Network Discovery with Multi-intelligence Sources

Michael J. Yee, Scott Philips, Gary R. Condon, Peter B. Jones, Edward K. Kao, Steven T. Smith, Christian C. Anderson, and Frederick R. Waugh

Analysts can glean much useful intelligence information from identifying relationships between individuals and groups, and tracking their activities. However, detecting networks of people and then investigating their activities are difficult tasks, especially in this era of information overload. Graph analysis has proven to be a useful tool for addressing these tasks, but it can be labor-intensive. To aid in this analysis, Lincoln Laboratory researchers developed a diffusion-based analytic that helps solve the problems of network discovery and prioritized exploration. This analytic, called *threat propagation,* has been demonstrated to effectively handle network detection and exploration tasks, and has been integrated into an interactive tool for generating networks from wide-area motion imagery.

» **As early as 2002, the U.S. government** was seeking a courier believed to be connected to Osama bin Laden. The courier was known as Abu Ahmed al-Kuwaiti, and his identity had been confirmed by more than one captured high-level al-Qaeda operative. In 2010, a telephone wiretap of a suspect led intelligence analysts to al-Kuwaiti. On the basis of this connection, al-Kuwaiti was eventually located in August 2010 and was tracked to a compound near Abottabad, Pakistan. After intelligence analysts gathered corroborating evidence, U.S. Navy SEALs raided the compound, and Osama bin Laden was found and killed.

Working from tips and cues, such as that of al-Kuwaiti, and tracking them back through network connections created by communications, personal movement, and monetary transactions, intelligence analysts are able to uncover criminal and terrorist networks. By following the connections, rather than relying solely on the tips or cues, analysts can find higher-level network operatives, and network operations can be more completely disrupted.

However, exploiting such connections requires resolving two related problems: network discovery and network exploration. The network-discovery problem consists of uncovering the subset of vertices within a network that exhibit a particular attribute or activity of interest. For example, given a social network, one may wish to detect all individuals belonging to a particular faction or gang.

A related problem is network exploration. Often, the activities used to construct a graph (such as communications between computers or transit between locations) take time and effort to verify. Given the scale of modern networks, constructing them edge by edge and link by link can be prohibitively costly in terms of the number of analyst hours required. For situations in which only a subset of the network or graph is of interest, a method to prioritize particular connections for investigation can dramatically reduce the amount of time and effort analysts must exert to construct the network graph.

Solving these complementary problems of graph discovery and exploration is a common objective in a wide variety of applications, including social network analysis, web advertising, law enforcement, and counterterrorism. A particular application of interest is the identification of geographic sites connected together by a set of time-stamped tracks [1–3]. As demonstrated by the example of bin Laden's courier, tracking individuals' movements from locations of known threatening activity to new locations can lead to significant intelligence gains.

Lincoln Laboratory researchers have developed a diffusion-based analytic that can be used to solve the problems of network detection and prioritized exploration. We have applied this analytic, called *threat propagation,* to the problem of network detection, demonstrating theoretically and empirically the effectiveness of the threat propagation metric. The threat propagation metric has also been adapted to the problem of prioritized exploration and has been integrated into an interactive tool for aiding intelligence analysts in creating threat networks from wide-area motion imagery (WAMI) data.

Analytic Framework

Starting from a classical definition of a graph, we have developed the conceptual basis of threat propagation, and then introduced a new graph concept: the space-time graph, which can be thought of as a time-sampled graph with a specific edge set. Using a continuous time stochastic process model, we show that the simple algebraic rules of the threat propagation algorithm accurately estimate the threat at specific vertices in the space-time graph. Additionally, the threat propagation algorithm can be viewed as the harmonic solution to Laplace's equation on the graph, and the corresponding algorithmic implementation is called harmonic threat propagation. We have also developed an alternative implementation, based on the application of the Perron-Frobenius theorem to a particular stochastic model.

Multi-intelligence Graphs

Mathematically, a graph G is simply a collection of vertices and edges, $G = (V, E)$. The vertices represent the entities in the network, and the edges represent relationships between entities. In a multi-intelligence (multi-INT) graph, the entities encoded by the vertices are observable across a range of sensor modalities, and the relationships encoded in the edges can represent correspondence between the entities in any of those modalities. For example, in a graph with geolocations represented as vertices, edges between them may arise because an electro-optical or infrared sensor (supplying imagery intelligence [IMINT]) detected transit between the two locations. Alternatively, an edge may exist because human intelligence (HUMINT) assets report that a single individual owns both properties. Or perhaps a particular radio signature (signal intelligence, or SIGINT) was observed at both locations. A multi-INT graph integrates all available relational information to create a unified topological view of the entities and their interactions. Figure 1 depicts a small multi-INT graph.

When the relations between entities are time-varying, the graph used to represent those relationships must also vary with time. A *space-time graph* is a sequence of graphs, $\{G_t\}_{t=0}^{T}$, that represents the sequence of interactions between graph entities. This notion of a space-time graph is particularly useful when the data underlying the graph are inherently transactional, such as vehicle tracks connecting geolocations or e-mails connecting users. Without the loss of generality, it can be assumed that the set of vertices in a space-time graph is identical across all time samples, and that only the edge set is varying. Thus, the set of vertices in a space-time graph will be denoted as V (as in the static graph), but the sequence of edge sets will be denoted as E_t.

Bayesian Diffusion

The concept of diffusion has been a very useful paradigm for performing various graph-based inference problems such as graph partitioning [4], attribute inference [5], and ranking [6]. At their heart, diffusion-based algorithms are based on the assumption that for a vertex in the graph, its features (e.g., importance or group membership) are correlated with those of its neighbors. Mathematically, such assumptions are often justified through the use of Markov random walk models, in which the transition between two vertices that are "close" to each

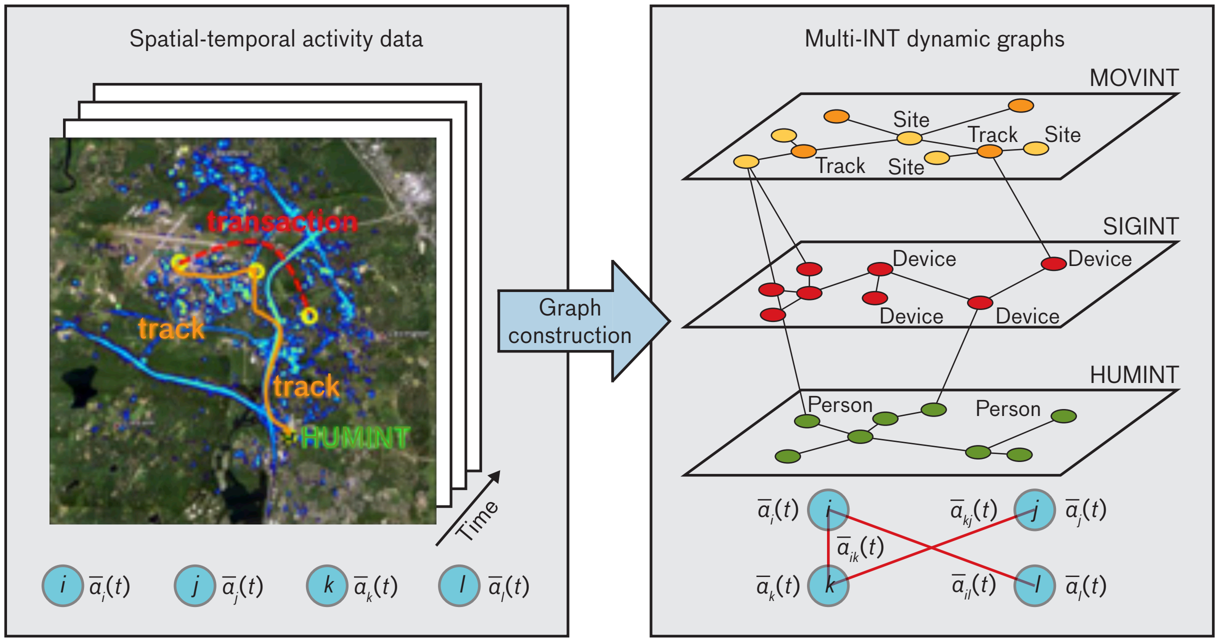


FIGURE 1. Multi-intelligence (Multi-INT) graphs fuse and enrich multimodal spatial-temporal data by adding associations and interactions between entities.

other in graph space is more likely than the transition between vertices that are "further apart."

The dynamic threat propagation algorithm presented in this article uses a diffusion process to predict which vertices in a space-time graph are expected to be interesting, given an observed cue. The mathematical formalism of this concept can be found in the companion article "Covert Network Detection" [7], in which it is shown that the basic equations of threat propagation follow directly from Bayes rule, given certain modeling assumptions and approximations. Detailed discussion of this concept can be found in the companion article, but a brief summary of the model is presented here.

At each vertex v, independent of interactions with other vertices, the presence or absence of threat is denoted by a binary stochastic process $\Theta_v(t)$, which is assumed to have the following dynamics: transitions from state 1 to state 0 follow a Markov jump process with Poisson rate λ_v, and no endogenous transitions occur from state 0 to state 1. Interactions between vertices affect the process in the following way: when vertex u interacts with vertex v beginning at time t_u and ending at t_v, with some fixed probability $P(v \leftarrow u)$, $\Theta_v(t_v) \leftarrow \Theta_u(t_u)$, where $\leftarrow$ indicates an assignment of one stochastic variable to the value of the other. Conceptually, in this model "threat" can be viewed as a virus that infects vertices for a random length of time and can replicate across vertices at the times of interactions.

Network Discovery

The network-discovery problem consists of uncovering within a network the subset of vertices that exhibits a particular attribute or activity of interest. It is closely related to the problem of graph partitioning in that it partitions the graph into two subsets (the *foreground* and the *background*); however, it differs in that most approaches to graph partitioning focus solely on the graph topology, while the network-discovery problem may incorporate additional information about the vertices. In our case, this additional information takes the form of tips: vertices that have been predetermined via some exogenous intelligence process to belong with some high probability to one of the partition elements. By using a diffusion process similar to that described in the section on Bayesian diffusion, the threat (meaning the probability a vertex belongs to the foreground) can be propagated to other vertices in the graph. The specific mechanism for diffusing this threat depends on whether the graph is a traditional (i.e., static) or space-time graph.

Propagation on Static Graphs

Starting with the static local community-detection problem, we assume that the presence and weight of each edge in the graph is known and fixed for all time. For example, in the case of an e-mail contact network, once an edge weight between a pair of vertices has been determined by integrating the history of e-mails between them, the actual timing of the e-mails is ignored. We detect vertices that belong to the community of interest by propagating the threat from tip vertex to other vertices along edges that represent interactions.

Similar propagation-based approaches have been effectively employed in a variety of network application areas. For example, in the spread of infectious disease, infectious agents are deposited at sites and transmitted from person to person as infection propagates throughout the graph. In social network analysis, the concept of eigenvector centrality (which posits that each vertex's importance is proportional to the sum of its neighbors' importances) can be calculated through distributed message propagation [8]. Google's PageRank algorithm [6] adapts eigenvector centrality to the information retrieval domain and posits that a web page has high rank if highly ranked pages link to it. Similarly, Kleinberg's HITS (for hyperlink-induced topic search) algorithm [9] defines hub and authority scores[1] in a mutually recursive way: the hub score of a page is a function of the authority scores of the pages that link to it, and the authority of a page is a function of the hub scores of the pages that link to it. When PageRank and HITS are computed iteratively by using power iteration to find the relevant dominant eigenvectors, the quantities of interest can be viewed as propagating along hyperlinks.

While based on the family of eigenvector centrality techniques, our approach to estimating threat generalizes previous developments in two key ways. First, we incorporate existing knowledge through threat estimates for tip vertices. In this framework, there can be multiple tip vertices—negative tip vertices (i.e., vertices with zero probability of threat) or tip vertices with any level of probability between zero and one. Second, we use a nonlinear propagation function that can vary by vertex type. This function adds flexibility in modeling the propagation "physics" for a particular domain.

Let $G = (V, E)$ be a graph, and denote the threat at vertex i as P_i. Denote the tip set to be $\mathcal{T} \subset V$ and the threat assessment π: $\mathcal{T} \rightarrow [0, 1]$. Define N: $V \rightarrow 2^V$ (where 2^V denotes the power set of V) to be the neighborhood functional (i.e., $j \in N(i)$ iff $(i, j) \in E$). We estimate P_i using a function of the estimated threat at neighboring vertices as follows:

$$P_i = \begin{cases} \alpha\left(\lambda \mu_i + (1-\lambda) \max_{j \in N(i)} P_j\right), & i \notin \mathcal{T} \\ \pi(i), & i \in \mathcal{T} \end{cases} \qquad (1)$$

where $\mu_i = \frac{\lambda}{|N(i)|} \sum_{j \in N(i)} P_j$, $\alpha \in (0, 1)$ is a dampening factor, and $\lambda \in [0, 1]$ varies the relative contributions of the mean and max terms in the propagation function. In this development, we assume the parameters α and λ are fixed, but in general they could be made to vary depending on some attribute of vertex i.

As a special case of Equation (1), if $\lambda = 1$, then P is simply the eigenvector centrality of the graph. Alternatively, if $\lambda = 0$ then $\forall i \notin \mathcal{T}, P_i = C$, where $C = \max_{i \in \mathcal{T}} \pi(i)$. Philosophically, the mean term in Equation (1) assumes each vertex's attributes are defined by the average of their neighbors' attributes, while the max term assumes they are defined by the most extreme values of the neighbors' attrributes. Practically, λ trades off the impact of high-degree[2] vertices in defining the threat across the graph; when λ is small, individual tip vertices can have dramatic impact on the graph, but when λ is large (i.e., close to one), tip vertices with low degree will have relatively modest effects on the graph.

Computationally, estimates for non-tip vertices can be computed iteratively by updating P_i using estimates of neighboring vertices from the previous iteration. The algorithm stops when the maximum change in estimated threat from one iteration to the next is small—essentially a form of fixed-point iteration. Similar to other techniques in the eigenvector centrality family, convergence is guaranteed with $\lambda = 1$. For a general propagation function, it has not been shown that the proposed technique has a guarantee of convergence. However, in practice, we observed convergence for a wide range of λ over all datasets evaluated.

1. Web pages known as *hubs* are large directories of links to information on a particular topic; pages called *authorities* contain focused information on one topic. A high hub score indicates the page points to many others; a high authority score indicates that many hubs point to it.
2. In graph theory, the degree of a vertex is the number of edges connected to that vertex.

Propagation on Space-Time Graphs

The threat propagation algorithm diffuses estimates of a vertex's foreground membership attribute (i.e., threat) through interactions with its local neighborhood. We have extended this concept to allow a vertex's threat to vary over time. By tracking threat over time, we can determine when a vertex is acting as a member of the foreground, controlling how and when threat propagates through the network. Details of the algorithms presented here were first reported in Smith et al. [3] and Philips et al. [10].

In the scenario depicted in Figure 2, the red vertex interacts with the black vertex at times t_1 and t_2. The blue vertex interacts with the black vertex at time t_3. Given the time-varying threat signatures of the red and blue vertices shown in the accompanying graphs, in order to estimate the threat signature of the black vertex, we must make some assumptions about the dynamic process governing the foreground behavior we wish to detect.

Recalling our definition of a space-time graph $G_t = (V, E_t)$, we will abuse notation by letting E denote a set of triples of the form (i, j, t), where $(i, j, t) \in E$ iff $(i, j) \in E_t$. Furthermore, let $E(i) = \{(j, t) \mid (i, j, t) \in E\}$. Equation (2) generalizes Equation (1) to allow for time-varying probabilities,

$$P_i(t) = \alpha\left(\frac{\lambda}{\mu_i(t)} + (1-\lambda)\max_{e_{ij}\in E(i)} g(t \mid e_{ij})\right), \qquad (2)$$

where $\mu_i(t) = \frac{1}{|E(i)|}\sum_{e_{ij}\in E(i)} g(t \mid e_{ij})$. The function $g(t|e_{ij})$ is an application-specific interaction model describing the effect of an edge between vertex i and j over all time. This function g can naturally be divided into two terms. The first term is a scale factor defined by the probability of threat transferred from j evaluated at the time that the edge is created. The second term is a kernel function defining how the probability of threat changes for times different from the edge creation time,

$$g(t \mid e_{ij}) = P_j(t_{e_{ij}})\, K(t - t_{e_{ij}})\,. \qquad (3)$$

Naturally, the kernel function $K(t)$ must be defined on an application-specific basis because the effect of an edge on community membership may change depending upon the process governing the community one wishes to detect. In the example of a group of collaborating colleagues, it is expected that people in the same community are ones who attend the same meetings at the same time. Therefore, the interaction kernel could be a Gaussian function whose width is defined by the duration of a typical meeting. In a disease-spreading application, a person may not be infectious until 24 to 48 hours after contracting the virus. In this situation, the desired kernel should be centered not at the time of interaction, but sometime after the interaction occurs. As defined in Equation (2), the overall estimated threat on vertex i is a weighted combination of all kernel functions arising from incoming edges. This property provides a smoothly varying function of threat that depends upon interaction times as well as the community membership kernel (as shown in Figure 3). Note that this formulation of community membership does not necessarily state that a vertex is a member of either the foreground or the background at any specific time. Rather, it provides the probability that the vertex is acting as a member of that community at a given time. This is analogous to the role indicator variable Z in "Mixed-Membership Stochastic Blockmodels" by Airoldi et al. [11].

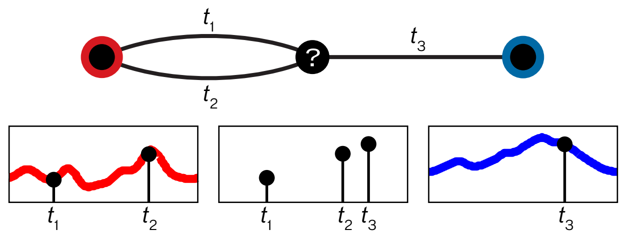


FIGURE 2. Notional example of the dynamic community-detection problem. Given threat over time on the outside red and blue vertices as well as edge times, one can estimate the continuous time-varying probability of threat on the center vertex.

Empirical Results

Membership propagation is evaluated on two simulated datasets as well as on the Enron e-mail dataset [12]. Clauset's *local modularity maximization* [13] and a cued version of Miller's *eigenspace analysis* [1, 2] are used as baselines for performance comparisons against the methods proposed in this article. These three approaches represent a diverse range of cued community-detection techniques, all of which leverage the modularity matrix. Results show that dynamic membership propagation is able to identify community membership better than the other two techniques because of its ability to leverage correlations between edges over time.

METHODOLOGY

Because the methods discussed in this article are local (or cued) methods, performance inherently depends on the tip into the community of interest. Depending on the location of the tip vertex, performance will naturally increase or decrease on the basis of the information contained in the tip. Therefore, detection results are calculated independently by using every possible tip into the foreground. Results are then averaged over all possible tips. This average result is then used to compare detection results across all methods. It is important to note that, unlike other methods, threat propagation is more general with the concept of a tip vertex. Threat propagation can take into account any type of prior information one may have about the network in question. In space-time threat propagation, tip vertices can even vary their probability over time. To create a fair comparison, we only use one tip vertex at a time and keep its value constant over all time ($P_{\text{tip}}(t) = 1$).

Declaration of community membership is carried out by setting a desired threshold on the threat level. In space-time threat propagation, that probability can vary over time. Therefore, for the purpose of making a single declaration on each vertex, the threat on each vertex is averaged over time. The assumption here is that vertices that spend more time acting as members of the community of interest are more likely to be members of that community.

DATA

The performance of each community-detection technique was determined by applying the techniques to three network datasets. A key feature of each dataset is that the communities within each network are defined by the coordinated dynamic interactions between their members and not necessarily by the static topology of the network. For each dataset, a subset of the network is chosen as the community of interest (the foreground) that the community-detection algorithms are tasked with distinguishing from the other remaining vertices (the background).

Recursive matrix data. The first dataset consists of a network topology that was constructed using the R-MAT (named for *recursive matrix*) graph generator [14], which produced a simulated network of 256 vertices and 3045 edges. A 16-vertex subset of the network was selected as the foreground community. A "rumor" was placed on one vertex in the foreground community. At each discrete time step, each foreground vertex communicates with a random subset of its neighbors at rates governed by a Poisson process, propagating the rumor. After a large number of time steps, the rumor will inevitably spread to every member of the foreground community. This foreground community is embedded within the remaining 240 background vertices through the R-MAT generation process. Unlike the foreground community, the background vertices do not interact in a coordinated fashion, but instead interact with neighbors at random. Figure 4a illustrates the topology of the rumor-spreading graph in a force-directed layout, with the foreground vertices shown in red and the background vertices in gray. Note that the foreground community is highly connected to the background vertices.

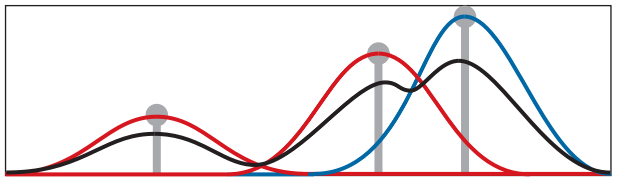


FIGURE 3. Notional result from dynamic membership propagation using a Gaussian kernel function. The black curve is a weighted combination of the red and blue membership kernels as defined in Equations (2) and (3).

Enron e-mail corpus. The second dataset is the Enron e-mail corpus [12], consisting of time-stamped e-mails exchanged between employees at the Enron Corporation. The entire network consists of 156 vertices and 38,390 interactions, where a vertex corresponds to an individual employee and an interaction corresponds to an e-mail sent from one employee to another. The foreground community for this network was chosen to be the 25 employees that the corpus identifies as members of the Enron legal department. The topology of this network is shown in Figure 4b; once again vertices with membership in the foreground community are colored red.

Vehicle movement simulation. The third and final dataset is a simulation of vehicle movement over a 48-hour time period in an urban environment. The simulation

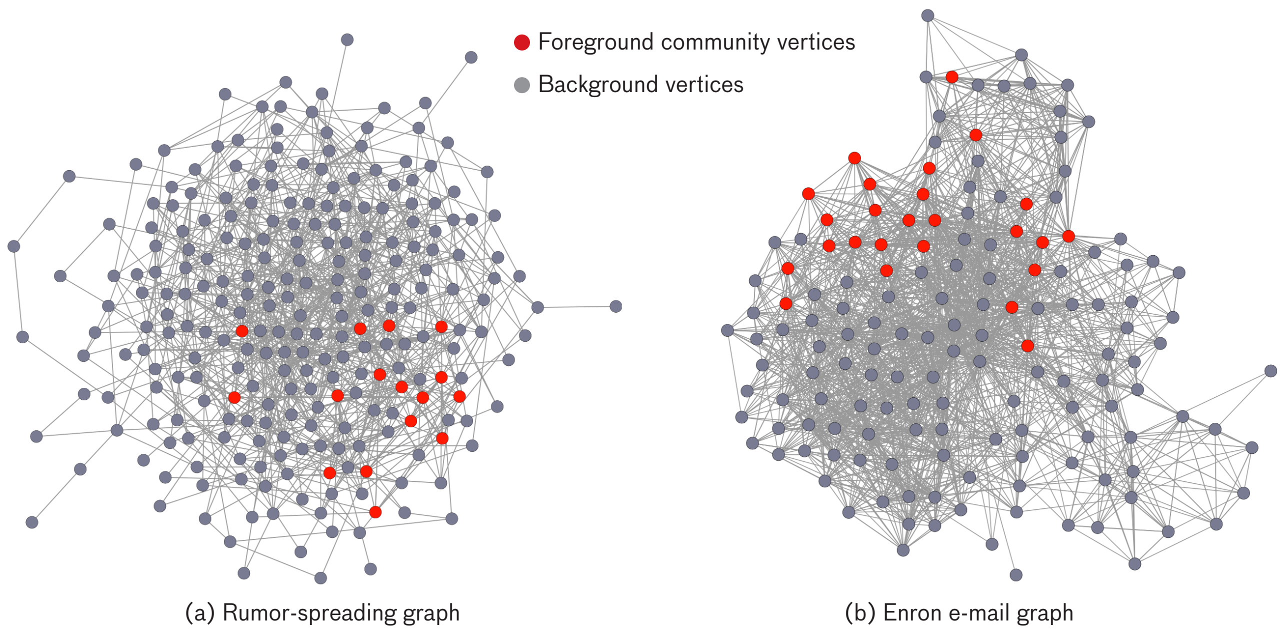


FIGURE 4. Graph representation of the (a) simulated rumor-spreading network and (b) Enron e-mail network datasets used to evaluate community-detection algorithms. Foreground community vertices for each dataset are colored red, and background vertices are colored gray.

was constructed by the National Geospatial-Intelligence Agency (NGA). The vertices in this network correspond to buildings at different locations within the city, and an edge between two vertices exists if a vehicle has traveled between the corresponding buildings. There are approximately 4400 vertices and over 116,000 edges in the network. A small subset of this network corresponds to the operations of an insurgent cell that conducts activities at 31 different vertices over the course of the 48-hour period. Time stamps corresponding to departure and arrival times for each vehicle track allow exploitation of the dynamic properties of the dataset. The topology of this network is not shown here because the size and extensive connectivity of the network render its visualization impractical for this article. For a more detailed discussion of this dataset, see Smith et al. [3].

DETECTION PERFORMANCE

Figure 5 shows detection performance curves for all three networks. Results on the simulated rumor-spreading graph are shown in Figure 5a. Both the eigenspace and local modularity methods are performing near chance. These results are not surprising, given that these methods are both designed to identify tightly connected communities and that the topology of this dataset was specifically designed to be the same for the foreground and the background. Static threat propagation shows detection performance well above chance. This performance increase demonstrates the potential power of a tip vertex even in the absence of static structure. While methods such as local modularity also use a tip vertex, they force a hard decision at every iteration of the algorithm. A bad decision, once made, is compounded as the method proceeds. In contrast, threat propagation passes soft probability estimates at every iteration, postponing a decision until the end. This feature mitigates the effect of any bad decision. Finally, space-time threat propagation shows the best performance of all. This boost above static threat propagation is due to space-time propagation's ability to utilize the correlations between interactions over time.

Figures 5b and 5c show detection performance for the Enron e-mail graph and the simulated vehicle movement graph, respectively. Both plots show similar performance to the previous results, with space-time threat propagation having the best detection performance. Static threat propagation and eigenspace detection performance fall off because of their inability to leverage the dynamic process. In both cases, local modularity initially has a large increase in probability of detection at very low probabilities of false alarm, but at some point in the calculation a wrong decision is made, and performance plateaus.

Network Exploration

Real-world networks can be immense in size or difficult to construct, rendering data collection and processing on the entire network infeasible. Efficient sampling strategies are therefore important in order to collect the most informative parts of the network. An example of a difficult network-exploration problem is the surveillance of hidden HIV populations [16], for which Magnani et al. investigated several sampling strategies, including the well-known "snowball" sampling and respondent-driven sampling. Similarly, for local community detection starting from a set of tip vertices, an efficient sampling strategy explores a small fraction of the complete network while maximizing the end detection performance. Threat propagation offers a natural way for prioritizing network exploration. Intuitively, edges that propagate a higher amount of threat potential should be prioritized. Results on the simulated insurgent network data show that propagation-driven sampling is able to explore a very small fraction of the network while achieving good detection performance.

Exploration of a moving intelligence[3] (MOVINT) graph is a manually intensive process. For example, automatic tracking of cars in dense urban environments is subject to many types of errors. Establishing connections between two locations requires a semiautomated approach with a human in the loop to correct potential tracking errors. The resulting workflow is a cued graph exploration approach whereby a location of interest is identified (a cue) and a graph is grown beginning at the cue location. This cue location is assumed to be known a priori and represents a site used by the foreground community. Beginning with a cue allows analysts to focus their attention on vehicles in a local region (in graph space) around a known foreground location.

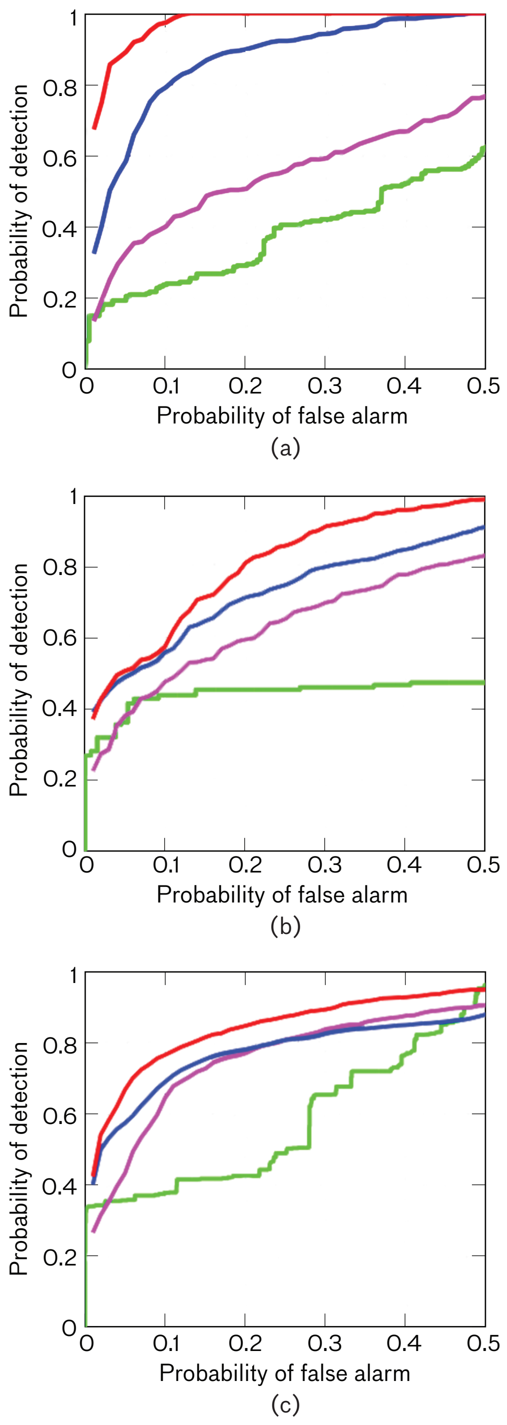


FIGURE 5. Community-detection results on the (a) rumor-spreading graph, (b) Enron e-mail graph, and (c) vehicle movement graph. Plots compare community-detection performance on a variety of algorithms including eigenspace detection (magenta), local modularity maximization (green), threat propagation (blue), and space-time threat propagation (red). Each of these graphs is a receiver operating characteristic, or ROC, curve, relating the probability of detection to the probability of false alarms.

3. Moving intelligence refers to the knowledge gained from the tracking of moving objects on land or sea.

Exploration Strategies

BREADTH-FIRST SEARCH

The breadth-first search (BFS) algorithm [17] provides a reasonable initial approach for exploration. In BFS, a graph is initially formed by following all vehicles departing or arriving at the cue location (vertex). Then for each location (vertex) found in step one, all vehicles are followed again. The priority assigned to exploring edge *E* in the graph is therefore given by

$$\text{Priority}\,(E) = \min\, d(v(E), V_c)\,,$$

where *d* is the standard graph distance between vertices, *v*(*E*) are the vertices of edge *E*, and V_c is the cued vertex. This procedure is repeated until some fixed number of tracks are explored. Vertices at the same distance away from the cue are explored in random order, and vehicles departing any given vertex are followed in random order.

Figure 6 shows three graphs at various stages of exploration using BFS on the simulated insurgent network data. Note that while the vehicles may traverse long distances in physical space, all locations discovered are within one, two, or three hops from the cue vertex.

DEGREE-WEIGHTED BREADTH-FIRST SEARCH

While BFS is good at exploring a neighborhood of vertices surrounding a cue vertex, the graph in Figure 6c demonstrates a major drawback of this approach. The vast majority of an analyst's time spent on a graph such as in Figure 6c is devoted to exploring tracks leaving a handful of high-degree vertices. Under a fixed time constraint, this would not be an efficient use of human resources. This observation that BFS can be biased toward high-degree vertices has also been observed in a number of other studies [18, 19].

In order to combat the bias toward high-degree vertices, a degree-weighted BFS approach is implemented. In degree-weighted BFS, vertices at the same distance from a cue vertex are explored in order of their degree, with low-degree vertices explored first and high-degree vertices explored last. Additional models of vertex relevancy may also be used, depending upon the observable information available for each vertex and track. Because vertices represent clustered track destinations, it is possible to estimate the degree of each vertex by counting the number of destinations in each cluster, thereby providing an estimate of vertex degree before the exploration stage. Degree-weighted BFS improves upon the regular BFS but still does not fully capture the essence of threat discovery.

PROPAGATION-DRIVEN SAMPLING

Threat propagation, described in the network detection section, provides a natural measure for prioritization. Intuitively, vehicle tracks arriving or departing locations of higher estimated threats at that specific time are of higher interest. Prioritization based on the estimated threat performs a propagation-driven sampling on the network as vertices with higher threat are expanded first and their threat propagated. This exploration strategy is tailored specifically to the objective of network discovery.

Exploration Performance

The proposed exploration strategies have been evaluated on the simulated insurgent network data by using two metrics. One metric measures time required to explore the graph (human resources), and another measures efficiency at uncovering the foreground relative to the background. Three search strategies—propagation-driven sampling, degree-weighted BFS, and standard BFS—are compared using each metric. The dataset used in the analysis is the previously discussed NGA dataset on vehicle movements, and the results represent averages across multiple independent runs, each seeded with a different tip into the foreground network.

Figure 7 shows the percentage of the foreground network found as a function of the number of tracks examined, which in general is proportional to the amount of human time required to uncover a certain percentage of the foreground network. Figure 7 shows that for a fixed percentage of the foreground network, space-time threat propagation is able to achieve a fixed percentage in approximately one-third the time of standard BFS, and half the time of even the degree-weighted BFS.

Figure 8 shows the average portion of foreground network found against the average portion of the background network found. While this plot is similar to a traditional receiver operating characteristic (ROC) curve indicating probability-of-detection and probability-of-false-alarm performance, there are subtle methodological differences. This figure represents, given a fixed number of vertices that have been explored, how much of the foreground and background networks have been uncovered, and is not immediately comparable to expected optimal detection performance, such as what

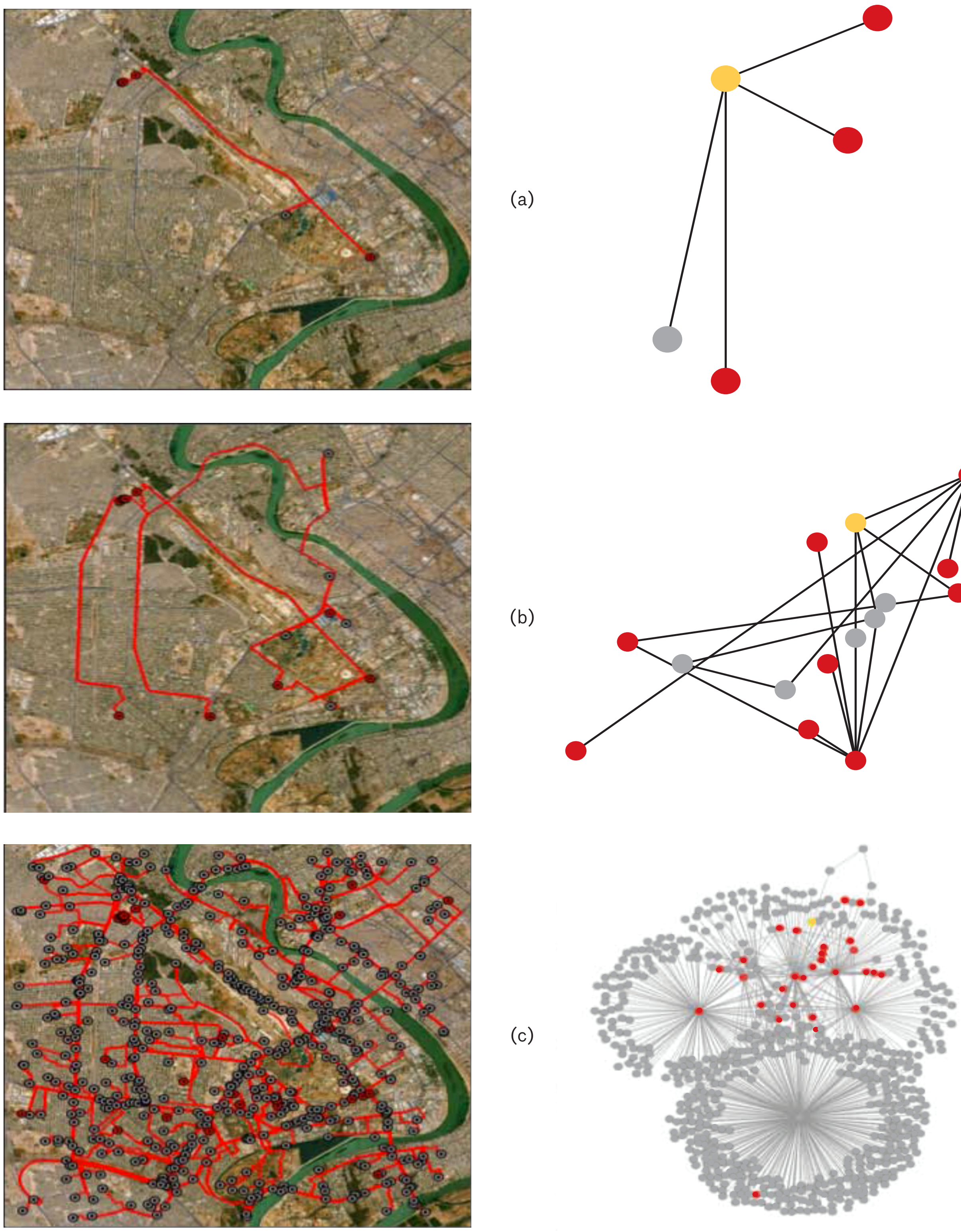


FIGURE 6. Graphs constructed by following vehicles from one location to another. The left figures show vehicle movement overlaid on the aerial imagery while the right figures show the same movement represented as graphs. From top to bottom, each figure shows the graph at various stages of exploration using breadth-first search. The foreground subgraph is shown using red vertices, and the background graph is shown using gray vertices. The cue vertex is shown in yellow. The left-hand images are from ©DigitalGlobe.

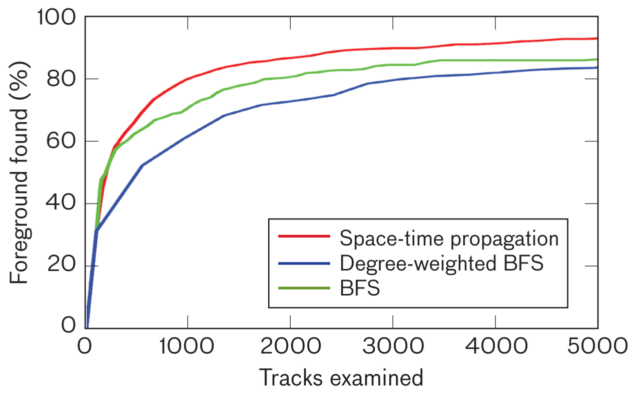


FIGURE 7. Comparison of operator exploration efficiency of space-time threat propagation (red), degree-weighted breadth-first search (BFS) (blue), and standard BFS (green). Propagation-based exploration uncovers the network approximately twice as fast as either of the breadth-first search methods.

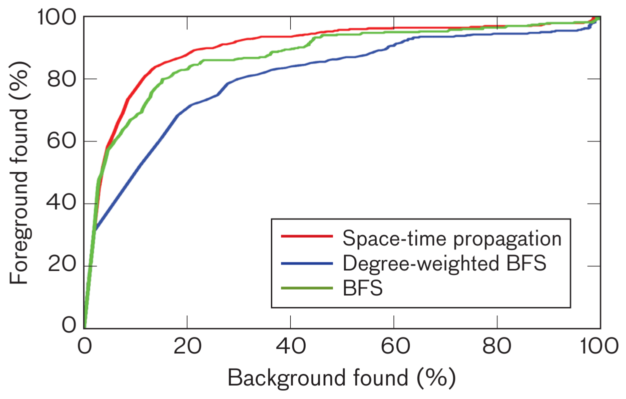


FIGURE 8. Comparison of exploration outcomes of space-time threat propagation (red), degree-weighted breadth first search (BFS) (blue), and standard BFS (green). For a given operating point (i.e., number of vertices explored), propagation-based exploration generally uncovers more threatening vertices than either of the breadth-first methods.

is shown in Figure 5c. In general, after a fixed number of vertices have been explored, the expected relative proportions of foreground and background vertices uncovered are maximized by using a propagation-based exploration method, relative to either of the BFS approaches.

Intelligence, Surveillance, and Reconnaissance Network-Discovery Tool

Software components for effective data visualization, exploration, and graph construction are critical tools for analysts, particularly when the raw data are voluminous and originate from diverse sensor types. In a typical network-discovery task, a team of analysts might be tasked with using wide-area motion imagery (WAMI) data to construct a network in which vertices are geospatial locations (sites) and edges are formed by vehicle tracks that originate from one location and terminate at another. To build this type of graph, the analysts are given a tip location from which to begin their analysis. They then monitor the site over several hours or perhaps even days of WAMI video in order to identify vehicles arriving at or departing from the location. Those vehicles are then tracked back to their points of origin or destination, which then become new vertices in the network. The process repeats, starting from the newly discovered vertices.

Although this network construction workflow is simple and effective, it is time-consuming and manually intensive. Depending on the size of the dataset and the level of activity at the locations of interest, analysts may spend large fractions of their time on tedious tasks; they may need to view several minutes of video simply to find a vehicle to track, and then spend several more minutes tracking the vehicle frame by frame. Furthermore, analysts typically require multiple software applications for exploitation and dissemination. For example, an analyst might use one tool to view the WAMI data and track vehicles, another to build and visualize a graph, and yet others to display and communicate their findings to their collaborating analysts. Many software packages excel at those tasks individually, but none are optimized for the workflows inherent to the collaborative network-discovery task.

BlueStreak Exploitation Tool

To address this problem of a labor-intensive process that requires multiple tools, we developed a software exploitation system named BlueStreak that supports data visualization, graph construction, algorithm services, and analyst collaboration within the network-discovery framework. The system, shown in Figure 9, is composed of analyst user interfaces (clients) that are connected via a collaboration server. Additional modular services, such as data-retrieval services, tracking services, or graph-exploitation algorithms, can be “plugged in” to the collaboration infrastructure as desired to provide additional functionality. Consequently, the clients’ main responsibility is data visualization, and heavy computation is carried out on the servers.

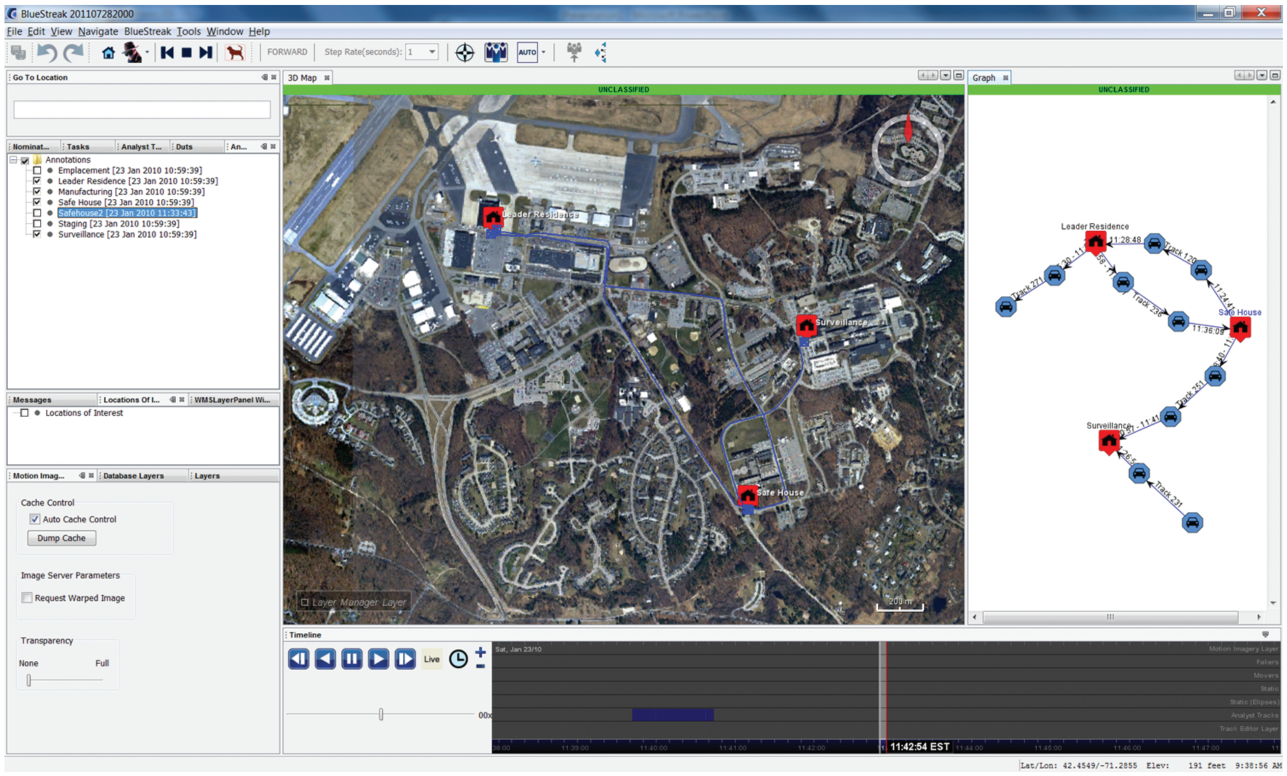

BlueStreak 201107282000
File Edit View Navigate BlueStreak Tools Window Help
Go To Location
3D Map
Graph
UNCLASSIFIED
Annotations
Emplacement [23 Jan 2010 10:59:39]
Leader Residence [23 Jan 2010 10:59:39]
Manufacturing [23 Jan 2010 10:59:39]
Safe House [23 Jan 2010 10:59:39]
Safehouse2 [23 Jan 2010 11:33:43]
Staging [23 Jan 2010 10:59:39]
Surveillance [23 Jan 2010 10:59:39]
Locations of Interest
Cache Control
Auto Cache Control
Dump Cache
Image Server Parameters
Request Warped Image
Transparency
None
Full
Leader Residence
Surveillance
Safe House
Timeline
11:42:54 EST

FIGURE 9. Screenshot of the BlueStreak client. The largest windows in the tool represent the map viewer (for displaying imagery and geospatial data) and the integrated graph viewer.

The BlueStreak client serves as the user interface and is built using the NetBeans Rich Client Platform (RCP) and other open-source components. The geospatial visualization component of the tool is based on the NASA World Wind Java SDK (software development kit). The tool also features a built-in graph construction capability using the Java Universal Network/Graph (JUNG) framework.

The BlueStreak collaboration server runs on Apache Tomcat and uses an in-house application framework called Maestro that allows deployment of modular services into the system. The clients and the servers communicate over a publish/subscribe channel, which is implemented using the Apache ActiveMQ message broker. This channel is shared among all clients, the collaboration server, and other services to synchronize information among them.

This service-oriented architecture offers several practical advantages. Because the major computational work is performed on the server side, the clients do not have to incorporate or even be physically close to a data storage and processing infrastructure. Data are made available as services, and users request data "on demand" from those services as they require the data. If new data sources become available and are exposed as services, the clients can be configured to ingest and display them. In this way, several modes of intelligence are able to be displayed simultaneously in the client. In addition, the modular "plug-and-play" nature of the system provides easy deployment of new capabilities and updating of older ones.

Multi-INT Workflow

With the BlueStreak tool, the operator's network-discovery workflow, shown in Figure 10, is enhanced. As in the manual workflow, the analyst begins from a tip site. However, instead of manually tracking movers to construct the network, the user has the ability to nominate a space-time region around the site of interest. This nomination is converted into "region-tracking" requests, which are then sent over the publish/subscribe channel to an on-demand tracking service. Tracking jobs are carried out on high-performance computing resources, and any tracks that begin or terminate within the space-time region are displayed to the user. These tracks are automatically extended by *entity trackers* that attempt to track the vehicle to its point of origin or destination. Automatic trackers are occasionally prone to swaps, breaks, and other errors, so the client includes two track-repair tools that enable the user to throw out erroneous track points and restart an automatic tracker on the appropriate vehicle. As a failsafe measure, the analyst can at any point seamlessly revert back to a more manual tracking mode, in which mouse clicks on the image data generate and extend tracks.

Once the track is complete, the analyst can annotate a new location of interest at its endpoint. This new annotation automatically appears as a vertex in the graph component of the BlueStreak client. When the user performs a link action to connect the track to a location, the track automatically appears as an edge in the graph viewer. Thus, graph construction proceeds as a natural extension of the tracking workflow.

When vertices and edges are created, update messages are sent over the publish/subscribe channel to a graph algorithm service. This service uses the space-time threat propagation algorithm to propagate threat from the tip site(s) to other vertices in the graph. A list of sites ordered by threat level is presented on a separate window in the BlueStreak client in order to help the analysts prioritize their efforts to sites with the highest threat level. Tracks are also prioritized according to the level of threat that they carry, permitting users to select the next highest threatening track as their next analysis task. This feature is particularly useful when several tracks are generated from a region-tracking request, and the user must decide which of those tracks to follow first.

The BlueStreak collaboration server plays a critical role in the workflow. The service-oriented architecture allows many clients to access the same data and same graph simultaneously; therefore, multiple analysts may build the same graph at the same time. Each connected user can see the tracks, graph vertices, and graph edges generated by other analysts, so dissemination of the analysts' work occurs organically. However, the danger of such an arrangement is that multiple analysts follow the same track, duplicating effort and wasting valuable time. To mitigate the chances of such an event, the collaboration server maintains a *check-in, check-out* system for tracking tasks. When a region-tracking request returns tracks to the client, each track is

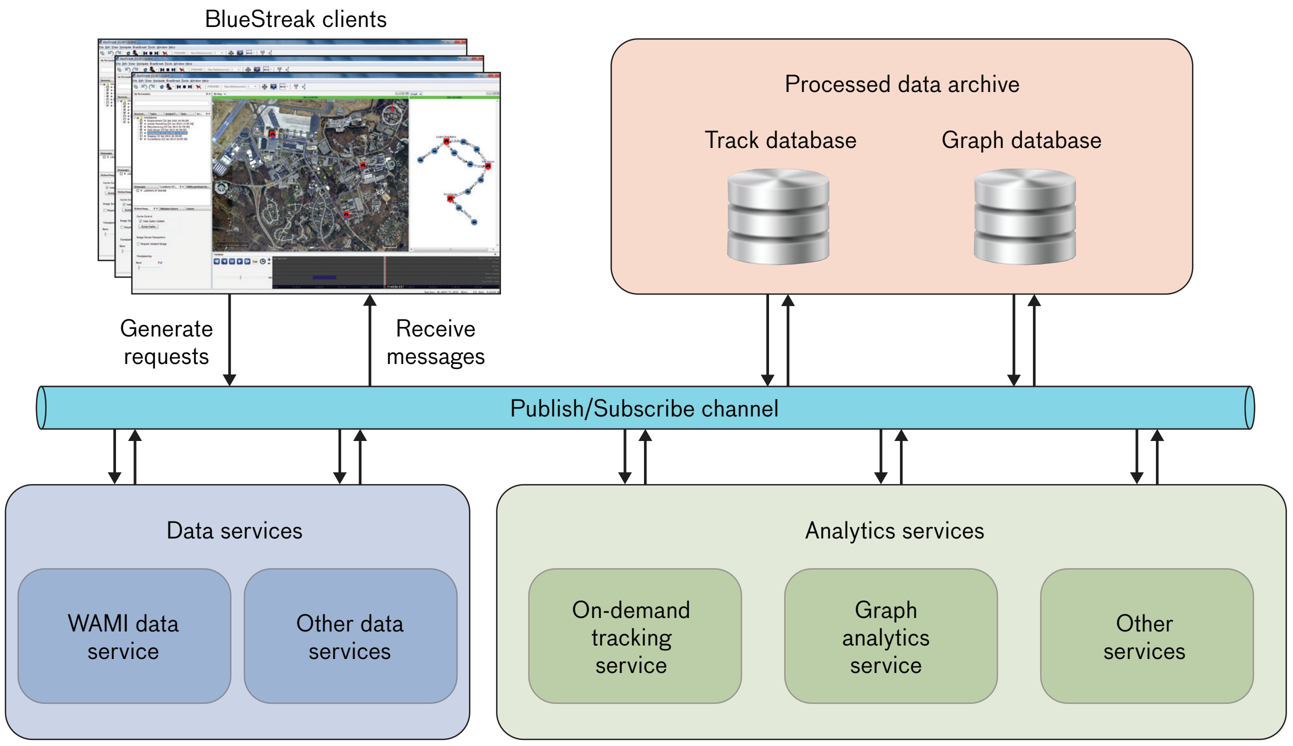


FIGURE 10. Schematic of the BlueStreak architecture. The publish/subscribe channel serves as the messaging backbone between the clients and various services. The system accommodates data services, algorithm services, and storage services.

assigned a unique task identification. The users can check out a particular task, and the checked-out status becomes visible to other analysts. When the tracking task is complete, the user who checked it out marks it as such. If this system is used optimally, the chance that effort is duplicated becomes extremely remote.

Future Directions

The methodology described in this article addresses the complementary problems of cued network discovery and exploration. Given a sequence of dynamic point-to-point connections and a set of known points of interest, the dynamic threat propagation algorithm can infer where to look for additional connections, as well as which currently known points are of primary interest. The dynamic threat propagation algorithm has been shown to be effective for network discovery and exploration on a diverse collection of datasets, including the Enron e-mail corpus, a set of vehicle tracks, and an artificially constructed rumor-spreading graph.

Future enhancements to the algorithm include (1) improving the process by which the temporal kernel is selected and (2) automatically tuning the Bayesian diffusion model. In the current implementation, the temporal kernel was selected by the analyst to provide optimal performance. Also, the kernel is identical for all connection events. In the future, the threat propagation kernel will be automatically chosen and tuned to the particular statistical characteristics of the connection event.

A second area of potential improvement is the extension of the diffusion model to account for constrained paths of diffusion. Currently, all temporal diffusion paths are treated equally by the algorithm; however, some types of multi-INT information may restrict or constrain the threat diffusion paths. Incorporating these types of information into the multi-INT algorithm will broaden the applicability of the method and improve its inferential ability. ■

References

1. B. Miller, M. Beard, and N. Bliss, "Eigenspace Analysis for Threat Detection in Social Networks," *Proceedings of the 14th International Conference on Information Fusion (FUSION),* 2011.
2. B. Miller, N. Bliss, and P. Wolfe, "Subgraph Detection Using Eigenvector L_1 Norms," *Proceedings of the 2010 Neural Information Processing Systems (NIPS) Conference,* pp. 1633–1641.
3. S.T. Smith, A. Silberfarb, S. Philips, E.K. Kao, and C. Anderson, "Network Discovery Using Wide-Area Surveillance Data," *Proceedings of the 14th International Conference on Information Fusion (FUSION),* 2011.
4. R.R. Coifman, S. Lafon, A.B. Lee, M. Maggioni, F. Warner, and S. Zucker, "Geometric Diffusions as a Tool for Harmonic Analysis and Structure Definition of Data: Diffusion Maps," *Proceedings of the National Academy of Sciences,* vol. 102, no. 21, 2005, pp. 7426–7431.
5. R.I. Kondor and J. Lafferty, "Diffusion Kernels on Graphs and Other Discrete Structures," *Proceedings of the 19th International Conference on Machine Learning,* 2002, pp. 315–322.
6. S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Proceedings of the 7th International Conference on World Wide Web (WWW),* vol. 30, no. 1–7, 1998, pp. 107–117.
7. S. Smith, K. Senne, S. Philips, E. Kao, and G. Bernstein, "Covert Network Detection," *Lincoln Laboratory Journal,* vol. 20, no. 1, 2013, pp. 47–61.
8. P. Bonacich, "Power and Centrality: A Family of Measures," *American Journal of Sociology,* vol. 92, no. 5, 1987, pp. 1170–1182.
9. J. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *Journal of the ACM,* vol. 46, no. 5, 1999, pp. 604–632.
10. S. Philips, E.K. Kao, M. Yee, and C. Anderson, "Detecting Activity-Based Communities Using Dynamic Membership Propagation," *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing,* pp. 2085–2088.
11. E. Airoldi, D. Blei, S. Fienberg, and E. Xing, "Mixed Membership Stochastic Blockmodels," *Journal of Machine Learning Research,* vol. 9, 2008, pp. 1981–2014.
12. W. Cohen, "Enron E-mail Dataset," http://www.cs.cmu.edu/~enron/, 2009.
13. A. Clauset, "Finding Local Community Structure in Networks," *Physical Review E,* vol. 72, 026132, 2005.
14. D. Chakrabarti, Y. Zhan, and C. Faloutsos, "RMAT: A Recursive Model for Graph Mining," SIAM International Conference on Data Mining, 2004.
15. P. Perry and P. Wolfe, "Point Process Modeling for Directed Interaction Networks," arXiv:1011.1703v1 [stat.ME], 2010.
16. R. Magnani, K. Sabin, T. Saidel, and D. Heckathorn, "Review of Sampling Hard-to-Reach and Hidden Populations for HIV Surveillance," *AIDS,* vol. 19, suppl 2:S, 2005, pp. 67–72.
17. M.E.J. Newman, *Networks: An Introduction.* Oxford, UK: Oxford University Press, 2010.
18. L. Becchetti, C. Castillo, D. Donato, and A. Fazzone, "A Comparison of Sampling Techniques for Web Graph Characterization," *Proceedings of the Workshop on Link Analysis (LinkKDD),* 2006.
19. M. Kurant, A. Markopoulou, and P. Thiran, "On the Bias of BFS (Breadth First Search)," *Proceedings of the 22nd International Teletraffic Congress (ITC),* 2010.

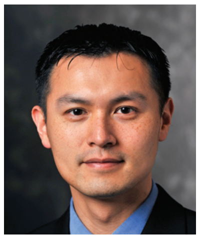
About the Authors

Michael J. Yee is a a member of the technical staff in the Intelligence and Decision Technologies Group. Since joining the Laboratory in 2006, his research activities have included information retrieval, graph visualization, and text analytics. He received a bachelor's degree in math and computer science from Gordon College and master's and doctoral degrees in operations research from MIT, where his research focused on inferring decision-making heuristics. He interned at Lincoln Laboratory for a summer, investigating techniques for visualizing and analyzing social networks.

Scott Philips is currently a data scientist at Palantir Technologies. Before joining Palantir, Scott spent five years as a member of the technical staff in the Intelligence and Decision Technologies Group. While he was at Lincoln Laboratory, his research focused on developing statistical algorithms for the exploitation of data from intelligence, surveillance, and reconnaissance sensors. Scott received his doctoral degree in electrical engineering in 2007 from the University of Washington, where his research focused on signal processing and machine learning algorithms for the detection and classification of sonar signals.

Gary R. Condon is the associate leader of the Intelligence and Decision Technologies Group. His research focuses on the applications of machine learning and human workflow optimization to improve the exploitation of sensor data to support military and intelligence decision making. Since joining Lincoln Laboratory in 1999, he has led the development of novel analytic tools and techniques for intelligence users, participating in the design and demonstration of advanced battle management capabilities for conventional and Special Operations Forces, and supported the architectural development and operational evaluation of various intelligence, surveillance, and reconnaissance (ISR) systems. From 2011 to 2013, he served as the Science & Technology Advisor to the ISR Task Force in the Office of the Under Secretary of Defense for Intelligence, where he focused on quick-reaction capabilities in support of Operation Enduring Freedom in Afghanistan. He earned a bachelor's degree in astronomy from the University of Massachusetts in Amherst, where he studied the formation of stars and planets, and a doctoral degree in physics from the University of New Mexico, where he developed the photometric field-emission electron microscope.

Peter B. Jones is a member of the technical staff in the Intelligence and Decision Technologies Group. He joined Lincoln Laboratory in 2002 after receiving a bachelor's degree in electrical engineering from Brigham Young University. He subsequently received the master's and doctoral degrees in electrical engineering from MIT through the Lincoln Scholars program in 2005 and 2011, respectively. His research focus is on applying information and decision theory to the estimation and control of graph-based processes.



Edward K. Kao is a Lincoln Scholar in the Intelligence and Decision Technologies Group. Since joining the Laboratory in 2008, he has been working on graph-based intelligence, in which actionable intelligence is inferred from interactions and relationships between entities. Applications include wide-area surveillance, threat network detection, homeland security, and cyber warfare. In 2011, he entered the doctoral program at Harvard University in the statistics department. Current research topics include causal inference on peer influence effects, statistical models for community membership estimation, information content in network inference, and optimal sampling and experimental design for network inference.

Steven T. Smith is a senior staff member of the Intelligence and Decision Technologies Group with many distinguished accomplishments in radar, sonar, and systems concept development. His contributions span all aspects of signal processing, from data modeling and measurement, to novel signal detection and estimation algorithms, to target tracking. Professional and career highlights involve novel solutions and analysis in new problems, such as geometric optimization for signal processing, statistical resolution limits, bounds for nonlinear parameter estimation, and optimum network detection. He is the recipient of outstanding paper awards from the IEEE and SIAM. He received his bachelor's degree in electrical engineering and mathematics from the University of British Columbia, Vancouver, and his doctoral degree in applied mathematics from Harvard University.

Christian C. Anderson is a member of the technical staff in the Intelligence and Decision Technologies Group. He joined Lincoln Laboratory in 2010 after receiving a doctoral degree in physics from Washington University in St. Louis, where his primary research interests involved solving inverse problems related to ultrasonic imaging and tissue characterization. His focus since joining the Laboratory has been on the development of multi-intelligence exploitation algorithms and architectures, including fusion, graph analytics, and Red/Blue exercises applied to intelligence, surveillance, and reconnaissance scenarios.

Frederick R. Waugh is a member of the technical staff in the Intelligence and Decision Technologies Group. His work at Lincoln Laboratory has included algorithms for and design of processing, exploitation, and dissemination software systems; systems analysis of counterterrorism technologies; planning and managing of complex multisensor test exercises; and algorithm development for active imaging sensors. Prior to joining the Laboratory in 2002, he worked on remote sensing and image processing at Photon Research Associates in San Diego, California. He received a bachelor's degree in physics from Princeton University in 1986 and a doctoral degree in physics from Harvard University in 1994, with thesis research on neural networks and quantum confinement in nanostructures.