COD: Online Temporal Clustering for Outbreak Detection

Tomáš Šingliar[1]
tomas@cs.pitt.edu

Denver H. Dash[2,3]
denver.h.dash@intel.com

[1]Department of Computer Science
University of Pittsburgh
210 S Bouquet St
Pittsburgh, PA 15260

[2]Intel Research
4720 Forbes Ave
Pittsburgh, PA 15213

[3]Department of Biomedical Informatics
University of Pittsburgh
200 Meyran Ave
Pittsburgh, PA 15260

Abstract

We present *Cluster Onset Detection* (COD), a novel algorithm to aid in detection of epidemic outbreaks. COD employs unsupervised learning techniques in an online setting to partition the population into subgroups, thus increasing the ability to make a detection over the population as a whole by decreasing the signal-to-noise ratio. The method is adaptive and able to alter its clustering in real-time without the need for detailed background knowledge of the population. COD attempts to detect a cluster made up primarily of infected hosts. We argue that this technique is largely complementary to the existing methods for outbreak detection and can generally be combined with one or more of them. We show empirical results applying COD to the problem of detecting a worm attack on a system of networked computers, and show that this method results in approximately 40% lower infection rate at a false positive rate of 1 per week than the best previously reported results on this data set achieved using an HMM model customized for the outbreak detection task.

Introduction

The increasing connectivity of people and devices has given rise to much interest in the ability to detect epidemic outbreaks, both biological and computational. This connectedness facilitates the spread of viral agents, and at the same time provides us with unprecedented computation, communication, and data collection capabilities to prevent and detect them.

The interest in automatic event detection is hardly new and many methods based on time-series analysis have been proposed. They range in complexity from simple count aggregation (Cooke *et al.* 2004; Rajab, Monrose, & Terzis 2005; Anagnostakis *et al.* 2003) to cumulative sums (Bissell 1969) to hidden Markov models capturing the dynamics of the epidemics and its underlying population (Rath, Carreras, & Sebastiani 2003; Dash *et al.* 2006). For a summary of these techniques, see (Wong & Moore 2006).

Many traditional outbreak detection methods treat the population as a monolithic entity; however, real populations are heterogeneous, and different subpopulations are susceptible to different degrees. For instance, living in proximity of an infection source defines a subpopulation at risk of suffering a disease outbreak; the set of Windows machines with an unpatched vulnerability form a subpopulation at risk of being infected with a virus, etc. In principle, this fact can be exploited to boost the signal-to-noise ratio for detection by breaking the population up into smaller subgroups which contain individuals who are more likely to be infected than the population as a whole.

COD uses unsupervised clustering in an attempt to detect susceptible subpopulations without detailed background knowledge of what makes the subgroups susceptible. We design our detection method around a general probabilistic clustering algorithm formulated as a Bayesian network. The design of clustering features is dictated by the available information and by the necessity to model daily variation which otherwise becomes a source of false positives. The clustering is periodically updated as the data flows in, and COD looks for a highly active cluster in the resulting cluster set.

There has been quite a bit of work seeking to exploit small susceptible subpopulations. The method of spatial scan statistics (Kulldorff & Nagarwalla 1995; Neill, Moore, & Cooper 2006) attempts to look for spatial over-densities of disease indicators. Spatial scans are a special case of COD, where the feature driving the clustering is the spatial Euclidean distance. COD allows clustering over arbitrary attributes, including indicators of infection, so in principle is capable of detecting less obvious subpopulations. WSARE (Wong *et al.* 2003) and PANDA (Cooper *et al.* 2004) are two other methods for epidemic outbreak detection that operate on the principle of partitioning the population into subgroups. WSARE uses rules which encode background knowledge regarding which combinations of attributes are likely to define a meaningful subgroup for the purposes of detection. PANDA uses a Bayesian network to specify conditional independence between attributes. These methods differ from COD in that subpopulations over which they search are specified in advance using background knowledge implicit either in the rules or in the structure of the Bayesian network. The main advantage of COD on the other hand is that it allows arbitrary clusters to be determined dynamically. This endows it with the ability to detect susceptible subpopulations that have not been envisioned in advance. While COD does not explicitly require prior knowledge, any background knowledge in the form of causal influences of some attributes can be easily incorporated.

One of the key applications we study in this paper is

that of detecting a slow worm outbreak in a system of networked computers. State-of-the-art techniques for worm detection look for strings of bits that uniquely identify a particular known worm attack (Roesch 1999; Paxson 1999); however these techniques are limited to worms that have known signatures. More advanced techniques attempt to learn the signatures on the fly by searching for repeated bit patterns in packet payloads (Kim & Karp 2004); however these are intensive in both computation and bandwidth, requiring the processing of data deep within packets. Also the effectiveness of these techniques on slowly propagating worms has not been established[1]. It has been fairly well established (Weaver, Staniford, & Paxson 2004) that fast-spreading worms can easily be detected simply by checking for abnormally high connection rates at the local host; therefore these types of worms are not a suitable test to detect outbreaks in computer networks. Slow worms, on the other hand, deliberately space out their attacks in time and can be extremely difficult to detect based on traffic activity alone, yet they still can exploit exponential growth (Dash *et al.* 2006).

Worm detection in general is an active area of research, and there has been much work taking systems approaches (Cooke *et al.* 2004; Rajab, Monrose, & Terzis 2005; Anagnostakis *et al.* 2003) where aggregation points in the network attempt to count events over several machines and draw conclusions about the system as a whole. However, this work deals primarily with systems and networking related challenges and with identifying effective indicators of infection. COD is complementary to these techniques as it would take as input the indicators generated by them.

We demonstrate that in our application, COD can raise an alarm at approximately a 40% lower infection penetration at a low false positive rate than the best reporting detector on our data (Dash *et al.* 2006). As the number of hosts being monitored increases, COD not only scales gracefully with respect to resources, but the detection performance improves as well.

Intrusion Detection Architecture

Here, we describe in more detail the real-world detection setting to which we apply the COD algorithm, taken from (Dash *et al.* 2006).

A corporate network is being monitored for a worm outbreak, and each host is embedded with a noisy Local Detector (LD). The LDs use information available at the host level, such as the number of packets sent in a given time interval, to detect infection of their host by a worm. The local detectors periodically send their status to a global detector that resides on a centrally located machine. As the local detectors possess no signature of the worm, they inevitably suffer from a very high false positive rate, on the order of 10^4 per week. While the number of detections varies widely between hosts even in the clean, non-attack traffic, adaptive LDs that approach a nearly fixed pre-set false positive rate are being explored (Agosta *et al.* 2007).

[1]While a slow worm provides their algorithm more time to process bits, it also may reduce the number of independent corroborating packets at a given time.

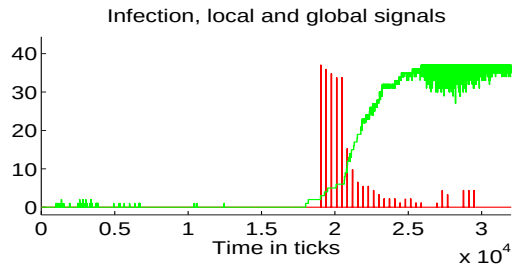

Figure 1: Example local and global signal plots. The sum of the local signals is represented by the green line, the global signal of the centralized detector is red. Infection starts at time step 18000 in this plot.

A global detector (GD) collects the messages from the local detectors and determines whether the positive local detections corroborate each other. The GD periodically outputs a signal that represents its belief of infection being present, based roughly on the strength of the corroboration. The local and global signals of a typical run are visualized in Figure 1. Shortly after the infection, the number of local detector firings begins to rise gradually, followed soon by a sharp spike in the global signal. The local detectors we use are simple rules that fire whenever the number of outgoing connections exceeds 4 in a 50 second interval; however, in principle the exact nature of the local detectors is not directly relevant to our technique. The feature that we depend on is that the local signals are a noisy indicator of some sort of outbreak. For example, the number of respiratory cases reported in an hospital Emergency Department in a unit time would be a noisy indicator of an influenza outbreak.

The COD Model

In this section, we describe the two components of the COD method; the clustering method and how the global signal is generated from the obtained clustering.

The soft clustering algorithm we use is equivalent to ML parameter learning of a graphical probability model (Figure 2) from a dataset $\mathbf{X}$ where each row $\mathbf{X}_{i.}$ corresponds to a single local detector i and each column $\mathbf{X}_{.j}$ to the value of a feature function in a discrete time interval j.

Each time interval has a fixed position with respect to the wall clock: the first interval might correspond to 12am-1am, the second to 1am-2am etc. This is a special case of temporal stratified sampling to account for obvious diurnal behavior in this system. The setup can generalize to any process with a different periodicity. Stratified sampling is thus one easy way to include background knowledge into this method; however, if a characteristic period is not known a priori, our method will in principle rectify non-stationarity as long as the *changes* in activity affect the clusters uniformly. This robustness is due to the way in which we perform cross-cluster hypothesis testing, as described below.

The particular graphical probability model we use in this proof-of-concept is a standard Naive Bayes clustering model with hidden class ("cluster") variable M, although in principle any clustering method could be used. In our worm-

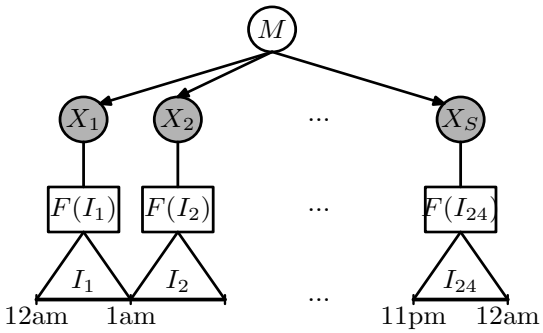

Figure 2: The NB soft-clustering model used for the worm intrusion detection task. The feature function computes, for each machine, the sum X_{ij} of positive detections reported by machine i in interval j. The counts are used as data for the graphical probability model that forms the top part of the drawing. To incorporate any global information that might be available, we would create random variable nodes as parents of the feature nodes in this model.

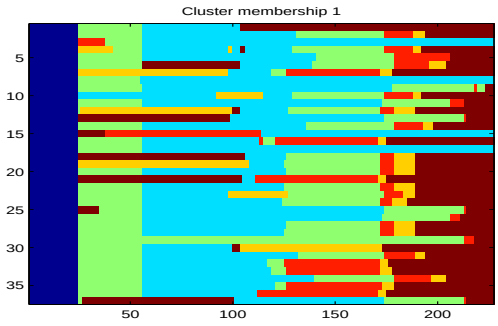

Figure 3: A membership plot. There are 5 clusters in this plot. Time in hours is on the x-axis, host index on the y-axis. Color indicates the cluster to which the host belongs at that time. The solid-colored portion on the left corresponds to the one-day burn-in period, where clustering is not performed. At the end of the run, most hosts lie in the reddish cluster because they have been infected.

detection application, the NB features are positive local detection counts X_{ij} arriving from a machine i during a time *interval* j. The time axis in the model of Figure 2 has been broken down into 24 one-hour intervals. If we assume that members of different classes generate their detections randomly at different rates and can take a fairly large range of values, it is reasonable to assume that X_{ij} are Poisson-distributed, conditional on their membership in cluster k:

$$P(X_{ij} = x | M = k) = \frac{e^{-\lambda_{kj}} \lambda_{kj}^{x}}{x!}.$$

There is nothing that prevents us from having several feature functions per interval; we stick with one feature per interval for direct comparison to previous approaches and to make use of the existing dataset.

At the end of each interval, the corresponding feature value (a column in the dataset $\mathbf{X}$) is updated and the graphical model is re-learned with the Expectation-Maximization algorithm (Dempster, Laird, & Rubin 1977), using as initialization the previous model parameterization. The convergence tends to be fast, as not much data has changed since the last learning iteration. The posterior on the cluster variable M defines the assignment of local detectors into clusters:

$$Membership(i) = \operatorname*{argmax}_{k} P(M = k | \boldsymbol{X}_{i.}),$$

where $\boldsymbol{X}_{i.}$ is the i-th row of the data matrix. Figure 3 shows a typical example of how the hosts in our data set get assigned into clusters. The clusters are rather stable and cluster membership changes rarely. This manifests itself in long horizontal streaks of the same color in the plot.

The clustering algorithm will effectively group the hosts according to the daily pattern of their local detection activity as shown in Figure 4. It is reasonable to assume that the detection activity pattern reflects the applications and habits of the host and provides the best estimate of the vulnerability pattern we can readily produce based on the binary local detection information alone.

When the infection occurs, the local detectors will detect it very reliably. [2] The dataset column having the most recent detection count will be filled with relatively high detection counts. Eventually, the infected hosts will be put into one cluster when the cluster assignment becomes dominated by the difference between the detection counts generated by the infected and those by the still healthy hosts.

One question to be resolved is how to determine the number m of clusters. We use a greedy heuristic that increments m until the Bayesian Information Criterion (BIC) (Chickering & Heckerman 1997) for the model stops improving. We have static and adaptive variants: The static variant learns the optimal m over a training set of data and uses this for subsequent testing. The adaptive variant keeps models for all values of m up to the highest warranted by the BIC so far and attempts to adapt the optimal m on the fly. We report results for both variants, but predominantly use the static version for sake of speed.

Cluster Interpretation

After the hosts are assigned into clusters, the COD detection algorithm attempts to detect the presence of a highly active – presumably infected – cluster by:

1. computing for each host i its average detection rate over the last T GD ticks: $ADR(i) = N_1(i)/T$, where $N_1(i)$ is the number of positive detections at host i. The parameter T is tunable.
2. computing the average (local) detection rate in each cluster and identifying the most active cluster
3. performing a one-sided, unbalanced-design t-test with null hypothesis that the host detection rates in the most active cluster and remainder of the population are the same. [3]

[2] They pay a price for that sensitivity in their false positive rate.

[3] Singleton clusters are excluded as they have divergent sample variances.

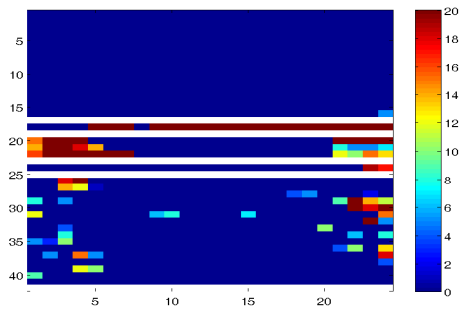

Figure 4: A detection rate plot shows the clustering dataset. The x-axis shows the 24 detection intervals, the most recent on the right. Color indicates the local detection count that the host reported for that interval. The y-axis corresponds to host index. The hosts are sorted according to their cluster memberships (the white horizontal lines separate the five clusters, two of which are composed of a single member).

4. comparing the outcome of the t-test to a historical histogram of values to determine if the system is in an anomalous state.

Note that we use step 4 rather than using a t-distribution, because by artificially gathering highly-firing hosts into the same cluster we would incur a selection effect and it would not be surprising at all to routinely see t-test statistics well above a typical p-value. Instead, we compare the outcome to past values and calculate an empirical p-value to rectify the selection bias, while still being able to take advantage of a higher signal-to-noise ratio that might occur in the smaller subpopulations.

Performance Metrics

We assume that as the infection spreads to the entire population, eventually we will be able to detect with a true-positive rate of 1. The question in outbreak detection is not *whether* we detect, but rather *how soon* we can detect. Thus natural performance metrics are the time to detection (TTD) and the false alarm rate (FAR). In the case of a worm spread, one may further be more interested in the fraction of hosts that have been infected (FI) at the time of detection. FI is directly related to TTD and can be viewed as TTD "normalized" with respect to worm propagation speed. The typical FAR that we deemed acceptable for this application was 1 false positive (FP) per week.

As for the TP-FP tradeoff of an ROC curve, there is an intrinsic tradeoff between these two metrics, depending on the p-value used to raise an alert. As this operating point is difficult to set in advance, we use a variant of the Activity Monitor Operating Characteristic (AMOC) curve (Wagner, Moore, & Aryel 2006) to succinctly represent the performance throughout the entire set of possible alarm thresholds.

Experimental Evaluation

The models are evaluated in a C++ simulation environment. The simulator replays approximately 5 weeks' worth of captured traffic traces from 37 hosts on a corporate network. There were no worm intrusions during the capturing process and the data is certified to be clean by IT professionals. Worm traffic is generated from a simplified worm model and is injected on top of the clean background traffic after an initial burn-in period. To determine the false alarm rate, the traffic traces were replayed and detectors run without the worm traffic injected. We currently do not model recovery from infection. Each positive run proceeds until saturation and the global detector is restarted with a new one-day burn-in period at the beginning of the next run.

The Local Detectors send a message containing their belief state every 10 seconds. The burn-in period for the model is set to two days. If more than 37 nodes are needed for an experiment, the traffic traces are recycled. To generate the data coming from a "fictitious" host, a circular buffer is used that begins at a random point in one of the trace files.

For each model tested, five AMOC curves are generated by running the experiment with different random seeds. For the sake of presentation clarity, we will not plot all five curves or standard deviation bars, but rather plot the "median" curve out of the 5 runs. More precisely, we plot the curve that achieves the median FI at the target FAR of 1 per week. The AMOC curves themselves are averages over as many independent runs as the amount of collected traffic traces permits (depending on the model, 9 to 20).

Epidemic-DBN The baseline is set by the Epidemic Dynamic Bayes Network detector (E-DBN), a model that has been compared to several standard statistical techniques for outbreak detection and shown to outperform them all (Dash *et al.* 2006). This detector is a Dynamic Bayesian Network (Dean & Kanazawa 1990) that captures the dynamics of an epidemically spreading worm (Chen, Gao, & Kwiat 2003). It is a three-layered HMM, where the top layer is a binary indicator of attack, the middle (unobserved) layer models the number of nodes infected and the bottom (observed) layer models the number of positive LD firings. E-DBN's messaging scheme can be centralized like COD's, or distributed with local detectors communicating to a random subset of their neighbors. We compare to the distributed setup of E-DBN, as it outperforms the centralized setup and therefore sets a stricter baseline (Dash *et al.* 2006). In our experiments, we use the E-DBN with chain length 20 to be consistent with (Dash *et al.* 2006).

Figure 5 compares the performance of the E-DBN global model to the proposed model. COD clearly outperforms the DBN global detector. At the target FAR, the FI is reduced by about a third from 5% to 3.3%. The performance advantage perhaps even widens in the lower FAR range.

Effect of model parameters. Our detection method has a number of parameters whose effect should be explored in order to set them meaningfully. The effect of parameter variation is examined by varying one parameter at a time in this configuration, leaving the others fixed to prevent the explosion of possible detector configurations. The "standard" parameterization is 500 nodes, centralized detector, number of clusters fixed at 5, 24 intervals per day.

Number of clusters. Figure 5 demonstrates our finding that the adaptive detector does not differ obviously from the statically optimized COD model with 5 clusters. As the per-

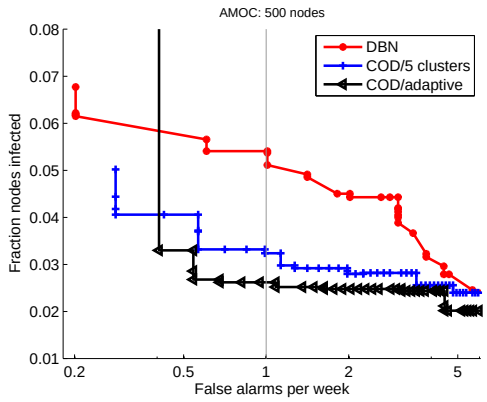

Figure 5: AMOC curves for the best known parameterizations of DBN versus the new model, obtained for 500 nodes. Error bars were dropped in most figures to improve clarity. Figure 6 gives a sense of the amount of variance in the AMOC curve.

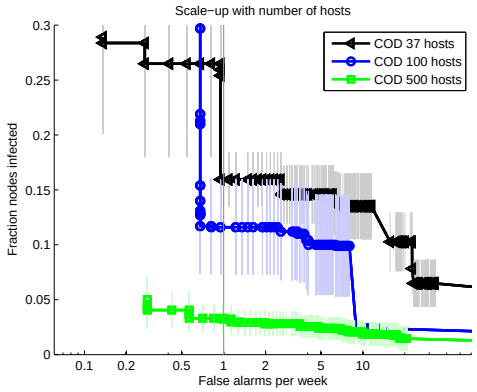

Figure 6: Scaling with the number of nodes: 37, 100 and 500 DDI hosts in the network. The error bars represent one standard deviation in FI, computed from 9 independent runs exhausting the available data. The curves themselves are "medians", defined in the text.

formance does not differ much, and as the adaptive detector was more costly to run, for the remainder we will report only the static COD model results.

Scaling with network size. Figure 6 documents a very favorable property which was also observed with the DBN detector: that the performance actually improves with scaling of the system. The larger number of datapoints gives the models more information and refines the clustering. With small number of nodes, the DBN distributed detector, which does not learn its parameters, tends to outperform COD. With fixed target FAR, the fraction of hosts infected at time of detection shrinks as the network size increases.

Interval length. The interval length affects the performance in two ways: 1) More frequent re-clustering eliminates part of the "mid-interval blind spot"; 2) longer intervals yield features with less variance. Accordingly, varying the interval length has an effect on performance that balances these opposite effects, as shown in Figure 7. Generally, better performance is achieved with longer intervals that permit longer accumulation of detection counts, perhaps better smoothing over any random fluctuation. This hypothesis is also supported by the lower variance in performance with longer intervals. Also, the lower frequency of the detection algorithm invocation gives fewer opportunities for generating false alarms. The price of delayed detection is not too great in the case of a slow worm, like the one modeled here; though the situation might change for faster worms we do not consider in this paper. For brevity, we forgo discussing the effect of the **activity window length** T.

Complexity and scalability. With 100 nodes and reclustering every hour of simulated time, the simulator runs several hundred times faster than real-time on a mass-market desktop computer, including the non-negligible simulation overhead. Thus scaling to enterprise-size networks with tens of thousands of computers is not a problem from the computational complexity perspective.

The messaging scalability presents more of a challenge for the centralized detector. Although the bandwidth required for the information content itself is small even in a large network – tens of kilobits per second – the protocol overhead is substantial. Fortunately, enterprise networks tend to be organized in a hierarchical manner. It is certainly feasible to set up a hierarchy of message aggregation nodes to deal with the networking overhead. Deeper connections between enterprise network topology and the detection framework are investigated in (Sherr *et al.* 2007).

Summary and Future Work

We proposed COD, a detection algorithm for rare events based on partitioning the potential hosts into subgroups that are more homogeneous in their usage and application patterns and, by implication, their vulnerability to a worm attack. The method does not require incorporation of extensive prior knowledge to achieve usable results. Experimental evaluation on the problem of detecting a worm with a limited attack rate showed that COD can cut the number of hosts that are infected at the time of detection by over a third. The algorithm is efficient and has a good scale-up potential.

Several avenues for possible improvement are open to consideration. Using more precise information about the operating system patch level, running proceses and similar features, groups of hosts could be identified that are likely to share a vulnerability. Presumably, elevated local detection activity in such cluster provides an even more reliable infection clue. We are developing the capability of collection and transmission of such OS data.[4] We would like to apply COD to detection of disease outbreaks. However, while COD can deal with missing data in a principled way, the stream of information from each person (host) would be very spotty, very possibly leading to performance degradation. Thus COD would be more promisingly employed in an

[4]Because such information is also valuable to the attacker, there are security issues to be addressed, but they are beyond this scope of this paper.

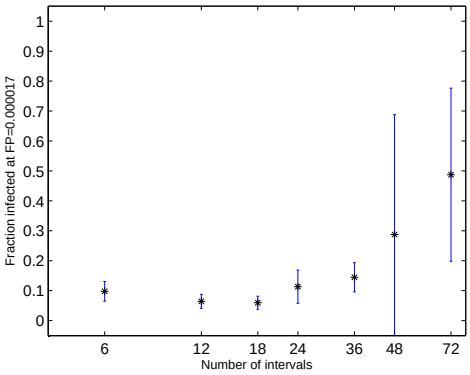

Figure 7: Detection performance changes with varying number of day slices. On the horizontal axis, the number of slices (e.g. 48 slices gives 30 minute aggregation). On the vertical axis, the fraction of nodes infected when detector is configured to operate at the target FPR (1 per week). Error bars represent one standard deviation.

environment with denser data, such as detection of nosocomial epidemics.

The signal-generating procedure is based a sound statistical technique, but the definition of the test populations is somewhat heuristic. Moreover, taking the argmax potentially loses some information contained in the class posterior. Other heuristics to interpret the clusters are being considered.

Acknowledgments

We would especially like to thank Micah Sherr for building much of the simulator infrastructure used in this paper. We would also like to thank John Mark Agosta, Jaideep Chandrashekar, Carlos Diuk, Carl Livadas, and Eve Schooler for invaluable feedback.

References

Agosta, J. M.; Diuk, C.; Chandrashekar, J.; and Livadas, C. 2007. An adaptive anomaly detector for worm detection. In *Proceedings of sysML-07*.

Anagnostakis, K. G.; Greenwald, M. B.; Ioannidis, S.; Keromytis, A. D.; and Li, D. 2003. A cooperative immunization system for an untrusting internet. In *Proceedings of ICON*.

Bissell, A. 1969. Cusum techniques for quality control. *Applied Statistics* 18:1–30.

Chen, Z.; Gao, L.; and Kwiat, K. A. 2003. Modeling the spread of active worms. In *Proceedings of INFOCOM*.

Chickering, D. M., and Heckerman, D. 1997. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning* 29(2-3):181–212.

Cooke, E.; Bailey, M.; Watson, D.; Jahanian, F.; and Nazario, J. 2004. The internet motion sensor: A distributed global scoped internet threat monitoring system. Technical Report CSE-TR-491-04, University of Michigan, Electrical Engineering and Computer Science.

Cooper, G.; Dash, D.; Levander, J.; Wong, W.-K.; Hogan, W.; and Wagner, M. 2004. Bayesian biosurveillance of disease outbreaks. In *Proceedings of UAI*, 94–103.

Dash, D.; Kveton, B.; Agosta, J. M.; Schooler, E.; Chandrashekar, J.; Bachrach, A.; and Newman, A. 2006. When gossip is good: Distributed probabilistic inference for detection of slow network intrusions. In *Proceedings of AAAI 2006*.

Dean, T., and Kanazawa, K. 1990. A model for reasoning about persistence and causation. *Computational Intelligence* 5(3):142–150.

Dempster, A.; Laird, N.; and Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B* 39(1):1–38.

Kim, H.-A., and Karp, B. 2004. Autograph: Toward automated, distributed worm signature detection. In *USENIX Security Symposium*, 271–286.

Kulldorff, M., and Nagarwalla, N. 1995. Spatial disease clusters: detection and inference. *Statistics in Medicine 14* 799–810.

Neill, D.; Moore, A.; and Cooper, G. 2006. A Bayesian spatial scan statistic. In Weiss, Y.; Schölkopf, B.; and Platt, J., eds., *NIPS*. Cambridge, MA: MIT Press. 1003–1010.

Paxson, V. 1999. Bro: a system for detecting network intruders in real-time. *Comput. Networks* 31(23-24):2435–2463.

Rajab, M. A.; Monrose, F.; and Terzis, A. 2005. On the Effectiveness of Distributed Worm Monitoring. In *Proceedings of the* 14^{th} *USENIX Security Symposium*, 225–237.

Rath, T. M.; Carreras, M.; and Sebastiani, P. 2003. Automated detection of influenza epidemics with hidden markov models. In *IDA*, 521–532.

Roesch, M. 1999. Snort - lightweight intrusion detection for networks. In *LISA '99: Proceedings of the 13th USENIX conference on System administration*, 229–238. Berkeley, CA, USA: USENIX Association.

Sherr, M.; Agosta, J. M.; Chandrashekar, J.; Dash, D. H.; Livadas, C.; and Schooler, E. M. 2007. Collaborative detection and containment of stealthy worms in the enterprise. In *Recent Advances in Intrusion Detection 2007 (under review)*.

Wagner, M. M.; Moore, A. W.; and Aryel, R. M., eds. 2006. *Handbook of Biosurveillance*. Elsevier.

Weaver, N.; Staniford, S.; and Paxson, V. 2004. Very fast containment of scanning worms. In *USENIX Security Symposium*, 29–44.

Wong, W.-K., and Moore, A. W. 2006. Classical time-series methods for biosurveillance. In Wagner, M. M.; Moore, A. W.; and Aryel, R. M., eds., *Handbook of Biosurveillance*. Elsevier. 217–234.

Wong, W.-K.; Moore, A.; Cooper, G.; and Wagner, M. 2003. What's strange about recent events. *Journal of Urban Health* 80:i66–i75. Supplement 1.