A Case Study in Tailoring a Bio-Inspired Cyber-Security Algorithm: Designing Anomaly Detection for Multilayer Networks

Gonzalo P. Suárez[1,2,*], Lazaros K. Gallos[1] and Nina H. Fefferman[2,3]

[1]*Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), Rutgers University, Piscataway, NJ, USA*
[2]*Department of Ecology and Evolutionary Biology, College of Arts & Sciences, University of Tennessee, Knoxville, TN, USA*
[3]*Department of Mathematics, College of Arts & Sciences, University of Tennessee, Knoxville, TN, USA*
E-mail: gsuarez1@utk.edu; lgallos@dimacs.rutgers.edu; nina.h.fefferman@gmail.com
[*]*Corresponding Author*

Received 29 September 2018; Accepted 01 October 2018;
Publication 01 November 2018

Abstract

Although bio-inspired designs for cybersecurity have yielded many elegant solutions to challenging problems, the vast majority of these efforts have been ad hoc analogies between the natural and human-designed systems. We propose to improve on the current approach of searching through the vast diversity of existing natural algorithms for one most closely resembling each new cybersecurity challenge, and then trying to replicate it in a designed cyber setting. Instead, we suggest that researchers should follow a protocol of functional abstraction, considering which features of the natural algorithm provide the efficiency/effectiveness in the real world, and then use those abstracted features as design components to build purposeful, tailored (perhaps even optimized) solutions. Here, we demonstrate how this can work by considering a case study employing this method. We design an extension of an existing (and ad hoc-created) algorithm, DIAMoND, for application beyond its originally intended solution space (detection of Distributed Denial

Journal of Cyber Security and Mobility, Vol. 8_1, 113–132. *River Publishers*
doi: 10.13052/jcsm2245-1439.815

of Service attacks in simple networks) to function on multilayer networks. We show how this protocol provides insights that might be harder or take longer to discover by direct analogy-building alone; in this case, we see that differential weighting of shared information by the providing network layer, and dynamic individual thresholds for independent analysis are likely to be effective.

Keywords: Cyber-Security, Bio-Inspired, Anomaly detection, Multilayer Networks.

1 Introduction

Motivation. Bio-Inspired algorithm development in cybersecurity frequently relies on isolated analogy building between particular threats and specific biological systems [1]. While meticulous care is often taken in building these analogies and then tailoring the borrowed algorithms to meet the needs of cyber systems (e.g. [2, 3]), straying too far from the initial natural setting can compromise the function of the solution, rendering the act of turning to nature for inspiration frustrating and even futile. It is therefore potentially hugely desirable to consider how one might consider, tailor, and test abstracted features of biological algorithms outside of their native context, without having to carry over the specific details of the biological system that shaped their use. Naturally, this runs the risk of eliminating the very scope of the system that made them appropriate and useful in the first place.

Balancing this desire for conceptual abstraction against the specificity of the iterative trial-and-error refinement that builds the natural world itself is an important step in discovering the building blocks nature uses to create robust structures; if we can learn the principles of architecture, perhaps we can design buildings rather than simply altering and implementing existing blueprints. To that end, we here describe and analyze a process of exploration in which we consider a single bio-inspired algorithm, focused on anomaly detection in networks, and attempt to abstract its function without losing its form for purposes of being able to describe its scope and performance. (Note: this description may even be possible analytically, rather than experimentally, which would allow us to bring to bear the vast toolkit available from the world of mathematical optimization). This desired shift to focus on functional features improves our ability to leverage the inspiration of biology without being restricted to efficient function only in direct analogy to the contexts of biological systems themselves.

Case Study in Anomaly Detection. One class of malicious Internet attack of broad concern involves the saturation of the network by purposefully

generated (and otherwise useless) traffic (e.g. DoS and DDoS attacks). These attacks can severely influence efficient handling of packets by any node that lies along the the path of the attack. The early detection of abnormal traffic volume or patterns is a central problem in cybersecurity since efficient methods allow more time to employ mitigation strategies (e.g. packet refusal, neighborhood isolation, etc.; [4, 5]). The past few decades of work have yielded a wealth of anomaly detection methods [6]. These methods can broadly be classified as either centralized or distributed. In centralized methods, a localized monitoring agent (or set of agents) collects and analyzes data monitoring traffic from the entire domain/network, and then employs some statistical or probabilistic metric of comparison against expectation to decide whether anomalies are present and if these should be attributed to malicious attacks. In contrast to that, distributed algorithms allow smaller network neighborhoods (including individual nodes themselves) to decide on the status of the network as it is perceived through data gathered about routed packets handled within the small neighborhood itself, without the need to monitor the behavior of traffic in other parts of the network. Distributed anomaly detection methods have been gaining traction lately as a means of mitigating attacks because they incur lesser costs in communications overhead, invoke no delay from the collation and analysis of large, streaming data from multiple (potentially asynchronous) sources of varying reliability, and are simply easier to implement and deploy [7].

Recent work developed a non-parametric, distributed, bio-inspired algorithm for anomaly detection in networks utilizing principles of self-organization observed in colonies of honey bees [8]. Mirroring the activities and communication that honey bee colonies use to collaborate efficiently to find and exploit external resources [9], this algorithm focuses on information from the local network neighborhood while also preserving privacy considerations, such as not sharing information about patterns in traffic handled by a specific node/domain. This DIAMoND method (Distributed Intrusion/Anomaly Monitoring for Nonparametric Detection) has been shown to be fast and efficient, and remains efficient under a quite broad range of traffic and attack conditions [7, 8, 10]. However, as with bio-inspired distributed methods, DIAMoND was constructed via ad-hoc analogy to the inspiring system of communication in foraging social insects. The complexity of the problem makes it necessary to implement many steps according to predefined decisions and use many parameters. In other words, current design and implementation are not based on any first-principles approach and it is not clear what their full capabilities or limitations are. It is clear that abstraction

of these methods to analytically tractable models or at least understanding the impact of the main model features would meaningfully improve our ability to optimize algorithms such as DIAMoND which have been inspired by nature, but are intended for use within very specific cybersecurity design cases.

As a starting point towards this abstraction, we focus in this work on the question of which features can be abstracted and which critical features are needed to retain the desired behaviors. We consider what kinds of simplifications and/or abstractions might be made to allow generalizable conclusions and suggest methods for purposeful tailoring of these algorithms to achieve goals that are further away from their bio-analogue systems. Thus using the bioinspired DIAMoND algorithm as our case study, we use this method of abstract interrogation to consider how the same inspiration might enable alternative implementations for traffic in a two-layer network system.

Modern societies are built on networks of interconnected and interdependent systems [11]. As initially isolated infrastructures become increasingly interconnected with each other, there is growing interest on the effect of interdependency between/among interacting networks (such as the Internet and the power-grid). Here we consider a scenario in which there are simultaneous attacks on a system comprised of two layers, which results in increased traffic in both layers. Nodes can use information on possible attacks through their connections to nodes of the other layer. The extension of anomaly detection methods to multi-layered systems is not straightforward and, to the best of our knowledge, there is no published method for determining optimal ways to achieve this. For this reason, we abstract the fundamental principle of distributed anomaly detection developed in DIAMoND: that nodes should leverage shared non-parametric information from other nodes to affect their own determination of anomaly detection. We thus consider different methods for utilizing those shared data, received from the other layer, focusing not on replicating the type of information-sharing observed in nature, but rather in replicating the method itself, by building the specifics of how to employ that method as purposefully engineered design.

2 DIAMoND on Multi-layer Networks

DIAMoND provides a non-parametric distributed coordination framework that decouples local intrusion detection functions from network-wide coordination. The aim is to provide a scheme where each node can detect traffic anomalies, using only the information that is available to it and the perceived state of its neighbors, without having to be informed of any parametric data

about either "normal" or current patterns in each of the neighbor's local traffic. The DIAMoND algorithm has been shown to be successful in both empirical and simulation implementations [8, 10].

In the DIAMoND algorithm (Figure 1(a)), each node keeps track of two sensitivity thresholds and uses them to analyze the total traffic that it is handling. This determines its current 'threat level' $T_i(t)$, which is a private parameter, i.e. its value is not shared with other nodes. The value of $T_i(t) = 0, 1$, or 2, summarizes what the node i believes about its status at time t: 'normal', 'concerned', or 'attacked'. Assuming that the normal traffic of node i corresponds to a normal distribution with mean μ_i and a variance σ_i, the lower threshold is chosen as $S_{iL} = \mu_i + 1.5\sigma_i$ and the upper threshold as $S_{i0} = \mu_i + 3\sigma_i$. The lower limit is fixed, but the upper limit, $S_{i,t}$, changes according to the threat level, $T_i(t)$. If the node considers itself under attack, then it lowers its upper threshold, but if the anomaly ceases, it will slowly restore the upper threshold to its initial level. To determine the extent to which the threshold should change, a node keeps a second private parameter, $L_i(t) = 0, 1, 2$, which summarizes its average level of concern such that higher values of $L_i(t)$ result in larger changes to the threshold. The average level of concern, in turn, is based on the concern level $c_i(t)$ of the node and the concern level $c_j(t)$ of all the node's neighbors. This concern level is considered public information and nodes freely share it with their neighbors. The value of $L_i(t)$ is based on the average over the k neighbors of node i, $\langle c_i \rangle = \sum_j c_j(t-1)/k$, so that $L_i(t) = 0$ when $\langle c_i \rangle < 0.4$, $L_i(t) = 1$ when $0.4 < \langle c_i \rangle < 1.3$, and $L_i(t) = 2$ when $\langle c_i \rangle > 1.3$. A complete description of

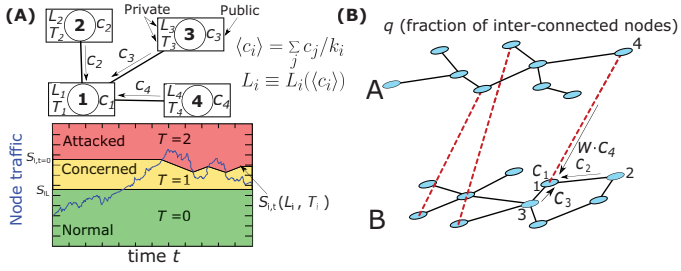

Figure 1 (A) Schematic description of the DIAMoND algorithm. Nodes share their concern level, c_i, which is then used to calculate their confidence that there is an attack taking place, as expressed through their threat level, T_i. The value of T_i is then used to modify the sensitivity threshold, according to the current traffic handled by the node. (B) In the two layer system, a fraction q of nodes are connected to the other layer. These nodes can exchange information and a node assigns a weight w on this inter-layer information.

the DIAMoND algorithm and how these parameters are related is discussed in [8, 10].

Baseline Implementation of DIAMoND in a 2-Layer Network. Before we can consider different abstracted cases, first we must implement the original algorithm itself in a multilayer context. We therefore begin by reporting results of the original DIAMoND implementation simulated in the simplest multilayer system possible: two inter-connected networks (Figure 1(b)). To determine the effect of topology each layer can be either an Erdos-Renyi (ER) network with average degree $\langle k \rangle = 3$ or a scale-free (SF) network where the degree distribution scales as a power-law $P(k) \sim k^{-\gamma}$. We use two scale-free networks, with $\gamma = 2.5$ where hubs are very strong and with $\gamma = 3.5$ where there are almost no hubs. In each layer, we consider that a fraction q of its $N \sim 5000$ nodes is connected to nodes in the other network. Here, we fix $q = 0.8$. We assume that traffic flows independently through each network and there are never any packets crossing between networks. The inter-layer links are used exclusively for sharing information, while both information and traffic pass through the intra-layer links.

As in previous simulations, we assume that at every time step each node *i* handles normal (i.e. standard) traffic which follows a gaussian distribution, with mean traffic μ_i and a variance σ_i^2. The mean value is typically around $\mu = 1000$ in arbitrary units. A fraction *p* of the nodes become compromised and start emitting "malicious" packets, each one of size *z* in the same arbitrary units as μ via a biased random walk towards a pre-assigned target node. The intensity of the attack is determined by the fraction, *p*, and the size z of the malicious packet. The parameters *p* and z may assume different values in each layer, resulting in attacks of different intensity in different layers. To evaluate the efficiency of the detection algorithm we calculate the detection accuracy in each layer, averaged over all nodes in the layer. We count how many of the nodes that reported an attack were actually under attack (true positives) and how many of them were not (false positives). In the same manner, we define true negative and false negative results. We compare the accuracy of this naive expansion of the algorithm into a multi-layer system to the accuracy achieved when each layer is independent and cannot use information from the other network. In this way, we can determine if there is any improvement in using information from both layers and can now begin to consider the conditions under which this may happen and how such implementations might be improved beyond the naive case. Further, by relaxing definitions of distance away from physical network layers, we can begin

to interpret multilayer algorithms for use on networks with different types of distance.

3 Models of Abstraction

To begin to consider how extension to multilayer networks might be best achieved, we first isolate the component features of the DIAMoND algorithm that made it effective both within its inspiring biological context, and in its original conceptualization for simple networks. As reflected in the computational parameters of the model, at its most fundamental, DIAMoND involves 3 features used by each participating node: (1) Information from direct observation of traffic handled by oneself, (2) Information from indirect sharing of information from network neighbors, and (3) Rules for how to interpret direct observations based on all information obtained.

The act of characterizing these fundamental, abstract features of DIAMoND, not as features that simply existed because they reflected the biological system that inspired them, but as features that enable the function of all such systems of this design instantly suggested directions for how to extend the same functional features to settings (such as multilayer networks) for which the direct biological analogy might have been stretched too thin to provide concrete inspiration. Furthermore, this characterization of abstract features also instantly suggested a set of initial experiments that would test the relative importance/impact of alterations in each of these features as they might be applied in the context of multilayer systems.

3.1 Model 1: Merged Attack Information

The first natural question suggested by our abstraction is to explore whether adding additional network layers is meaningfully different at all. We therefore considered a straightforward extension of the single-layer case, where the two layers are treated as one system and each node applies DIAMoND based on all available information, even though each layer may experience an attack of different intensity. This means that nodes can exchange information on whether an attack takes place but since they use data from both layers they are not able to distinguish if the attack takes place on any given layer. The only information that a node can use is whether its neighbor belongs to the same layer or not. As a result, each node averages the concern level of all its neighbors, independently of the neighboring node's layer. These values represent an averaged estimation on the status of the system as a whole,

i.e. there is no distinction between layers. The main difference from the one-layer system is in the calculation of the average concern level, which is now weighted depending on whether a neighbor node j is in the same layer or not. For this, we use a trust parameter, w, so that the average concern level now becomes $\langle c_i \rangle = \sum_j w_{ij} c_j(t-1)/k$, where $w_{ij} = 1$ for neighbors in the same layer and $w_{ij} = w$ for neighbors in different layers. In this way, a node can assign higher or lower weight to the information received from the other layer. As we increase w (Figure 2), the accuracy of the method increases slowly or remains similar to the accuracy for the isolated network in the ER layer when the attacks in the two layers are of relatively low intensity, i.e. small values of z and p. When the attack intensities differ significantly from each other, the accuracy in the SF layer drops significantly.

To study the effect of different attack intensities, we systematically changed the attack intensities from $z = 0$ to $z = 100$ in each layer. In Figure 3 we show changes in accuracy when one layer is an ER network and the other layer is a SF network with degree exponent $\gamma = 2.5$ (left panels) or $\gamma = 3.5$ (right panels). The detection accuracy in the ER layer improves by as much as 2%, compared to the already high accuracy of DIAMoND in an isolated ER layer, when the attack in the other network is of relatively high intensity, because it raises stronger attack awareness in the ER layer. For weak attacks in the other layer, accuracy remains the same or actually worsens, as the other layer tends to be less aware of the attack. At the same time the accuracy in the SF layer drops significantly, almost independently of the type of attack and remains similar to the isolated detection only when the attack in the SF layer is

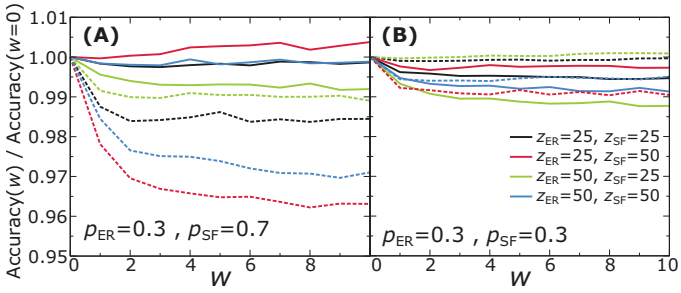

Figure 2 [MODEL 1] Ratio of accuracy in each layer of a two-layer system (ER connected to scale-free, $\gamma = 2.5$) over the accuracy in the isolated network, as a function of the trust parameter, w. Solid lines represent accuracy in the ER layer and dashed lines in the scale-free layer. Colors correspond to different attack packet sizes, z, as shown in the plot. (A) Different fraction of compromised nodes in each layer: $p_{ER} = 0.3$ and $p_{SF} = 0.7$. (B) Same fraction of compromised nodes in both layers: $p_{ER} = 0.3$ and $p_{SF} = 0.3$.

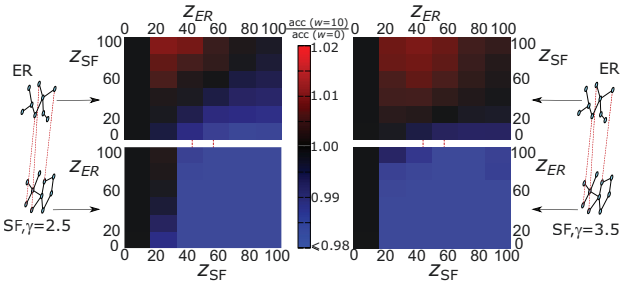

Figure 3 [MODEL 1] Ratio of accuracy in each layer of a 2-layer system over accuracy in the isolated layer, for different values of attack size, z, in each layer. Top-left: accuracy in the ER layer of the ER-SF ($\gamma = 2.5$) system. Bottom-left: accuracy in the SF layer of the ER-SF ($\gamma = 2.5$) system. The right panels show the corresponding results when the scale-free layer has a degree exponent $\gamma = 3.5$.

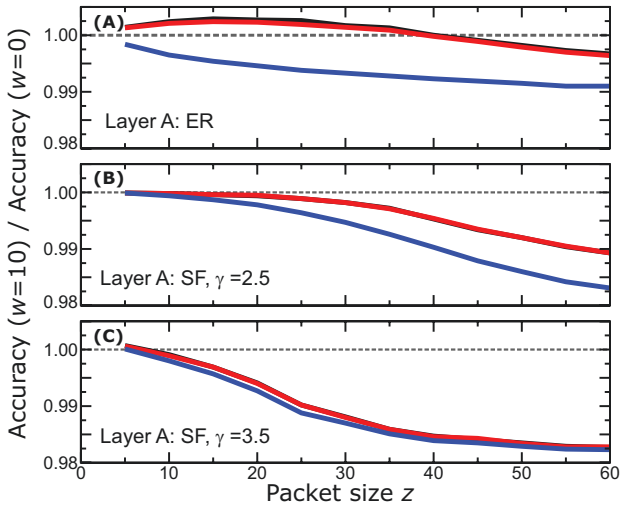

Figure 4 [MODEL 2] (A) Change in accuracy of the ER-layer when it is connected to either an ER layer (black line), a SF layer with $\gamma = 2.5$ (red), or a SF layer with $\gamma = 3.5$ (blue), as a function of the packet size z. In all cases, $p = 0.3$. (B) Change in accuracy when a SF layer with $\gamma = 2.5$ is connected to each of the other three layers described above. (C) Same information as above for accuracy in a SF layer with $\gamma = 3.5$. Notice that in almost all plots, the black and red lines coincide.

relatively weak. In terms of the topology influence, DIAMoND works better when the second layer does not include strong hubs, such as when $\gamma = 3.5$, because hubs tend to stabilize the opinion of most network nodes and it is difficult to increase their concern level.

We find, therefore, that even as detection in one layer improves, the accuracy in the second layer may diminish. As expected, this model does not take full advantage of information provided by the other layer because it merges the two layers into one broader entity and cannot separate attacks of different size. Therefore, our main result here is that DIAMoND-type algorithms do require distinct tailoring for multilayer implementation. Each node should, at least, be able to treat attacks in each layer separately. This therefore suggested the next logical feature experiment: a coupled parallel version of DIAMoND in which we assign an independent set of parameters for each layer to each node.

3.2 Model 2: Nodes Use Parallel DIAMoND Calculations for Each Layer

In the previous model, the concern level of a node was based on mixed information from both layers. When a node can identify whether the received information refers to its own layer or the other layer, it is reasonable that the node should maintain a different personal opinion on the status of each layer, such as when it feels certain that there is an attack in the other layer but not on its own, or vice versa. To keep these opinions separate each node keeps two sets of parameters: $\{c_i^{in}, L_i^{in}, T_i^{in}\}$ for information on its own layer and $\{c_i^{out}, L_i^{out}, T_i^{out}\}$ for information on the other layer. In this model, the node runs two parallel DIAMoND algorithms for each set and calculates the corresponding values. The node only acts based on information of its own layer (the *in* set) to change its sensitivity threshold, and maintains the *out* set in order to inform its neighbors in both layers about its opinion on possible attacks in the other layer. The calculation for the average concern levels of a node *i* is as follows (*j* runs over same-layer neighbors, and *m* runs over other-layer neighbors):

$$\langle c_i \rangle^{in} = \frac{\sum_j c_j^{in} + \sum_m w c_m^{out}}{\sum_j 1 + \sum_m w} \tag{1}$$

$$\langle c_i \rangle^{out} = \frac{\sum_j c_j^{out} + \sum_m w c_m^{in}}{\sum_j 1 + \sum_m w} \tag{2}$$

All concern levels are initialized to 0. In order to assess the performance of this model, in Figure 4 we show changes in accuracy of model 2 for layers in different combinations, expressed as a percentage of the performance of the DIAMoND algorithm if the layer was isolated.

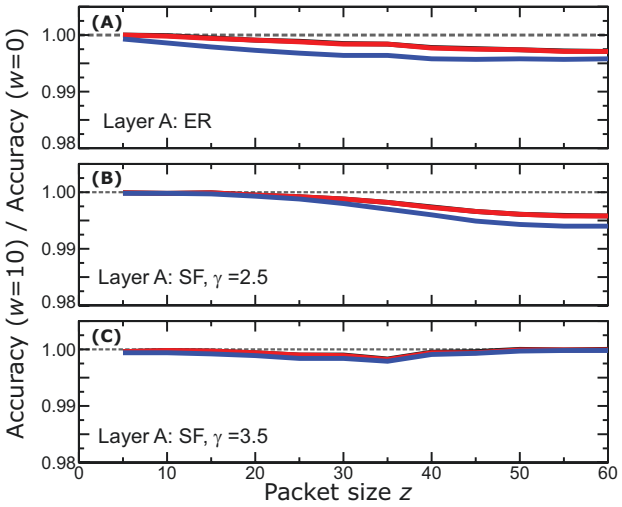

Figure 5 [MODEL 3] The change in accuracy for the same conditions described in Figure 3, where now nodes use Model 3 as their detection algorithm.

These results clearly show that the intensity of the attack is important and practically all layers improve or maintain their detection accuracy for low intensities. As the z value increases, the accuracy drops in all cases, however, the extent of the effect is influenced by the topology of each layer.

By focusing explicitly on the impact of expanding to a multilayer context for the 2nd feature, we find that explicit separation of information per layer does not improve the overall accuracy, but it does allow for different layers to independently improve their effectiveness.

3.3 Model 3: Nodes use Weighted Parallel DIAMoND Calculations for Each Layer

This case corresponds to the possibility that a node might be able to exploit greater confidence about the status of its own layer. Here, we allow a node to give different weights to the information received depending on where it comes from and what layer it refers to. We implement this distinction by modifying the calculation of the average concern level as follows:

$$\langle c_i \rangle^{in} = \frac{\sum_j c_j^{in} + \sum_m w_{in} c_m^{out}}{\sum_j 1 + \sum_m w_{in}} \tag{3}$$

$$\langle c_i \rangle^{out} = \frac{\sum_j c_j^{out} + \sum_m w_{out} c_m^{in}}{\sum_j 1 + \sum_m w_{out}} \tag{4}$$

We typically use $w_{in} = 1$ and $w_{out} = w$. This means that to assess the status of its own layer, a node gives equal weight to all its neighbors, independently of their layer. To calculate the concern level for the other layer, though, the node gives higher (or lower) weight to information received from nodes in the other layer. Again, all concern levels are initialized to 0.

In Figure 5 we see that all layers can detect attacks with very little change in their accuracy under almost any conditions, where the maximum accuracy loss is of the order of 0.5%. This shows that relating the information directly with its source improves the performance of DIAMoND, but only when this information is additionally weighted according to which layer contributes this information. In contrast, we observed lower accuracy of as much as almost *2%* in Model 2 where we do not distinguish between trust in same-layer or other-layer.

3.4 Model 4: Effect of the Default Concern Level

Here, we expand on the realization that there may be ambiguity in our 3^{rd} feature and that this ambiguity may influence our ability to understand our 2^{nd} feature: a node may interpret a large set of available information (from features 1 and/or 2) and come to conclusion of 'no concern', or instead, there may be no information to interpret, which may also lead to a conclusion of 'no concern'. To determine whether or not this potential ambiguity may impact the performance of our algorithm, we explore an alternative scenario compared to all models above which initialize their concern level to 0. This is, of course, reasonable since at the default state without attacks there is no reason to increase this value. Especially when dealing with two-layer systems, in which there is no potential for direct observation of information (i.e. feature 1), we need to distinguish whether the information of my same-layer neighbors is based on data received from the other layer or if they just report their default value, which carries no real content. The same-layer neighbors dominate the calculation of the average and even if a neighbor from the other layer attempts to inform about an attack, the same-layer neighbors may force the average to stay at the no-attack state, just by being in their default state. Therefore, we introduce a new value for the concern level, $c_i^{out} = -1$, which indicates that the node has never received information about the other layer from any neighbor. These nodes are ignored in the calculation of the averages above, which run only over nodes which have c^{out} values different than -1. Once a node gets a c^{out} value different than -1, it can never get back to -1.

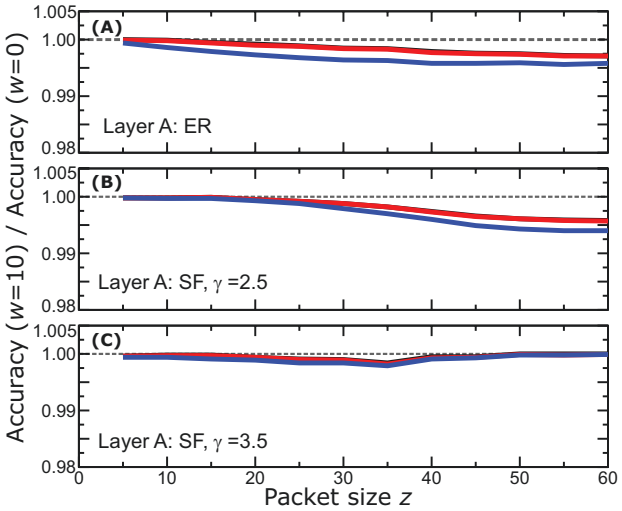

Figure 6 [MODEL 4] The change in accuracy for the same conditions described in Figure 3, where now nodes use Model 4 as their detection algorithm.

We can apply this idea of ignoring the nodes with no new information about the other layer to both of the earlier models as well. In Figure 6 we show the results that correspond to the conditions of Model 4, where we now initialize all concern levels to a default state of -1. The plots in this model are practically the same as in Model 3, and we have found a plot very similar to Figure 4, when simulated under the conditions of Model 2 (not shown here). Critically, this means the initial state of the system is irrelevant to the behavior of the algorithm because the nodes continuously update their status according to current information, and completely forget their initial status. This insight vastly simplifies the computational burden of design testing.

Again, our abstracted feature set allows us to tailor our desired algorithmic behavior based on the component principles that allow the original, biological algorithm to succeed, rather than attempting to understand some analogous version of weighted, multi-layer signal interpretation when the inspiring honey bee system may not provide a truly multi-layer template for exploitation.

3.5 Model 5: Effect of Initial Thresholds for Anomaly Definition

While the first four Models have explored the modification of every aspect of the system related to the connections between layers, and how the information is shared among neighboring nodes, they do not consider modifications of

the individual-layer process of DIAMoND itself. We therefore also consider the case in which appropriate tailoring for multi-layer function relies on the tailoring of the single-layer process.

Here, we investigate the effect of varying the set of parameters that individual nodes participating in the DIAMoND process use to determine whether their internal analysis of the patterns of network traffic constitute an anomaly. As a baseline simulation of packet traffic, we assumed that the total traffic for each node is a sum of a "normal," baseline amount of traffic, and the "extra" traffic that is formed by the malicious packets generated by the attack. We assume that each node has a different normal traffic, but all of them experience the same fundamental distribution. In each of the scenarios explored in the models above, we defined the baseline traffic to be normally distributed. In other words, the node i would have normally distributed baseline traffic with mean value μ_i and variance σ_i. The algorithm employs threshold values characterized by the distance from the mean: a value for the lower limit $S_{iL} = \mu_i + r_L\sigma_i$ and a value for the upper limit $S_{i0} = \mu_i + r_U\sigma_i$, for each node i. In each of our models explored above, we set $r_L = 1.5$ and $r_U = 3$. While the lower limit, r_L is constant for the whole process, the upper limit, r_U can change, as explained in Section 2, but it can never drop below the original value S_{iL}. For this reason, the accuracy of the algorithm's performance is tightly connected to these two values.

In abstract consideration of the role of these parameters in the algorithm, the values chosen for r_L and r_U should reflect both the distribution of baseline traffic expected by the node, and the average connectivity of the node in the network topology. As a result, selection of these parameter values can be calibrated empirically by each node based on "training" samples to characterize baseline traffic under non-attack conditions, if such conditions are identifiable and likely to hold constant over time. However, it is of course also possible to characterize the frequency of internal violations of these thresholds by the distribution of baseline traffic itself (e.g. hence the definition in the normally distributed case in terms of σ as an estimate of expected frequency of departure from the mean). Then the challenge becomes to select a starting r_U that increases the performance on iterative re-definition based on the algorithm's feedback from network neighbors, rather than approximating a random walk around the initial value. (Note: this naturally becomes more difficult as the underlying distribution becomes harder to characterize.) This is logically akin to searching a continuous curve for a local maximum from an auspicious starting value, and therefore only meaningful in expectation rather than concrete calculation for any given node/topology/distribution.

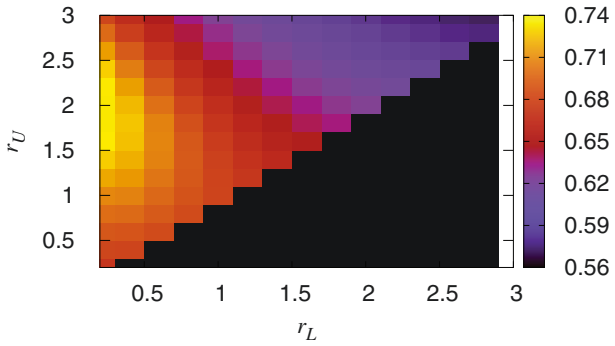

Figure 7 Accuracy for different combinations of the parameters r_L and r_U on a ER network with $N = 5000$ and $\langle k \rangle = 3$. The fraction of compromised nodes in the attack is $p = 0.3$ and the size of each packet is $z = 20$.

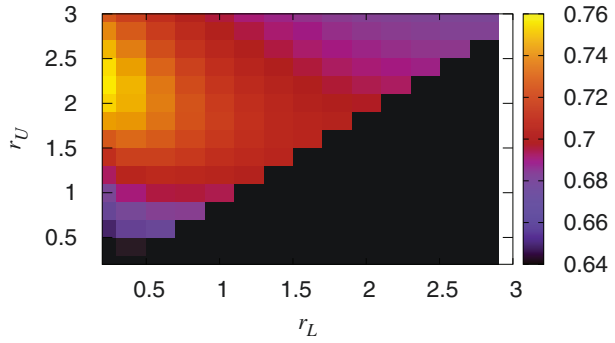

Figure 8 Accuracy for different combinations of the parameters r_L and r_U on a SF network with $N = 5000$ and $\gamma = 2.5$. The fraction of compromised nodes in the attack is $p = 0.3$ and the size of each packet is $z = 20$.

In this model, we exhaustively explore the behavior of the algorithm under different combinations of r_L and r_U in order to determine which one gives the more accurate results. Without loss of generality, we impose the simple, definitional condition that $r_L \leq r_U$. In Figures 7 and 8 we show the accuracy of the DIAMoND algorithm for an isolated Erdös-Rényi network (ER) and a Scale Free (SF) network with $\gamma = 2.5$ respectively, under a small attack (30% of the nodes are compromised and the size of malicious packets is $z = 20$). We found a maximum value for the pair $r_L = 0.2$ and $r_U = 2$ for both topologies. This can indicate that the most efficient combination of parameters is independent from the topology on top of which the algorithm is taking place.

4 Alternative Measurements of Performance

Although algorithmic performance metrics in cybersecurity and anomaly detection frequently focus on accuracy as the most relevant outcome, there may also be cases in which other metrics are more critical to ensuring stability or performance. We therefore, in addition to the already-presented results for accuracy, consider the effect that interconnection between layers has on algorithmic performance in terms of sensitivity, specificity, and precision. Each one is defined as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{6}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{7}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{8}$$

where $TP =$ true positive, $TN =$ true negative, $FP =$ false positive, and $FN =$ false negative.

In Figure 9, we analyze each of these four metrics of performance in the context of Model 2, presented above (Section 3.2). In particular, we present the results for an Erdös-Rényi (ER) network with average degree $\langle k \rangle = 3$, when it is connected to another ER network (black line), a SF network with $\gamma = 2.5$ (red line) or a SF network with $\gamma = 3.5$ (blue line). The two networks are connected using the rules explained in Section 3.2 (model 2). We show how these quantities differ from the case of an isolated network. It is important to point out that the precision and sensitivity corresponding to the isolated network are already superior to 99%. Thus, it is not possible to out-perform them considerably, limiting our potential for improvement to approximately 1, which is consistent with the observed results. Further, in analyzing the sensitivity of the algorithm (i.e. the fraction of true positive hits, out of all the positive reported cases), we find an increase in the sensitivity when the packets are small and when the network is connected either to another ER network or a SF network with $\gamma = 2.5$. Critically, however, we observe a decrease when the packets are larger, or when the network is connected to a SF network with $\gamma = 3.5$. In both cases, the difference is around 3%. We therefore note that, should sensitivity be the most critical performance metric in targeted application, careful analysis of the network/traffic scenario is necessary to

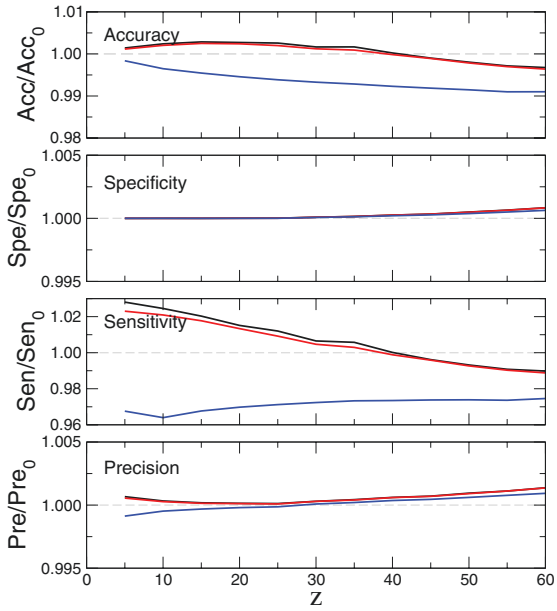

Figure 9 Performance of the DIAMoND algorithm as a function of the size of the packets for an ER network with average degree $\langle k \rangle = 3$, when it is connected to another ER network (black line), a SF network with $\gamma = 2.5$ (red line) or a SF network with $\gamma = 3.5$ (blue line). The networks are connected using the rules explained in Section 3.2 (Model 2). The rest of the parameters are: $N = 5000$, compromised nodes $p = 0.3$, fraction of connected nodes $q = 0.8$.

determine whether or not DIAMoND-type, distributed algorithms should be deployed.

5 Summary

We believe that it is time to shift the practices of the field of bio-inspired algorithm design away from case-by-case analogue building and experimental trial-and-error testing of minor design changes. Instead, we propose a method where bio-inspired work starts with an analogue system, but rather than trying to mimic that system, researchers focus on abstracting the fundamental component features and the interactions among them that yield algorithmic success across contexts. Once accomplished, these components and interactions may be designed and tailored to the needs of human systems, rather than trying to stretch analogies further and further from their initial context.

In the specific case of extending DIAMoND to function on multilayer networks, this approach (though still ongoing) has allowed us to eliminate some axes of potential concern (e.g. ambiguity in lack of concern) while instead directing our attention towards more potentially fruitful avenues of design (e.g. weighting the contribution of information learned from network neighbors depending on layer). We hope that, although we here focused on a case study in which each next proposed component was still studied via empirical manipulation, processes of abstraction may also enable the eventual analytic optimization of such designed systems.

We believe these "mathematical experiments" in abstraction can lay the groundwork for understanding what types of functional components determine algorithmic behaviors of all bio-inspired systems, but in particular will be critical in our understanding of distributed detection systems such as DIAMoND.

Acknowledgments

The authors acknowledge support by NSF under Grants CNS-1646856 (GPS and LKG) and CNS-1646890 (NHF).

References

[1] Mazurczyk, W., Drobniak, S., and Moore, S. (2016). Towards a systematic view on cybersecurity ecology. In *Combatting Cybercrime and Cyberterrorism* (pp. 17–37). Springer, Cham.

[2] Enache, A. C., and Sgârciu, V. (2015). Anomaly intrusions detection based on support vector machines with an improved bat algorithm. In *20th International Conference on Control Systems and Computer Science (CSCS)*, (pp. 317–321). IEEE.

[3] Gowri, R., and Rathipriya, R. (2016). Venus Flytrap Optimization. In *Computational Intelligence, Cyber Security and Computational Models* (pp. 519–531). Springer, Singapore.

[4] Cho, J. H., Shin, J. Y., Lee, H., Kim, J. M., and Lee, G. (2015). DDoS Prevention System Using Multi-Filtering Method. In *International Conference on Chemical, Material and Food Engineering*. Atlantis Press.

[5] Mirkovic, J., and Reiher, P. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2), 39–53.

[6] Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. (2017). *Network traffic anomaly detection and prevention: concepts, techniques, and tools.* Springer.

[7] Korczynski, M., Hamieh, A., Huh, J. H., Holm, H., Rajagopalan, S. R., and Fefferman, N. H. (2015). DIAMoND: Distributed intrusion/anomaly monitoring for nonparametric detection. In *24th International Conference on Computer Communication and Networks (ICCCN), 2015* (pp. 1–8). IEEE.

[8] Korczynski, M., Hamieh, A., Huh, J. H., Holm, H., Rajagopalan, S. R., and Fefferman, N. H. (2016). Hive oversight for network intrusion early warning using DIAMoND: a bee-inspired method for fully distributed cyber defense. *IEEE Communications Magazine*, 54(6), 60–67.

[9] Winston, M. L. (1991). *The biology of the honey bee.* harvard university press.

[10] Gallos, L. K., Korczyński, M., and Fefferman, N. H. (2017). Anomaly detection through information sharing under different topologies," *EURASIP Journal on Information Security*, No. 1, p. 5.

[11] Rinaldi, S. M., Peerenboom, J. P., and Kelly, T. K. (2001). Identifying, understanding, and analyzing critical infrastructure interdependencies. *IEEE Control Systems*, 21(6), 11–25.

Biographies

Gonzalo P. Suárez is a Postdoctoral Associate at the Department of Ecology and Evolutionary Biology (EEB) at the University of Tennessee. Before that, he held a Postdoctoral Associate position at the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) at Rutgers University. He received his PhD in Physics from the National University of Mar del Plata, Argentina.

Lazaros K. Gallos is Associate Director and Research Professor at DIMACS (the center for Discrete Mathematics and theoretical Computer Science) at Rutgers University. He has been a Research Associate at the Department of Ecology at Rutgers University and at the Levich Institute at the City College of New York. He received his PhD in Computational Physics from the Department of Physics in the University of Thessaloniki.

Nina H. Fefferman is Full Professor in the Department of Ecology and Evolutionary Biology and in the Department of Mathematics at the University of Tennessee. She is also a member of The National Institute for Mathematical and Biological Synthesis (NIMBioS). She received her PhD from Tufts University in Biology, her M.S. from Rutgers University in Mathematics and her A.B. from Princeton University.