed in section 4. The SOMAS architecture using offline and online "optimization" techniques is covered in section 5 including POMDPs. The design of experiments is covered in 6, with the testing results and analysis following in section 7.

2. COMPUTER NETWORK DEFENSE

Network security threats run the gambit from internal to external attacks. Particular threats include web based insider attacks, denial of service attacks, Botnets, and information exploitation and corruption [8]. Network defense of these attacks can be structured from a centralized or decentralized framework. Such elements of network security are addressed.

Counter Defense: The defenders of information are the in-

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.
ACM 978-1-60558-505-5/09/07.

direct, yet primary, target of malicious agents; since defense keeps attackers from desired information. In order to defend a computer network from various threats, one must focus on the level of defense and the process of defense. Thus, the developer should consider the following in the design network security systems:

Secure Middleware: To defend against a network attack, a comprehensive solution is needed, allowing security to quickly be pushed out to all nodes and rapidly report back incidents. This security solution can be considered a form of middleware, a distributed computing system that all users of the network interact within. The Air Force is developing secure middleware called Cybercraft. While the exact implementation design is still in development, it is based on a container model. Each node receives and deploys software "payloads" governed by a policy.

Efficiency and Flexibility of Distributed Systems: Stytz, et al [17], argue military networks require a distributed intelligent agent framework for security to avoid the weaknesses of centralized control structures, i.e. lack of scalability, single points of failure, fragility. Servat and Drogoul [15] predict future networks are characterized by a ubiquity of mobile end devices, even nanotechnology. Such large, heterogeneous environments make centralized control extremely difficult and costly, further imply ing a MAS is necessary for system control.

In [12], the authors argue a MAS is given a greater range of ability by mobilizing software agents. Thus, we realize that there is a danger of our proposed system devolving into chaos; mobility adds new degrees of freedom, threatening to produce unstable systems.

3. SELF ORGANIZATION

In order to develop MAS based security systems, it is necessary to create a scalable, flexible control structure. However, a scalable, flexible control structure tends to be an anachronism in computer science, as added flexibility and scaling leads to rapidly increasing overhead. Consequently, as evidenced by the literature, research is drawing inspiration from the many successful large scale, self organizing MASs in nature [4] [15] [13] [14].

Many scientists have developed methods of creating or measuring a particular characteristic of self organization. For instance, a set of robots that group together can be said to show self organization [2]. A decentralized intrusion detection system that uses two agent populations to detect and eliminate intrusions relies upon self organization for the populations to work together without centralized control [7]. Robots that evolve specific, global behaviors are self organizing as well [2].

Other researchers have developed more general metrics for self organization that can be applied to a general class of problems, or ways of characterizing the conditions for self organization. [5] lists and relates the general characteristics of self organization. [18] describes a set of conditions that lead to emergence of global properties (behaviors) from local agent interactions, one essential characteristic of self organization. A generalized metric for self organization has been developed by [16] based on predictive information. The predictive metric is the most general of the set, and is fairly straightforward to implement. The essential idea is that as the number of chronological sequences (l^-, causal states) that can be used to predict future sequences (l^+) in a history increases, then the system is developing more structure, and hence becoming more organized. Per Shannon's information theory, the $\log_2$ of the number of causal states measures the amount of information contained in the set of causal states, detailed in equation (1), where $< \cdots >$ means expectation.

$$I[x;y] =< \log_2 \frac{P(x,y)}{P(x)P(y)} > \quad (1)$$

Once a network system's history is obtained, the sets L^- and L^+ are extracted and $I[l^+;l^-]$ is calculated for each element, resulting in a measurement of systemic self organization. If the metric is used on sub sequences of the history, then the increase in self organization can be measured.

Observed that the metric is opposed to both random and trivial histories. If a history is random, that means any given l^- and l^+ are equally likely to occur together and $P(x,y) = P(x)P(y)$. On the other hand, if a history is trivial, then the sets L^- and L^+ are minimal, again resulting in $P(x,y) = P(x)P(y)$. As long as $P(x,y) \leq P(x)P(y)$ then equation (1) produces a zero or negative value. Thus, once the metric gives a positive reading, the history contains self organization metric value. While the metric measures the amount of self organization, this still does not specify whether the self organization is created or emergent.

4. NETWORK SECURITY DOMAIN

Network security is a very general concept. While it can refer to the standard wired network, it can also refer to wireless ad hoc networks as well as logical networks that are instantiated in software. As such, the problem domain is defined symbolically in such a way that it can be mapped to any such network. It is important to define the problem domain symbolically in this way in order to determine different general problem characteristics, such as runtime complexity or scenario dynamics.

To symbolically defined the problem domain, first the elements and their relations are identified in this section. Then the elements are mapped to formal models in order to produce complexity characteristics. The problem domain elements consisting of SOMAS agent models and Scenario "compromise" agent models are covered in Table 1. Although not shown, the network environment model consists of locations, edges, and network security scenarios [8].

Table 1: Problem domain elements

SOMAS and Scenario Agent Models
$< Sensors_{Regional}, Rules, Acutators_{Regional} >$
Sensors Real value vectors in limited range $\{(0 \ldots 1), \ldots, (0 \ldots 1)\}$ *Rules* Modus ponens $condition(sensors) \rightarrow actuator(region)$
Actuators $actuator(region) \rightarrow region' : \Box(region' = region)$

The problem domain is composed of two primary elements: agent (A) and environment ($\mathcal{E}$), which in the case of this research is a computer network (N). The objective

Table 2: Problem Domain Nomenclature

	Element	States	Action	Transition Functions
1	Agent (A)	$A_S \equiv A_P$	A_A	$R_S \times A_A \times R_S$
2	Region (R)	$R_S \equiv \{V' : hops(V, V') <= 1\} \cup \{L : connects(L, V, V')\}$ $\forall v', l \in R_S : v', l \in N_S$	N/A	N/A
3	Container (C)	$C_S \equiv \{C_P \cup A\}$	C_A	$C_S \times C_A \times C_S$
4	Link (L)	$L_S \equiv L_P$	L_A	$L_S \times L_A \times L_S$
5	Network (N)	$N_S \equiv \{N_P \cup \{L : connects(L, V, V') \wedge active(V) \wedge active(V')\} \cup \{V : active(V)\} \cup \mathbb{A}\}$	N_A	$N_S \times N_A \times N_S$
6	Scenario (S)	$S_S \equiv \{S_P \cup N\}$	S_A	$S_S \times S_A \times S_S$

Global reward function : $S_S \times \text{Time} \rightarrow [-\infty \ldots \infty]$
Local reward function : $A_O \rightarrow [-\infty \ldots \infty]$

S subscript	A state.
P subscript	A parameter.
A subscript	An action.
O subscript	An observation, which only agents have.

in this problem domain is, in general, to optimize the network communication and control. This encompasses many different network metrics, ranging from the high level metrics such as mission accomplishment, mid level metrics such as quality of service, and low level metrics such as intrusion detection.

It may be mathematically impossible to represent all the network objectives in terms of a quantifiable metric. Ultimately, it comes down to human judgement. Consequently, solutions to the network security problem should leave as much flexibility to human judgement as possible, as the SOMAS approach does by allowing any goal to be defined for SOMAS that can be qualified in terms of the elements in table 1. Thus, SOMAS is a form of intelligence augmentation.

For the purposes of this research all hardware in the network is assumed to only be controlled by software. Physically securing the network hardware is outside the scope of a computer science solution. Additionally, the container model assumption implies that only relevant software state information is accessible for reading and writing. If all locations in the network are constantly connected, then the container can be considered to encompass the whole network. However, in general, this is an invalid assumption. Thus, Cybercraft containers are segmented according to the links connecting hosts in a network.

The network graph is composed of vertices (V) and links (L). The vertices are equivalent to agent containers (C) for the purposes of this research. Table 2 further symbolically details each problem domain element, and the following list explains each symbol. Each state element is indexed with a time-stamp. Thus, the scenario state $S_S(t)$ consists of all scenario state elements with time-stamp t. To reduce notational confusion, the (t) index is only used on the global scenario state S_S. The objective for the scenario is described by the reward functions. The reward can either be specified at the global or local level. The objective can also be considered the swarm's behavior. However, behavior can also refer to the swarm's ruleset, in which case a swarm's behavior is not necessarily the same as the objective since the ruleset may not achieve the objective. There are many objectives that the swarm can achieve. The specific behaviors that are examined in this research are: Unobtrusive network activity, Self organization, DDoS prevention, Avoiding occupied nodes, Removing intruders from the network, Competition between friendly and malicious agents, and Protecting the network information flows.

5. OFFLINE AND ONLINE PROCESSES

There are two stages to the swarm's lifecycle: when it is first generated and when it is run in the environment. Consequently, there are two problem representation (model) domains and two search (algorithm) domains, respectively corresponding to the offline generation and online runtime. Each of theses processes employs a POMDP model which helps to generate individual agent policies (rules and parameter values).

5.1 Mapping Problem Domain to POMDPs

The problem domain is a planning problem [11]. All scenario state transitions only depend on the previous state, so the Markov assumption holds for this problem domain. Thus, transition functions only require the previous state, as shown in the *Transition Function* column of Table 2. As discussed in the previous section, there are two perspectives the multi-agent system can be viewed from: the global or local perspective. From the global perspective, the viewer has access to the global state and corresponding reward. However, from the local perspective, the viewer only has access to the local state. Since the local state does not necessarily embed any global information, it does not necessarily provide any global reward information. Thus, from the local perspective agents may have to compete with each other to maximize the swarm's global reward. Since the two views imply two different kinds of reward functions, two different representational models are necessary: the DEC-POMDP and I-POMDP models.

5.2 POMDP Models

A partially observable Markov decision process (POMDP) framework is modified in order for proper use in our network security model. The decentralized POMDP (DEC-POMDP) model is for multiple agents such that agents can interact with the same environment and share the same *global reward* function. On the other hand, agents need to be able to examine each other, have beliefs about each other, and have *local reward* functions. These capabilities define the interative POMDP (I-POMDP) model. Formal representations of these forms for the security application can be found in [8].

DEC-POMDP Model: The DEC-POMDP model is for multiple agents that can interact with the same environment and share the same global reward function. This model represents a global perspective of the multi agent system, since only from the global perspective can the group reward be known. The use of the model here does not exactly match that of [3] since only the elements necessary to derive the problem domain complexity are used.

$$\Psi_{DEC} :< S, \mathbf{A}, T, \mathbf{\Omega}, O, R > \quad (2)$$

A DEC-POMDP model is used for the offline "optimization" problem representation and an I-POMDP model is used for the online problem representation.

- S consists of all scenario states $P_S \in P$. This includes the state of each vertex (V) and link (L), extra

scenario info, etc. [11] is equivalent to the POMDP model. The DEC-POMDP and I-POMDP models are the most comprehensive for characterizing multi agent

- **A** is a set of sets, where each set is drawn from $\mathbb{N}_A$, the universe of possible agent actions including the null action. Each set consists of possible aggregate actions of agents in the swarm. It is defined according to formula (3), which shows every action set is the same size and every agent accomplishes exactly one action. a_i is the action of agent i. n is the number of agents in the swarm. $\{\beta\}^\alpha$ means the closure of alphabet β, where all strings in β are of length α.systems, while the first provides a global, and the second a local, perspective. The differing perspectives match the two stages in the swarm's life.

$$\forall A[A \in \mathbf{A} : A \in \{\mathbb{N}_A\}^n \wedge \forall a_i, a_j \in A\{i \neq j\}] \quad (3)$$

Both are additionally augmented to be F(*-POMDP) models in the case where agents can be added or removed from the environment. The agent schema in cycle, thus they are chosen as being the most suitable. Note that the optimization problem refers to optimizing a user selected security behavior. We formally define a number of desired behavior scenarios for testing SOMAS. Each action in $\mathbb{N}_A$ is composed of agent actuators of the form $\delta(\Phi_S, \gamma)$ where δ is a function with possibly irreversible side effects, Φ_S is a set of elements $\forall \phi_S\{\phi_S \in \Phi_S : \phi_S \in p_S\}$, and γ is a set of parameters. The scenario can have rules as well, but these are implied in the state transition and do not need to be explicit in $\mathbb{N}_A$.

- T maps state/action set pairs to states, according to formula (4).igure 1 represents the overall process of offline search generating the online search, through creating the agents' local f

$$T : S \times \mathbf{A} \rightarrow S \quad (4)$$

Fitness function. (Agents are represented by faces, the "eye-in-the-sky" implies that global information (**reward**) is used to generate agent swarm policy for a given behavior.) Again, the intent is to generate individual agent rules and parameter values (**a policy**) within the swarm t

- $\mathbf{\Omega}$ is a set of observation sets, defined similarly to **A**, detailed in formula (5).hat generates the desired behavior with this feedback structure. These polices for each behavior are to be employed then in a real-world network security

$$\forall \Omega[\Omega \in \mathbf{\Omega} : \Omega \in \{\mathbb{N}_\Omega\}^n \wedge \forall \omega_i, \omega_j \in \Omega\{i \neq j\}] \quad (5)$$

situation with a dynamic I-POMDP execution.

- O maps T element/observation set pairs to probabilities, according to formula (6). An element f_e of function f is pair $(x, y) : f(x) \rightarrow y$.

$$O : T_E \times \mathbf{\Omega} \rightarrow (0 \ldots 1) \quad (6)$$

- R is the global reward, identical for every agent since it is global, accruing a value in $\mathbb{R}$ for each element of T, detailed in formula (5.2).

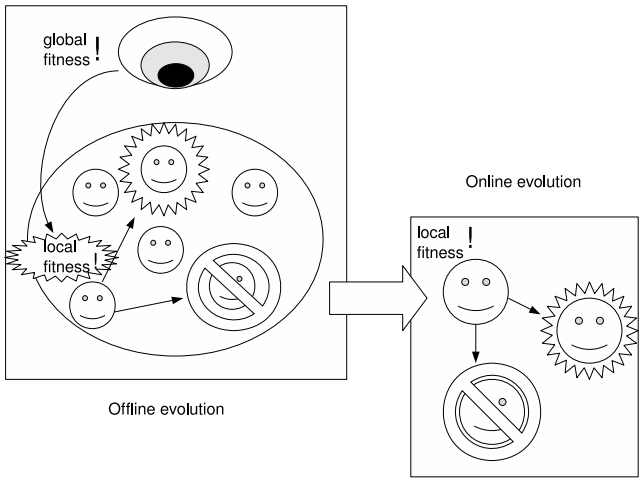

Figure 1: Process of offline to online MAS evolution

$$R : T_E \rightarrow [-\infty \ldots \infty] \quad (7)$$

As an example of how this model can be used, consider the scenario where SOMAS agents have to identify and eliminate malicious agents from a network. The environment states, observations, and actions are straightforward. A state transition in the T function is, for example, a SOMAS agent changing its location. An observation in the O function is when the SOMAS agent at a location runs an intrusion sensor on the location. The state (there is a malicious agent at the location) and the scan action have a probability that the SOMAS agent will actually identify the presence of the malicious agent. Finally, an example of an element of the R function is when the malicious agent has been located, the SOMAS agent successfully deletes the malicious agent.

The detailed I-POMDP and F(*-POMDP) structures are quite similar to those of the DEC-POMDP with appropriate modification in their development [8].

5.3 Deriving POMDP Model Policies

From the POMDP model complexity results, it is clear that even though the DEC-POMDP and I-POMDP formalizations capture the SOMAS problem domain, it is impractical at this stage of the cutting edge to generate the optimal SOMAS policies from either the DEC-POMDP or I-POMDP models. Therefore, it is logically impossible or computationally tractable to solve F(*-POMDP) models in the general case algorithmically. However, this does not rule out the possibility that the models can generally be solved mathematically, i.e. with proofs. If a policy solution is generated from the F(*-POMDP) model, it becomes a part of the model since agents can manipulate the policies of agents, so an infinite horizon policy has to be a fixed point function.

As can be observed in the case of each POMDP model, while they succinctly capture the theoretical aspects of each stage of SOMAS production, optimal or approximate policies cannot be generally derived in a tractable manner using deterministic search approaches. This means a different representation and policy generation technique is required for the solution domain for proper exploration. Moreover, as

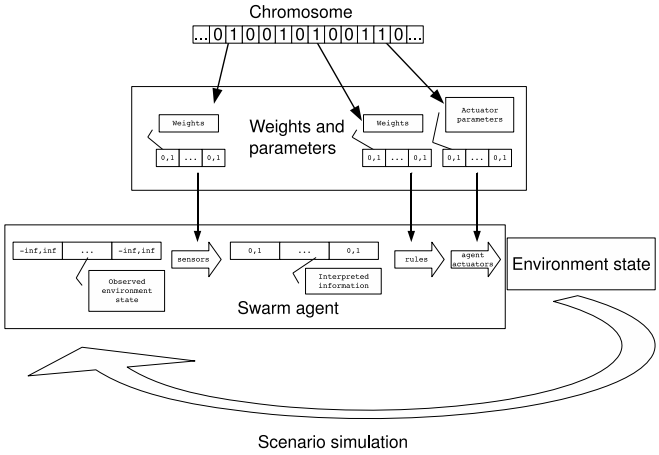

Figure 2: Translating the chromosome to the agent rules/-parameters

the number of objectives increases, it is more difficult to find an approximate Pareto front (PF_{KNOWN}) significantly close to the PF_{TRUE}. Thus, we elect a specific multiobjective evolutionary algorithm (MOEA) stochastic search technique, the Indicator Based Evolutionary Algorithm (IBEA) algorithm [19], that combines the exploitation of NSGA2 and exploration of SPEA2. Solution representation for the associated MOEA chromosome is depicted in Figure 2. The figure indicates how the binary chromosome string is converted to real values which correspond to agent rule sets and parameters [8].

5.4 Solution Domain

A solution consists of a behavior that accomplishes a certain objective. The behavior depends on the agent rulesets ($\mathcal{R}$) and the environment. A behavior from the global perspective is a state/action tuple to probability mapping function, formula (8). A behavior from the agent perspective is a rule (r) to probability mapping function, formula (9). An objective is an environment tuple set to reward mapping function, formula (10). $\wp(\Omega)$ is the powerset of Ω.

$$\mathcal{B}_g : S \times A \rightarrow (0 \dots 1) \quad (8)$$

$$\mathcal{B}_l : r \rightarrow (0 \dots 1) \quad (9)$$

$$\mathcal{R} : \wp(S) \rightarrow [-\infty \dots \infty] \quad (10)$$

The intent is to find a solution $\mathbb{S}$ given an initial definition of self organized behavior that maximizes the reward, and this can be done at either the global or local level. Thus, a solution consists of a set of rulesets, out of all possible rulesets, that maximizes the reward. A rule is sequence of operators (δ) and operands (γ). If the rules and the sequence of operators in the rules of a ruleset are fixed, then finding the solution at the local level implies that a solution is a fixed size set of operands.

6. DESIGN OF EXPERIMENTS

The design of experiments focuses on effectiveness and efficiency of SOMA. It requires a statement of test objectives, algorithm testing process with appropriate parameters, selection of presentations that should reflect the experimental results and in this case, the swarm behavior scenarios as reflections of network threats in section 2. Note that a fixed agent ruleset is chosen in order to reduce the state space complexity and based on the assumption that the required rulesets are accessible to the SOMAS creator.

6.1 Test Objectives

The primary objective is to explore the impact of self organization on generating effective swarm behaviors in a variety of security scenarios. Since all the swarms are evolved using the self organization metric, the PF_{KNOWN} of each scenario is examined to visually identify the relation between the self organization objective value and the other objective values.

6.2 Algorithm Testing

In order to determine the comparative effectiveness of different swarm types in a scenario, the PF_{known} for each is compared. The metric for this measure is Pareto compliant. If the metric is not Pareto compliant, then it may rank a dominated front higher than the dominating front. The Platform and Programming Language Interface for Search Algorithms (PISA) framework contains three Pareto compliant indicators. Pareto compliance means that the indicator does not rank a non-dominant set as dominating the dominant set. The indicators are the set of R indicators, and the ϵ and hypervolume indicators. In depth detail can be found in [6].

As explained in [10], 25 samples are required in order to produce statistically significant results, regardless of whether the samples come from a normal distribution or not. This result is due to the central limits theorem, which says that the distribution of independent samples from a given distribution tends towards the normal distribution as the number of samples approaches infinity, and the distribution of the samples should be approximately normal when the sample count is large enough. Empirically, 25 samples is shown to be "large enough." Consequently, to obtain a good statistic for each chromosome, the simulation runs 30 times for each chromosome.

The population size for each generation is 40, where 20 members come from the previous generation and 20 members are produced by the variator. The initial population consists of 40 individuals. Each algorithm is run for 10 generations to produce a total of 240 individuals, and 7200 simulation evaluations. These parameters are chosen so that a run takes at most 2 days to complete, while still providing enough results to run the statistical tests and generate dependable p-values. Note that the setting of these MOEA parameter values is based upon insight to the problem domain and experience of other researchers [6]

There are two aspects to the presentation of results. The analytic results are displayed in tables showing the p-values of the different hypotheses. Additionally, the Pareto fronts are plotted to give a visualization of how the EA explored the solution space, as well as giving an impression of how the solutions are distributed throughout the solution space.

6.3 Scenarios

Compromise agents and SOMAS agents are generally pitted against each other. The compromise agents attempt to destroy the network by compromising nodes and creating DDoS agents, which in turn mount a DDoS attack against every node on the network. Whenever a node is destroyed, all agents on the node are removed from the simulation.

The SOMAS agents can delete agents they find and also destroy or bring back up network nodes. However, the SOMAS agents do not know which agents are friendly and which agents are malicious. The SOMAS agents have these same capabilities for the rest of the scenarios.

Success is measured by counting the number of malicious agents destroyed and the number of SOMAS agents destroyed. For each of the following scenarios a description of the objective is stated. Formal mathematical descriptions can be found in [8]. These formal models are used in the DEC-POMDP and I-POMDP "optimization" offline and online processes.

DDoS Defense: In the network there is a set of target nodes that the DDoS agents attempt to destroy. The SOMAS agents use their capabilities to eliminate as many of the DDoS agents and packets as possible while also keeping all the targeted nodes alive. Success is measured by the number of DDoS agents and packets deleted, and by the number of targeted nodes alive by the end of the simulation; a maximization model. As an example of a formal description of the desired for DDOS consider that there is only a single objective for this behavior, which is to maximize the number of targeted nodes that are active at the end of the simulation.

$$\frac{1}{|C^{at}|} : C^{at} \equiv C^a \cap C^t \cap S_S(f) \tag{11}$$

C^a	The set of active containers.
C^t	The set of targeted containers.
$S_S(f)$	The scenario state at the final time step.

Intrusion Elimination: A set of compromise agents randomly travel around the network compromising nodes and creating DDoS agents. The SOMAS agents attempt to uncompromise as many nodes as possible. Success is measured by the number of nodes compromised at the end of the simulation; a maximization model.

Information Warfare: Information Warfare is the most complex scenario, and combines elements of most of the other scenarios. The network layout is composed of multiple subcomponents, instead of being flat. Within the network, there are a set of sender and receiver nodes. The sender nodes attempt to continually send information packets to the receiver nodes. There are also a number of compromise agents, who begin on nodes situated outside of the main network. They enter the network, compromised as many nodes as possible, and create as many DDoS agents as possible. Meanwhile, they also delete friendly information packets and steal the information and send it out of the network. The DDoS agents target all the receiver nodes in order to stop all friendly information traffic. Success is measured by maximizing the friendly information delivered and minimizing the enemy information delivered.

Other scenarios developed and tested include enemy avoidance, agent competition, and vital network graph optimization [8].

7. RESULTS AND ANALYSIS

Since the general objectives of self organization and criticality and online evolution effectiveness are critical to the success of our approach, they are address in terms of generic performance over the various scenarios.

7.1 Self Organization and Criticality

The use of the self organization metric leads to self organized criticality during the policy solution search. Throughout many of the scenarios, the PF_{known} plot shows a gap in structures of dominant chromosomes. Chromosome evaluation produces fairly tight variances along a path up to a certain midway point, then the variance becomes large very rapidly and certain sections along the path do not contain any values. By comparing the different objective plots, it is clear that this relationship holds between the feasibility, self organization, and bad agent deletion. One may wonder why it does not also hold between activity and good agent deletion. This is because the number of good agents is static if they are not deleted, while the number of bad agents increases if they are not deleted.

Due to the chaos created by many agents, it might be assumed that the greatest variance would be found at the path end with the most agent activity. The discrepancy is explained by the concept of self organized criticality [1]. In self organized systems, self organization reaches its highest level at a point right before chaos, where as many of the system's energy levels are filled. Thus, a disturbance in the system results in an avalanche of energy dissipation.

Since the system is so close to chaos, yet is very well ordered at the same time, there is great potential for a variety of behaviors to emerge. However, when the system is not very self organized, due to significant chaos or simple behavior, there are not many new behaviors that can develop. Consequently, the greatest range of behavior, and thus the greatest range of objective values, are found around the point of self organized criticality. Self organized criticality can even lead to new realms of the solution space that would not otherwise have been reached.

7.2 Effectiveness of online evolution

The statistical tests in Table 4 demonstrate that as the scenario becomes more complex, swarms capable of more complex and influential entangled hierarchies become more effective than less capable swarms. As described in Table 3 the scenarios are categorized according to the complexity of the network layout, and whether the environments and agents are static/dynamic. The results show that in almost all network scenarios, swarms with online search (EV and NE swarms) decisively outperform swarms that lack online search. Not only are online search swarms more capable than non searching swarms in more complex scenarios, the searching swarms are more capable almost across all scenarios. The PF_{known} plot in figure 3, implies indirectly that online evolution chromosomes have a much more exploratory path than the chromosomes without online evolution. Thus, the online evolution chromosomes are able to acquire a greater number of building blocks and construct better solutions [8]. While the use of online search explains why the EV and NE swarms do better than the NS swarm, it does not explain why the EV swarm outperforms the NE swarm in only certain scenarios. Contrary to expectations, the increase in the EV swarm's effectiveness does not strictly follow the increase in scenario complexity, though such a trend is evident.

The distinguishing feature is that the scenarios where the EV swarm is more effective require a greater specificity of behavior than the scenarios where the NE swarm is more effective. For instance, even though the complexity of the Competition scenario is greater than the complexity of the

Behavior Name	Scenario Name	Topology Complexity	Topology Dynamics	Scenario Agent Dynamics
Minimal activity	all	n/a	n/a	n/a
Self organization	all	n/a	n/a	n/a
Preserve network	all	n/a	n/a	n/a
Intrusion elimination	Intrusion elimination	Homogeneous	Static	Dynamic
Network defense	DDoS	Homogeneous	Dynamic	Dynamic
Competition	Competition	Homogeneous	Dynamic	Dynamic
Information defense	Information war	Heterogeneous	Dynamic	Dynamic

Table 3: Behavior experiment summary

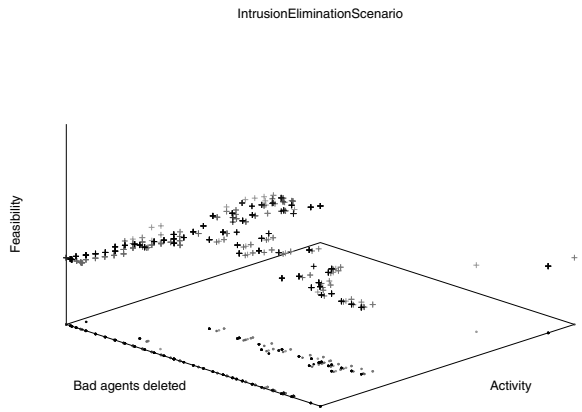

Figure 3: Pareto front evolution with online evolution

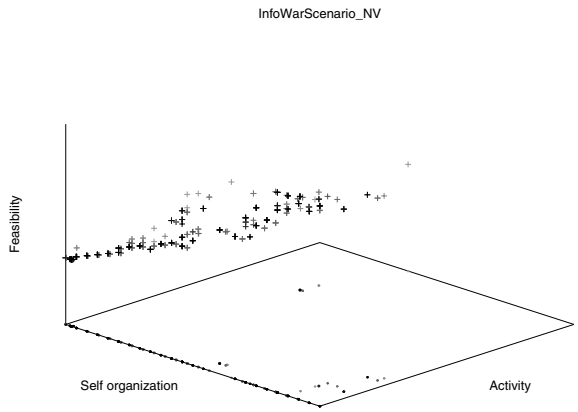

Figure 4: Pareto front evolution with network destruction

DDoS scenario, the actual network and agent activity in the Competition scenario is much more homogeneous. In the Competition scenario it isn't important to perform specific actions in specific portions of the network, since the elimination of enemy agents is worth the same amount regardless of network location. Thus, the same swarm behavior is generally effective regardless of swarm location.

In the DDoS scenario the swarm has to protect specific targets, so the swarm needs to use different behaviors for different parts of the network. Thus, the swarm in this scenario needs to be capable of greater specificity in its behavior. The same pattern can be seen with the Information War scenario. When the swarm has node deactivation and activation actuators, the EV swarm is more effective.

Online evolution still outperforms the alternatives according to certain indicators and tests, but other combinations give the alternative swarms better results. The PF_{known} plot in figure 4, indirectly indicates that the evolution of the destructive chromosomes is less exploratory than when the non destructive swarms are evolved [8]. As with the success of online evolution, greater exploration means a greater range of building blocks for constructing solutions, and so a lack of exploration implies the chromosomes becomes stuck on local optima. Additionally, network destruction makes the environment more chaotic, since not only is the agent population at each node changing, but the network topology is also changing. With so many more degrees of freedom, the good building blocks become much more localized and harder to find.

8. CONCLUSIONS

The formal multi-agent model of a network security system has lead through the use of various POMDP models to a formal architectural design of the Self Organized Multi-Agent System (SOMAS). SOMAS provides a model for defining desired network security behaviors and implementing and testing them for various network attack scenarios. The experiments indicate that the dynamic SOMAS architecture has creditability as a vehicle for developing a network security entity. Although a small number of behavior policies are explored via SOMAS, the statistical results and visualizations show that a multi-agent swarm can protect a computer network. The use of POMDP models and MOEAs in the architectural design is critical to generation of stochastic agent policies (rules and parameter values). The experiments shows that self organization plays an interesting role in the search process through self organized criticality as represented in Pareto fronts. The self organized entangled hierarchical SOMAS architecture dynamically decentralizes network security control and communication providing not only efficient operations, but effective as well. Since online evolution is adaptation through the use of entangled hierarchies, the success of these multi-agent swarms is a good indicator that entangled hierarchies provide the needed flexibility for network security management. Potential research projects range from the theoretical to particular applications from adding more potential agent operators to using a wider variety of swarm polices in a real-world network.

This investigation is a research effort of the AFIT Cyberspace Technical Center of Excellence, Director: Dr. Rick Raines.

Behavior	R_2			Epsilon			Hypervolume		
Intrusion elimination	Better	Worse	P-value	Better	Worse	P-value	Better	Worse	P-value
	2	1	***0.004***	2	1	***0.004***	2	1	***0.008***
	3	1	0.995	3	1	0.995	3	1	0.995
	1	2	0.995	1	2	0.995	1	2	0.991
	3	2	0.995	3	2	0.995	3	2	0.995
	1	3	***0.004***	1	3	***0.004***	1	3	***0.004***
	2	3	***0.004***	2	3	***0.004***	2	3	***0.004***
Network defense	Better	Worse	P-value	Better	Worse	P-value	Better	Worse	P-value
	2	1	0.458	2	1	0.622	2	1	0.826
	3	1	0.995	3	1	0.995	3	1	0.995
	1	2	0.541	1	2	0.377	1	2	0.173
	3	2	0.995	3	2	0.995	3	2	0.995
	1	3	***0.004***	1	3	***0.004***	1	3	***0.004***
	2	3	***0.004***	2	3	***0.004***	2	3	***0.004***
Competition	Better	Worse	P-value	Better	Worse	P-value	Better	Worse	P-value
	2	1	0.300	2	1	0.623	2	1	0.458
	3	1	0.173	3	1	0.929	3	1	0.699
	1	2	0.699	1	2	0.376	1	2	0.541
	3	2	0.300	3	2	0.913	3	2	0.826
	1	3	0.826	1	3	**0.070**	1	3	0.300
	2	3	0.699	2	3	**0.086**	2	3	0.173
Information defense	Better	Worse	P-value	Better	Worse	P-value	Better	Worse	P-value
	2	1	0.173	2	1	0.962	2	1	0.941
	3	1	0.962	3	1	0.995	3	1	0.985
	1	2	0.826	1	2	***0.037***	1	2	**0.058**
	3	2	0.962	3	2	0.976	3	2	0.976
	1	3	***0.037***	1	3	***0.004***	1	3	***0.014***
	2	3	***0.037***	2	3	***0.023***	2	3	***0.023***

Table 4: Hypotheses results

9. REFERENCES

[1] P. Bak, C. Tang, and K. Wiesenfeld. Self-organised criticality. *The American Physical Society*, 1988.

[2] G. Baldassarre, S. Nol, and D. Parisi. Evolving mobile robots able to display collective behaviors. In *Artificial Life 9*. Massachusetts Institute of Technology, 2003.

[3] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[4] J. Branke, M. Mnif, C. Miller-Schloer, H. Prothmann, U. Richter, F. Rochner, and H. Schmeck. Organic computing : Addressing complexity by controlled self-organization. 2007.

[5] S. Camazine, J. Deneubourg, N. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton University Press, 2003.

[6] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2 edition, 2007.

[7] N. Foukia. Idream: Intrusion detection and response executed with agent mobility the conceptual model based on self- organizing natural systems. *ESOA*, 2004.

[8] E. Holloway. Self organized multi agent swarms (SOMAS) for network security control. Master's thesis, Air Force Institute of Technology, March 2009.

[9] J. Kim and P. J. Bentley. Towards an artificial immune system for network intrusion detection: An investigation of clonal selection with a negative selection operator. Technical report, University College London, 2001.

[10] J. S. Milton and J. C. Arnold. *Introduction to Probability and Statistics*. McGraw Hill, 2003.

[11] P. Norwig and S. Russell. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[12] T. Schlegel, P. Braun, and R. Kowalczyk. Towards autonomous mobile agents with emergent migration behaviour. In *ACM 5th international joint conference on Autonomous agents and multiagent systems (AAMAS'06)*, pages 585–592, New York, 2006.

[13] G. D. M. Serugendo, M.-P. Gleizes, and A. Karageorgos. Self-organization in multi-agent systems. In *The Knowledge Engineering Review*, pages 165–189. Cambridge University Press, 2005.

[14] G. D. M. Serugendo, M.-P. Gleizes, and A. Karageorgos. Self organization and emergence in mas: An overview. In *Informatica*. The Slovene Society Informatika, 2006.

[15] D. Servat and A. Drogoul. Combining amorphous computing and reactive agent-based systems: a paradigm for pervasive intelligence? In *AAMAS'02*, pages 441–448. ACM, 2002.

[16] C. R. Shalizi, R. Haslinger, J.-B. Rouquier, K. L. Klinkner, and C. Moore. Automatic filters for the detection of coherent structure in spatiotemporal systems. *Physical Review E*, 73, July 2005.

[17] M. R. Stytz, D. E. Lichtblau, and S. B. Banks. Toward using intelligent agents to detect, assess, and counter cyberattacks in a network-centric environment. Technical report, Institute for Defense Analysis, 2005.

[18] D. Yamins. The emergence of global properties from local interactions: static properties and one-dimensional patterns. In *AAMAS'06*, pages 1122–1124. ACM, May 2006.

[19] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In X. Yao et al., editors, *Parallel Problem Solving from Nature (PPSN VIII)*, pages 832–842, Berlin, Germany, 2004. Springer-Verlag.