2013 IEEE 13th International Conference on Data Mining

Dynamic Pattern Detection with Temporal Consistency and Connectivity Constraints

Skyler Speakman, Yating Zhang, Daniel B. Neill*
Event and Pattern Detection Laboratory
Carnegie Mellon University, Pittsburgh PA 15213
*Email: neill@cs.cmu.edu**

***Abstract*—We explore scalable and accurate *dynamic* pattern detection methods in graph-based data sets. We apply our proposed *Dynamic Subset Scan* method to the task of detecting, tracking, and source-tracing contaminant plumes spreading through a water distribution system equipped with noisy, binary sensors. While static patterns affect the same subset of data over a period of time, dynamic patterns may affect different subsets of the data at each time step. These dynamic patterns require a new approach to define and optimize penalized likelihood ratio statistics in the subset scan framework, as well as new computational techniques that scale to large, real-world networks. To address the first concern, we develop new subset scan methods that allow the detected subset of nodes to change over time, while incorporating *temporal consistency constraints* to reward patterns that do not dramatically change between adjacent time steps. Second, our *Additive GraphScan* algorithm allows our novel scan statistic to process small graphs (500 nodes) in 4.1 seconds on average while maintaining an approximation ratio over 99% compared to an exact optimization method, and to scale to large graphs with over 12,000 nodes in 30 minutes on average. Evaluation results across multiple detection, tracking, and source-tracing tasks demonstrate substantial performance gains achieved by the Dynamic Subset Scan approach.**

***Keywords*-sensor fusion; likelihood ratio statistics; spatial and subset scan statistics; water distribution systems**

I. Introduction and Background

Detecting patterns in massive data sets has multiple real-world applications in fields such as public health, law enforcement, and security. The “subset scan” approach to pattern detection treats the problem as a search over subsets of data, with the goal of finding anomalous subsets. This approach poses two main challenges: appropriately evaluating the “anomalousness” of a given subset, and the computational issue of searching through the exponentially many subsets of the data. Previous approaches ([8], [12], [10], [11]) have addressed the first concern by “scoring” each subset using likelihood ratio statistics such as the expectation-based Poisson (EBP) ([12], [10]) or expectation-based binomial (EBB) [8] scan statistics. Our current work allows for a more sophisticated scoring function by *penalizing* the likelihood ratio statistic, considering additional prior information from each data element. The penalized likelihood ratio typically does not satisfy useful properties such as “linear-time subset scanning” [11], making efficient optimization over subsets a challenging task. However, we demonstrate that the EBB likelihood ratio statistic can be written as an additive function, enabling efficient optimization of a penalized version of that statistic over subgraphs. The first major contribution of our current work is the development of *temporal consistency constraints* which allow for additional penalties or rewards to act on the scoring function, rewarding spatial subsets that are temporally consistent with each other, and efficient optimization of the resulting, penalized scan statistic to detect dynamic clusters subject to these constraints. Our second major contribution is the *Additive GraphScan* algorithm, which efficiently identifies anomalous (high scoring), connected subgraphs, thus incorporating both (hard) connectivity constraints and (soft) temporal consistency constraints. Each of these contributions will be discussed in detail below.

Many complex data sets containing emerging events or patterns are commonly represented in a known and fixed graph structure. Examples of this include water pipelines, transportation routes, power grids, and supply chains in general. While other recent work ([9], [5]) has focused on learning graph structure, here we assume a given graph structure and wish to detect which nodes are currently affected, by observing data produced at the nodes of the graph on each time step.

Our motivating example comes from the field of public health: we focus on detecting, tracking, and source-tracing contaminant plumes in a water distribution system. Creating sensor networks for detecting deliberate or accidental contamination of these systems has been a popular research domain following the terror attacks of September 11, 2001. The “Battle of the Water Sensor Networks” (BWSN) [1] provided real-world data to teams tasked with placing *perfect* sensors to quickly detect contaminants and limit the amount of contaminated water consumed by the population. The placement problem is an interesting one explored further in ([3], [7]). Our current work focuses on the complementary problem of fusing data collected from *noisy* sensors assuming a given placement. Sensor fusion attempts to combine data from multiple distributed sensors in order to increase the detection power of the entire network [14].

We proceed by modeling simple, binary sensors at each pipe junction (graph node) in the system. We assume that a fixed false positive rate (e.g., FPR = 0.1) and true positive rate (e.g., TPR = 0.9) are known and that each sensor operates independently of the others in the network. Our simulations use the network structure and plumes provided in the BWSN data to generate sensor readings over the course of 12 one-hour intervals.

Our task is then: Given (1) the graph structure (pipe network), (2) false positive and true positive rates of the sensors, and (3) independent observations from the sensors over time, we must provide: (A) whether or not a contaminant is present in the system for each one-hour time step, (B) hour-to-hour tracking of which pipe nodes have been affected by the plume on the current and recent past time steps, and (C) source-tracing to determine which node(s) in the system spawned the contaminant. Corresponding evaluation metrics include: (A) average time (in hours) to plume detection as a function of false positive rate (number of false alarms per month), (B) spatial-temporal overlap coefficient between the true and detected subsets of nodes over time, and (C) spatial overlap coefficient between the true and identified subsets of source nodes.

This is not the first work to apply spatial or subset scan statistics to contamination early warning systems. Koch and McKenna [6] used Kulldorff's spatial scan [8] to detect statistically significant circular clusters of anomalous activity. They used properties of the pipe network to create a distance metric based on travel time between sensing nodes in order to define their "circles". However, they were not able to enforce connectivity constraints and take advantage of the topology of the network. Through our Additive GraphScan algorithm, we are able to search over connected subsets of the pipe network to find anomalous connected subgraphs. Berry et al. [2] have also considered the detection power of a network of imperfect sensors, showing that it is worth deploying a sensor network even when individual sensors have low detection probability. However, their experiments did not allow for sensors with false positives, making the detection and source tracing problems much easier to solve as compared to the more difficult scenario we consider here.

Spatial scan statistics attempt to identify regions of interest or "hot spots". This is achieved by maximizing a scoring function $F(S)$, typically defined as the likelihood ratio $F(S) = \frac{\Pr(\text{Data} \mid H_1(S))}{\Pr(\text{Data} \mid H_0)}$, over spatial regions S. In this expression $H_1(S)$ assumes increased activity in region S, and H_0 assumes regular behavior. In this work, we will be monitoring binary sensors s_i each producing $c_i \sim$ Bernoulli (FPR) "triggers" under H_0 or $c_i \sim$ Bernoulli (TPR) "triggers" for $H_1(S)$ containing node s_i. This makes the expectation-based binomial scan statistic [8] a logical choice.

Spatial-*temporal* scan statistics incorporate the time dimension. It is standard to aggregate this temporal information over a time window w so that $c_i = \sum_{t=1 \ldots w} c_i^t$. Once the temporal information has been aggregated for each window $w = 1 \ldots W$, maximizing the spatial-temporal scan statistic for that window proceeds identically to the regular spatial scan statistic; we then maximize over all window sizes from 1 to W. However, an inherent assumption in this aggregation of temporal information is that the affected spatial-temporal subset *does not change* over time. Therefore, we will refer to this approach as the *Static* scan method throughout this text.

We wish to relax this strong assumption on the spatial-temporal structure in order to increase the power to detect *dynamic* patterns that change the affected region over time. One simple approach is to optimize each of the w time steps independently. This allows for each time step t to identify an entirely different spatial region, but does not allow the sharing of information between time steps, possibly reducing detection power. We refer to this approach as the *Independent* scan method throughout this text.

As a compromise between Static and Independent methods, we propose the *Dynamic Subset Scan* which enforces temporal consistency constraints to allow temporally adjacent time steps to share information forward and backward in time. We demonstrate below that this flexibility increases power to detect and track dynamic patterns while scaling to the size of real-world networks.

The rest of the paper is laid out as follows. Section II introduces temporal consistency constraints, applied to the expectation-based binomial scoring function, and demonstrates how the penalized scan statistic can be efficiently optimized over (not necessarily connected) dynamic subsets. Section III explains the Additive GraphScan algorithm, which efficiently identifies high-scoring, *connected* subsets of data with an underlying graph structure. Section IV provides empirical results of our simulations of the water distribution network from the BWSN, comparing our new Dynamic Subset Scan approach to several other approaches, and demonstrating improvements in detection, tracking, and source-tracing performance. Finally, Section V concludes the paper.

II. Temporal Consistency Constraints

This section has four sequential objectives. First, we demonstrate how the expectation-based binomial (EBB) scoring function may incorporate additional constraints while remaining straightforward to optimize over all possible subsets S (i.e., we show that EBB can be written as an additive function over the data elements $s_i \in S$). Second, we show that these constraints may be interpreted as the prior log-odds for a given node s_i to be in the detected subset. We then provide the formal definition of our temporal consistency constraints based on a probabilistic generative model that incorporates both forward and backward temporal consistency. Lastly, we describe the iterative optimization process that "lines up" the spatial-temporal region according to the provided temporal consistency constraints.

A. Additive Scoring Function and Additional Terms

We demonstrate that, conditioned on the false and true positive rates (FPR, TPR) of the sensors, the EBB statistic can be written as an additive set function over the data elements $s_i \in S$. This is an important feature for two reasons. First, additive functions are easy to optimize over all possible subsets. Without connectivity constraints, the optimization process is simply including all records making a positive contribution and excluding the rest. Determining the "most positive" *connected* subset is more complicated, and is covered in the Additive GraphScan section below. Second, additive functions allow for additional penalty terms Δ_i to be included at the element level while the total *penalized* scoring function remains additive and thus amenable to efficient optimization.

Theorem 1: The expectation-based binomial statistic may be written as $F(S) = \sum_{s_i \in S} \lambda_i$, where λ_i depends only on the binary sensor response c_i for sensor s_i (i.e., whether that sensor triggers or not) as well as the false and true positive rates of the sensors in general.

Proof: The log-likelihood ratio form of the EBB scan statistic can be written as follows:

$$\begin{aligned}F(S) &= \log \frac{\Pr(\text{Data}|H_1(S))}{\Pr(\text{Data}|H_0)}\\ &= \log \frac{\prod_{s_i \in S} \Pr(c_i \sim \text{Bernoulli(TPR)})}{\prod_{s_i \in S} \Pr(c_i \sim \text{Bernoulli(FPR)})}\\ &= \log \prod_{s_i \in S} \frac{(\text{TPR})^{c_i}(1-\text{TPR})^{1-c_i}}{(\text{FPR})^{c_i}(1-\text{FPR})^{1-c_i}}\\ &= \sum_{s_i \in S} \left[c_i \log\left(\frac{\text{TPR}}{\text{FPR}}\right) + (1-c_i)\log\left(\frac{1-\text{TPR}}{1-\text{FPR}}\right)\right]\end{aligned}$$

Then $\lambda_i = c_i \log\left(\frac{\text{TPR}}{\text{FPR}}\right) + (1-c_i)\log\left(\frac{1-\text{TPR}}{1-\text{FPR}}\right)$. ■

Next, if we assume a bonus or penalty Δ_i for each $s_i \in S$, then we can easily incorporate these terms into the score function. We define:

$$F_{pen}(S) = F(S) + \sum_{s_i \in S} \Delta_i = \sum_{s_i \in S} (\lambda_i + \Delta_i).$$

$F_{pen}(S)$ is a penalized form of the EBB scan statistic that is still additive over the data elements s_i. We note that the Δ_i terms are assumed to be a function of only the given data element s_i; they cannot depend on the entire subset S. We acknowledge this as a limitation of the current work, and will investigate extensions to more sophisticated penalties in future work.

B. Prior Log-odds Interpretation

These soft constraints, Δ_i, have a convenient interpretation as the prior log-odds that each data element s_i will be included in the detected subset. Let p_i be the prior probability that data element s_i will be contained in the detected subset. Then Δ_i can be defined as $\log\left(\frac{p_i}{1-p_i}\right)$. The prior log probability of selecting a subset S is then:

$$\begin{aligned}\log \Pr(S) &= \log\Big(\prod_{s_i \in S} p_i \prod_{s_i \notin S} (1-p_i)\Big)\\ &= \sum_{s_i \in S} (\log p_i - \log(1-p_i)) + \sum_{i=1}^{N} \log(1-p_i)\\ &= \sum_{s_i \in S} \Delta_i - \sum_{i=1}^{N} \log(1+\exp(\Delta_i)).\end{aligned}$$

However, we note that the term $\sum_{i=1}^{N} \log(1+\exp(\Delta_i))$ is constant and does not affect the probability of selecting any particular subset. Thus we can ignore this term when optimizing over all subsets of the data, and subtract it once the highest-scoring subset of the data has been identified. When $\Delta_i > 0$, record s_i is more likely to be included in the detected subset, and the opposite is true when $\Delta_i < 0$. When $\Delta_i = 0$ for all i, which is the default setting for spatial-temporal scan statistics, then every subset S is considered equally likely *a priori*.

C. Derivation of Δ_i^t

We now derive the formulas for Δ_i^t that correspond to the following generative model for temporal consistency. Let p_i^t be the prior probability that data element s_i will be contained in the detected subset S^t on time step t. Let x_i^t be 1 if data element s_i is included in S^t, and 0 otherwise. Let n_i^t be the number of neighbors of s_i that are included in S^t, and let k_i be the degree of node s_i. Then our generative model of event propagation, which incorporates temporal consistency constraints, is defined as:

$$\log\left(\frac{p_i^t}{1-p_i^t}\right) = \beta_0 + \beta_1 x_i^{t-1} + \beta_2 \frac{n_i^{t-1}}{k_i}. \tag{1}$$

As a concrete example of the interpretation of this model, assume $\beta_0 = -1.5$, $\beta_1 = 5$, and $\beta_2 = 0$. Then, if a node is included in the previous detected subset, S^{t-1}, it has a 97% prior probability of being included in the current detected subset, S^t. If it was not included in the previous subset, then it only has an 18% probability of being included in the current subset. When $\beta_2 > 0$, the proportion of neighbors included in S^{t-1} will further influence the prior probability of s_i being included in the current subset.

We now compute the total impact Δ_i^t of including x_i^t on the overall penalized log-likelihood ratio score $F(S)$, as compared to the score $F(S \backslash x_i^t)$ when x_i^t is excluded. Given the log-linear model of p_i^t above, we have:

$$\Delta_i^t = \left(\log(p_i^t) - \log(1-p_i^t)\right) + \sum_{j\in S^{t+1}} \left(\log(p_j^{t+1} \mid x_i^t) - \log(p_j^{t+1} \mid \bar{x_i^t})\right) + \sum_{j\notin S^{t+1}} \left(\log(1-p_j^{t+1} \mid x_i^t) - \log(1-p_j^{t+1} \mid \bar{x_i^t})\right). \quad (2)$$

In equation (2), the initial difference results from the prior probability of x_i^t, conditioned on x_i^{t-1} and its number of included neighbors n_i^{t-1} from the *previous* time step. This difference can be calculated directly from the model:

$$\log(p_i^t) - \log(1-p_i^t) = \beta_0 + \beta_1 x_i^{t-1} + \beta_2 \frac{n_i^{t-1}}{k_i}. \quad (3)$$

The two sums in (2) account for the fact that including x_i^t changes the prior probabilities of x_i^{t+1} and its neighbors n_i^{t+1} for the *next* time step. These sums can be rewritten as:

$$\begin{aligned}
&\sum_{j\in S^{t+1}} \left(\log(p_j^{t+1} \mid x_i^t) - \log(p_j^{t+1} \mid \bar{x_i^t})\right) \\
&+ \sum_{j\notin S^{t+1}} \left(\log(1-p_j^{t+1} \mid x_i^t) - \log(1-p_j^{t+1} \mid \bar{x_i^t})\right) \\
&= \sum_{j\in S^{t+1}} (\beta_0 + \beta_1 x_j^t + \beta_2 \frac{n_j^t}{k_j} \mid x_i^t) \\
&- \sum_j f(\beta_0 + \beta_1 x_j^t + \beta_2 \frac{n_j^t}{k_j} \mid x_i^t) \\
&- \sum_{j\in S^{t+1}} (\beta_0 + \beta_1 x_j^t + \beta_2 \frac{n_j^t}{k_j} \mid \bar{x_i^t}) \\
&+ \sum_j f(\beta_0 + \beta_1 x_j^t + \beta_2 \frac{n_j^t}{k_j} \mid \bar{x_i^t}),
\end{aligned} \quad (4)$$

where the function $f(x) = \log(1+\exp(x))$. Next, we note that the contributions to equation (4) are equal to 0 for all nodes j except for node i and its neighbors. For $j = i$, the corresponding terms in (4) simplify to:

$$\beta_1 x_i^{t+1} + f\left(\beta_0 + \beta_2 \frac{n_i^t}{k_i}\right) - f\left(\beta_0 + \beta_1 + \beta_2 \frac{n_i^t}{k_i}\right). \quad (5)$$

For each neighbor j of i, the corresponding terms in (4) simplify to:

$$\beta_2 \left(\frac{x_j^{t+1}}{k_j}\right) + f\left(\beta_0 + \beta_1 x_j^t + \beta_2 \frac{n_j^t}{k_j}\right) - f\left(\beta_0 + \beta_1 x_j^t + \beta_2 \frac{n_j^t + 1}{k_j}\right). \quad (6)$$

Adding the contributions of equations (3), (5), and (6), we obtain:

$$\begin{aligned}
\Delta_i^t &= \beta_0 + \beta_1 \left(x_i^{t-1} + x_i^{t+1}\right) + \beta_2 \left(\frac{n_i^{t-1}}{k_i} + \sum_{j\in S^{t+1}} \frac{1}{k_j}\right) \\
&+ f\left(\beta_0 + \beta_2 \frac{n_i^t}{k_i}\right) - f\left(\beta_0 + \beta_1 + \beta_2 \frac{n_i^t}{k_i}\right) \\
&+ \sum_j f\left(\beta_0 + \beta_1 x_j^t + \beta_2 \frac{n_j^t}{k_j}\right) \\
&- \sum_j f\left(\beta_0 + \beta_1 x_j^t + \beta_2 \frac{n_j^t + 1}{k_j}\right),
\end{aligned} \quad (7)$$

where the sums are taken over all neighbors j of i. In the special case of $\beta_2 = 0$, equation (7) simplifies to:

$$\Delta_i^t = \beta_0 + \beta_1 \left(x_i^{t-1} + x_i^{t+1}\right) + f(\beta_0) - f(\beta_0 + \beta_1). \quad (8)$$

Note that, when $\beta_2 = 0$, we can compute Δ_i^t exactly. When $\beta_2 \neq 0$, we approximate Δ_i^t assuming that $\beta_0 + \beta_2 \ll 0$ and $\beta_0 + \beta_1 \gg 0$. Noting that $f(x) \approx 0$ when $x \ll 0$, and $f(x) \approx x$ when $x \gg 0$, we obtain:

$$\begin{aligned}
\Delta_i^t &\approx \beta_0 + \beta_1 \left(x_i^{t-1} + x_i^{t+1}\right) + \beta_2 \left(\frac{n_i^{t-1}}{k_i} + \sum_{j\in S^{t+1}} \frac{1}{k_j}\right) \\
&- \left(\beta_0 + \beta_1 + \beta_2 \frac{n_i^t}{k_i}\right) + \sum_{j\in S^t} \left(\beta_0 + \beta_1 + \beta_2 \frac{n_j^t}{k_j}\right) \\
&- \sum_{j\in S^t} \left(\beta_0 + \beta_1 + \beta_2 \frac{n_j^t + 1}{k_j}\right) \\
&= \beta_1 \left(x_i^{t-1} + x_i^{t+1} - 1\right) \\
&+ \beta_2 \left(\frac{n_i^{t-1}}{k_i} + \sum_{j\in S^{t+1}} \frac{1}{k_j} - \sum_{j\in S^t} \left(\frac{1}{k_i} + \frac{1}{k_j}\right)\right).
\end{aligned} \quad (9)$$

However, equation (9) assumes knowledge of which other elements are contained in S^t, and this information would not be known in advance. Thus we approximate the final sum over $j \in S^t$ with half the corresponding sum over all neighbors j of i:

$$\begin{aligned}
\Delta_i^t &\approx \beta_1 \left(x_i^{t-1} + x_i^{t+1} - 1\right) \\
&+ \beta_2 \left(\frac{n_i^{t-1}}{k_i} + \sum_{j\in S^{t+1}} \frac{1}{k_j} - \frac{1}{2} \sum_j \left(\frac{1}{k_i} + \frac{1}{k_j}\right)\right).
\end{aligned} \quad (10)$$

The intuitive role of Δ_i^t is that it must *simultaneously* make S^t appear likely to have been generated from S^{t-1} and able to generate S^{t+1}, thus conveying temporal consistency information both forwards and backwards in time.

D. Iterative Convergence

We now have clear definitions and interpretations for $F_{pen}(S) = \sum_{s_i^t \in S}(\lambda_i^t + \Delta_i^t)$ for the EBB scan statistic with temporal consistency constraints. However, recall that the values of Δ_i^t for a given time step t depend on the detected subsets at $t-1$ and $t+1$, which creates a computational paradox. To solve this, our Dynamic Subset Scan uses an iterative method that converges to a (local) optimum. To better approach the global optimum, we can use multiple restarts as well as simulated annealing (which gradually increases the strength of the Δ_i^t from 0 to their full values), wrapped around steps (3)-(13) in the algorithm below.

Algorithm 1 Iterative convergence to local optimum for Dynamic Subset Scan (without multiple restarts or simulated annealing)

```
1:  for window duration $w$ from 1 to *max window* $W$ do
2:    Initialize each of the $w$ spatial subsets independently
      (i.e., separately compute the highest scoring subsets
      $S^t$ for each time step $t$, assuming $\Delta_i^t = 0$ for all $s_i$).
3:    repeat
4:      Randomly select a time step $t$ that is not flagged
        as "Checked". Copy current spatial subset $S^t$.
5:      Compute $\Delta_i^t$ for each node $s_i$ given subsets $S^{t-1}$
        and $S^{t+1}$, using equation (8) or (10).
6:      Compute new optimal subset $S'^t$ for time step $t$
        using $\Delta_i^t$. Without connectivity constraints, simply
        include all positive contributions $\lambda_i^t + \Delta_i^t$; with
        connectivity constraints, call Additive GraphScan.
7:      if new subset $S'^t$ *does not* improve penalized log-
        likelihood ratio of spatial-temporal subset $S$ then
8:        Revert to $S^t$ and mark time step $t$ as "Checked".
9:      end if
10:     if new subset $S'^t$ *does* improve penalized log-
        likelihood ratio of spatial-temporal subset $S$ then
11:       Replace $S^t$ with $S'^t$ and remove "Checked" flags
          from time steps $t-1, t+1$, and $t$.
12:     end if
13:   until no further changes improve penalized log-
      likelihood ratio of spatial-temporal subset $S$, i.e., all
      time steps have been flagged as "Checked".
14: end for
15: Return the highest scoring spatial-temporal subset $S_w^*$.
```

III. Additive GraphScan

The previous section outlined how the expectation-based binomial (EBB) scoring function may be penalized with temporal consistency constraints $F_{pen}(S) = \sum_{s_i \in S}(\lambda_i + \Delta_i)$ while remaining an additive set function over the data elements $s_i \in S$. Optimizing additive functions *without* connectivity constraints is very straightforward and consists of including all records with positive contributions $(\lambda_i + \Delta_i > 0)$ and excluding the rest. Enforcing hard connectivity constraints on additive functions (i.e. determining the "most positive" connected subset) is an interesting and difficult problem. For example, not all nodes making positive contributions will be included in a high-scoring connected subset because they are likely disconnected in the underlying graph structure. Also, a high-scoring connected subset may include a node with a negative contribution in order to connect two positive nodes.

We note that GraphScan [13] exactly identifies the highest scoring connected subset for any scoring function that satisfies the Linear-Time Subset Scanning (LTSS) property [11]. It is trivially shown that additive functions satisfy LTSS, and therefore GraphScan could be used to determine the highest scoring ("most positive", in the case of an additive scoring function) connected subset. However, GraphScan is designed to optimize over more complex scoring functions; most importantly, its computation time is exponential in the graph size and therefore it does not scale well in this setting. Therefore, we propose *Additive GraphScan* as an efficient heuristic alternative to GraphScan which can be used to identify high-scoring (most positive) connected subsets in a given graph structure with real-valued weights at each node.

A. Additive GraphScan Algorithm

Additive GraphScan makes use of the following notation. $w(n)$ is the real-valued weight of node n. A path p is any connected subgraph of nodes. $w(p)$ is the sum of weights for every node in the path. $g(p)$ is the *gain* that would result from merging path p into a single node. It is the difference between the weight of the resulting merged node and the highest weighted node in the path. Identifying and merging paths with positive gains is an integral part of Additive GraphScan. $g(n, p^*)$ is the gain that would result from merging two paths together. The first path, p^*, is a previously identified path of interest with positive gain. The second path is the shortest path between node n and any point along path p^*. $g(n, p^*)$ is the difference between the weight of the resulting merged paths and $\max(w(n), w(p^*))$.

$pw(n)$ is the *pathweight* of a node used when calculating single source, shortest paths traversing through the node. Note the difference between the weight of a node $w(n)$ (which may be positive or negative and is used in the *gain* calculations above) and the pathweight of a node $pw(n)$ (which is non-negative and used in shortest path calculations). Pathweights of positive nodes are set to 0, reflecting no penalty (or reward) for traversing positive nodes while identifying shortest paths. Pathweights for negative nodes with no positive neighbors are $-w(n)$. Pathweights for negative nodes with positive neighbors have $pw(n) = -\min\left(0, w(n) + \sum_{\text{pos neighbors}, n_i} \frac{w(n_i)}{\text{degree}(n_i)}\right)$. We may think of the positive weights uniformly "diffusing" over their negative neighbors and then using this altered weight as the pathweight for negative nodes with positive neighbors. In the case where a large positive node overwhelms its negative neighbor, the negative neighbor's pathweight is set to 0.

Finally, $s(n_a, n_b, n_c)$ determines a fourth node, n_s in the graph as a *Steiner* point for n_a, n_b, and n_c. A Steiner point in this setting is a node that forms the shortest interconnect between the three provided nodes using the pathweights of the graph. $s(n_a, n_b, n_c)$ returns the shortest interconnecting path formed between the three nodes going through n_s.

Some basic pre-processing may be applied to the graph before running Additive GraphScan. For example, any positive node with a positive neighbor may be merged together into a larger, single positive node (adding their weights) and repeated until no further merges exist. Also, any negative nodes with degree of 1 or less may be recursively removed because these are guaranteed to not be included in a high scoring connected subset. Lastly, any negative node with at least two positive neighbors may be merged into a single node if the resulting merged node has a higher weight than any individual positive neighbor. Additive GraphScan can then be applied to the pre-processed graph. While detailed discussion of computational complexity has been omitted due to space limitations, we note that the algorithm scales as $O(kN^2) = O(N^{2.5})$, dominated by steps (3) and (5).

Algorithm 2 Additive GraphScan

```
1: while positive gain path merges exist do
2:     Identify top-$k$ positive nodes where $k = \sqrt{N}$.
3:     Compute path weights $pw(n)$ for all nodes and create
       single-source shortest paths from each top-$k$ node.
4:     Compute $g(p)$ for each shortest path $p$ between top-
       $k$ pairs. Determine highest gain path $p^*$ and record
       endpoints as $n_a$ and $n_b$.
5:     Compute $g(n_i, p^*)$ for each remaining top-$k$ node, $n_i$.
       Determine highest gain node for $p^*$ and record as $n_c$.
       If no positive gain exists between $p^*$ and any $n_i$, then
       merge $p^*$ and restart.
6:     Form new path $p^{**}$ as the union of $p^*$ and the path
       connecting $p^*$ to $n_c$.
7:     Compute $s(n_a, n_b, n_c)$. Compare $w(s(n_a, n_b, n_c))$
       and $w(p^{**})$. Merge the one with higher weight.
8: end while
9: The highest weight merged node is returned as the most
   positive connected subset found by Additive GraphScan.
   Note that this node may need to be "unpacked" to
   determine the contents in the original graph form.
```

B. Additive GraphScan Example

We conclude this section by applying Additive GraphScan to a sample pre-processed graph found in Figure 1. The most positive connected subgraph consists of nodes $\{0, 1, 6, 3, 4, 8, 9\}$ where node 6 is the Steiner point used

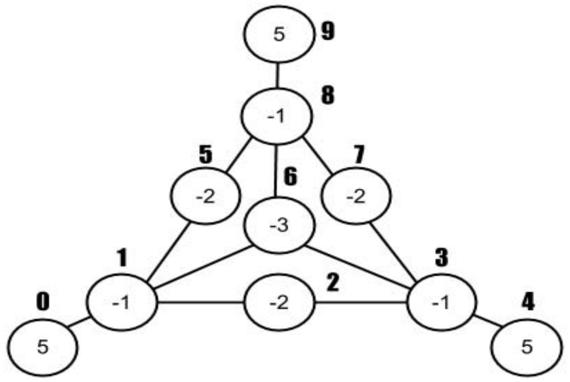

Figure 1. An example graph to demonstrate the Additive GraphScan algorithm. The large bolded numbers are node identifiers and the small numbers within each node are the nodes' corresponding weights. The most positive subgraph consists of nodes $\{0, 1, 6, 3, 4, 8, 9\}$ and is correctly identified by Additive GraphScan.

to connect nodes 0, 4, and 9. Additive GraphScan correctly identifies this subgraph even though node 6 is not on the shortest paths connecting nodes 0 and 4 or nodes 4 and 9. A key insight into the strong performance of Additive GraphScan is delaying path merges while searching for a potential Steiner point. We begin at step (2:) Nodes 0, 4, and 9 are identified as the top-k nodes. (3:) Dijkstra's algorithm is called on nodes 0, 4, and 9 providing single-source shortest path information from each of them. (4:) The shortest path from node 0 to node 4, $p^* = \{0, 1, 2, 3, 4\}$, has highest gain of $g(p^*) = (5-1-2-1+5)-5 = +1$. Because we found a positive gain path between nodes $n_a = 0$ and $n_b = 4$, we continue searching for a third node, n_c. (5,6:) We find $n_c = 9$ with $p^{**} = \{0, 1, 2, 3, 4, 7, 8, 9\}$ and $w(p^{**}) = 8$. (7:) Calculate a Steiner point for nodes 0, 4, and 9 and note that node 6 forms the shortest interconnect between these three points. This interconnect is formed by the nodes $\{0, 1, 6, 3, 4, 8, 9\}$ and $w(s(0, 4, 9)) = 5 - 1 - 3 - 1 + 5 - 1 + 5 = 9$. Because $w(s(0, 4, 9)) > w(p^{**})$ we condense $s(0, 4, 9)$ into a single node with weight 9. After this merge, no more positive gain path merges exist and the loop exits. (9:) The highest scoring connected subset is then $\{0, 1, 6, 3, 4, 8, 9\}$. Notice that greedily merging either p^* or p^{**} would have resulted in a sub-optimal merge.

IV. Results

A. Comparison of Additive GraphScan vs. GraphScan

In this section, we compare our fast heuristic, Additive GraphScan, to the slower, but exact, GraphScan algorithm. First, we present a runtime analysis comparing the two optimization algorithms. We used the much larger "network 2" provided in the Battle of the Water Sensor Networks [1] and created connected subgraphs of various sizes from 50 to 500 nodes from the network. We then processed the graphs with three different scans: Dynamic Subset Scan with

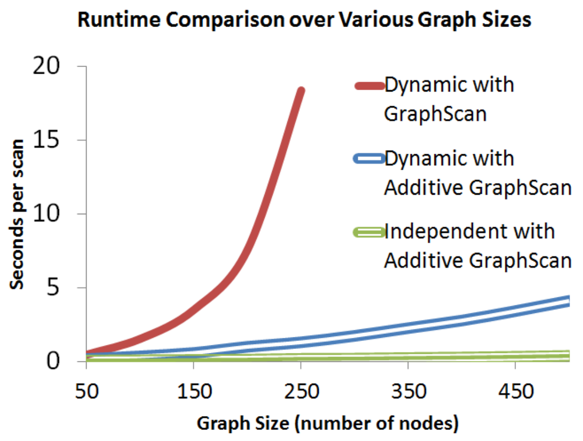

Figure 2. Runtime comparisons for the Dynamic Subset Scan with GraphScan and Additive GraphScan as the optimization algorithm. Independent Scan with Additive GraphScan is also shown.

GraphScan, Dynamic Subset Scan with Additive GraphScan, and Independent with Additive GraphScan, and reported the average runtime for each method. These results are shown in Figure 2.

GraphScan begins to struggle with graph sizes of 250 nodes while Additive GraphScan quickly scans graphs of 500 nodes in approximately 4.1 seconds. Independent with Additive GraphScan processed the entire 12,000+ node "Network 2" in 221 seconds while Dynamic with Additive GraphScan required 1830 seconds (approximately a half hour). This difference represents the additional calls to Additive GraphScan required by Dynamic Subset Scan to "align" the individual spatial subsets according to the temporal consistency constraints.

We conclude our comparison of Additive GraphScan and GraphScan by analyzing the scores of the spatial-temporal subsets identified by the scanning methods using both Additive GraphScan and GraphScan. Our approximation ratio results compare the highest-scoring subsets found by Additive GraphScan and GraphScan as a percentage averaged over 2000 simulations. The ratio of Additive GraphScan's to GraphScan's score did not fall below 99.1%, indicating that Additive GraphScan is providing a huge speed increase with minimal loss of accuracy compared to scan statistics using GraphScan.

B. *Detecting, Tracking, and Source-Tracing Plumes*

In this section, we evaluate the detection, tracking, and source-tracing abilities of the Dynamic Subset Scan. The 129-node "Network 1" from the Battle of the Water Sensor Networks [1] served as the test bed for these evaluations. We ran two simulations, one with sensors at FPR = 0.1 and TPR = 0.9 and a second with weaker sensors at FPR = 0.2 and TPR = 0.8. All results below are averaged over 200 contaminant plumes simulated for 12 hours each. A separate

Table I
SUMMARY OF LEARNED PARAMETER VALUES

method	FPR	TPR	β_0	β_1	β_2
Dynamic	0.1	0.9	-1.3	6.6	1.8
Dynamic	0.2	0.8	-1.3	3.6	2.0
Dynamic Alt.	0.1	0.9	-1.1	8.2	0
Dynamic Alt.	0.2	0.8	-1.2	2.4	0

100-plume training set was used for cross-validation for the scan statistics that required learning parameters.

We compare 4 different spatial-temporal scan statistics:

- *Static* scan does not allow the detected spatial region to change over time.
- *Independent* scan allows the detected spatial region to change over time but does not share temporal information between time steps.
- *Dynamic* scan allows the detected spatial region to change over time and uses temporal consistency constraints to "align" the individual time steps.
- *Dynamic Alt.* scan is similar to Dynamic scan but does not use any information from neighbors when enforcing temporal consistency constraints, i.e., we force $\beta_2 = 0$ when learning the model parameters.

For the temporal component of the scans, we set *max window size W* = 12. This allows Static, Dynamic, and Dynamic Alt. to detect a spatial-temporal subset between 1 and 12 hours in duration. However, for the Independent scan, the highest scoring spatial-temporal region will always be maximum duration. This consequence of the Independent scan is discussed further below.

Each of these methods have graph connectivity constraints enforced through Additive GraphScan. Simpler versions of the scans without connectivity constraints were evaluated and had much lower performance for detection, tracking, and source-tracing. (Detailed results are omitted due to space limitations.)

We set the $\beta_0 \ldots \beta_2$ parameters for the Dynamic scan, and the β_0 and β_1 parameters for Dynamic Alt., using a grid search on the separate 100-plume training set. The parameter values that maximized spatial-temporal overlap in the training data are shown in Table I. It is interesting to note the large changes in β_1 when moving from the easier to harder scenarios, while β_0 and β_2 remain relatively constant.

Figure 3 reports the average time required by each method to detect a contaminant plume for various false positive rates. These results were calculated by processing 2160 "background" hours (approximately 3 months of data) with no contaminants. These were compared with scores produced by the scan statistics during the 200 simulated plumes. The 0 false positive alarms interpretation is that it took 4.3 hours

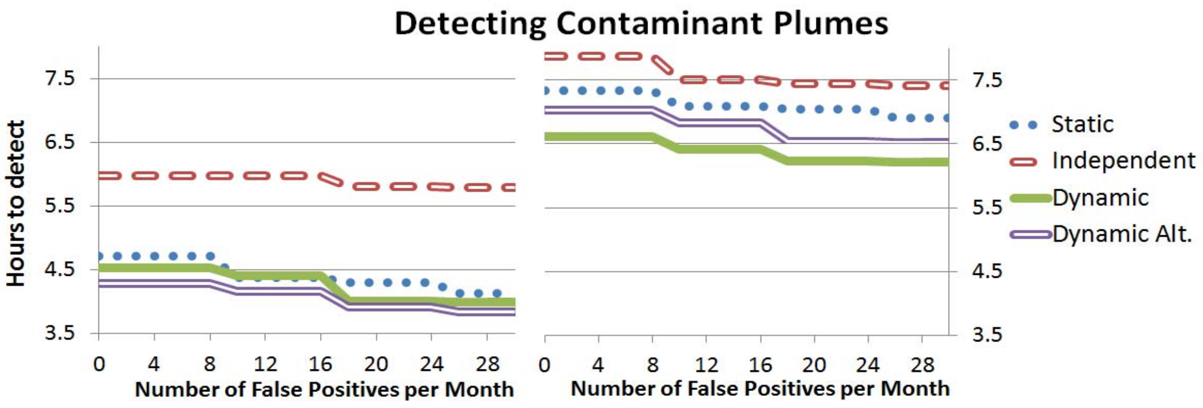

Figure 3. Detection results for FPR = 0.1 and TPR = 0.9 sensors on the left and FPR = 0.2 and TPR = 0.8 sensors on the right. We present our detection results through Activity Monitoring Operating Characteristic (AMOC) curves. These show the average time required for each method to detect a contaminant in the system, assuming a fixed number of allowable false positives per month.

(on average) for the scores produced by Dynamic Scan to exceed the *largest* score found by Dynamic Scan in the 2160 "background" hours. As the threshold for detection is lowered, the number of false positive alarms increases but the time to detect decreases, as shown by the Activity Monitoring Operating Characteristic (AMOC) curves [4] in Figure 3.

The Static, Dynamic, and Dynamic Alt. methods achieve similar power for event detection in the easier scenario (FPR = 0.1, TPR = 0.9). However, Dynamic achieves the overall best performance (6.62 hours to detect at 0 false positives) when detecting a weaker signal (FPR = 0.2, TPR = 0.8). Note the influence that a node's neighbors have on distinguishing performance between Dynamic and Dynamic Alt. in the 0.2/0.8 scenario. The easier scenario did not require additional information from neighbors in order to obtain similar performance, but this information is important when working with weaker sensors. The Independent method's poor performance is due to the relatively high subset scores found by Independent when no contaminant is present. Its *unconstrained* flexibility allows it to overfit to noise in the background, making detection of a true contamination event more difficult.

Figure 5 reports the methods' tracking ability over the duration of a spreading contaminant plume (12 hours). We measure a scan statistic's tracking ability through *spatial-temporal overlap*. Spatial-temporal overlap is a combination of precision and recall applied to spatial-temporal subsets. A measure of 1.0 corresponds to perfect agreement between the affected and detected spatial-temporal regions, while 0.0 means the affected and detected regions are disjoint. For two spatial-temporal subsets, *Affected* and *Detected*, the overlap is defined as: $\frac{|\text{Affected} \bigcap \text{Detected}|}{|\text{Affected} \bigcup \text{Detected}|}$. See Figure 4 for details.

The relative performance of the Static and Dynamic methods in the easier scenario demonstrates Static's lack

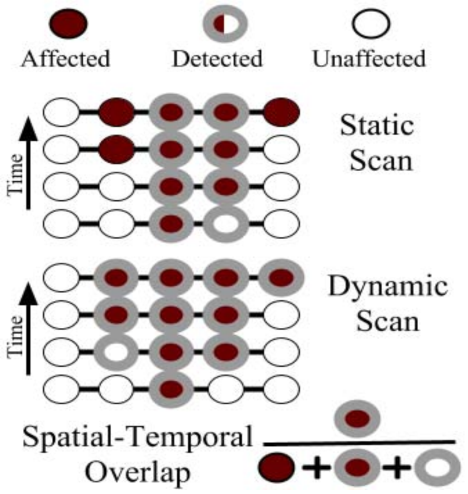

Figure 4. This figure demonstrates the calculation of spatial-temporal overlap for plume tracking. A plume spreads through a simple 5-node line graph over the course of four time steps. Affected nodes turn from white to red as the contaminant spreads. The Static scan method is constrained to keep the exact same detected spatial region throughout the event duration. Hence, it may fail to capture the plume at later time steps. Dynamic Scan allows the detected spatial region to change at each time step, tracking the plume as it spreads. Due to connectivity constraints, both methods must return a connected subgraph as the detected spatial region at each time step. Spatial-temporal overlap is penalized for both false positives and false negatives. A measure of 1.0 corresponds to perfect agreement between the affected and detected spatial-temporal regions, while 0.0 means that the affected and detected regions are disjoint.

of tracking ability as the plume grows over time. Static's tracking performance quickly levels off while Dynamic continues to achieve higher spatial-temporal overlap over the course of the contamination event. This increase in performance is due to the (constrained) flexibility allowed to the Dynamic Subset Scan. The difference in tracking performance between Static and Dynamic methods is not as large in the harder scenario, but we again observe

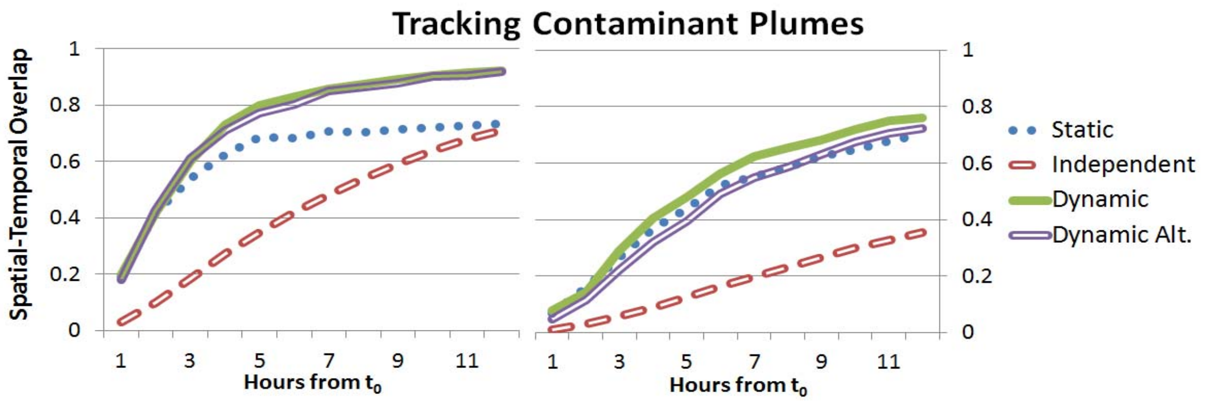

Figure 5. Tracking results for FPR = 0.1 and TPR = 0.9 sensors on the left and FPR = 0.2 and TPR = 0.8 sensors on the right. We measure tracking ability by reporting the spatial-temporal overlap of the detected and affected subsets over the course of 12 hours.

the importance of incorporating information from a node's neighbors. The poor performance of the Independent scan, particularly in the early stages of the contamination event, is due to its tendency to report spatial-temporal regions of maximum duration which are not a good match to the true affected region.

Figure 6 reports the methods' ability to identify *where* the contaminant originated over the duration of the plume (12 hours). This is measured through purely spatial overlap between the *earliest* time step in the detected region and the source node(s) of the plume. We note that it is possible for a quickly spreading plume to affect multiple nodes within the first hour. In such cases, we treat all of these nodes as source nodes.

The source-tracing results clearly demonstrate the advantage of sharing information between time steps during the optimization process. Static's ability to identify the source nodes actually *decreases* over the course of the contamination event as more information is gathered. Exploring this result further, we observe that Static has very high spatial recall (0.995) but very low precision (0.144) for identifying the source nodes on hour 12. This suggests that Static tends to return very large subsets at the later stages of the plume. These regions may accurately reflect the affected subset of nodes on the current time step: a follow-up investigation, with detailed results omitted due to space limitations, suggests that the relative performance of Static and Dynamic as measured by spatial overlap coefficient on the *current* time step is very similar. However, the large regions returned by the Static method harm its ability to accurately identify the source of the contaminant.

The key to the Dynamic Subset Scan's success for source-tracing is the *backwards* flow of temporal consistency information allowed in our model. Dynamic is able to change the detected subset for previous time steps based on new, more current data. This gives it superior source-tracing abilities in both the easier (0.1/0.9) and harder (0.2/0.8) scenarios. For the harder scenario, we again observe the importance of neighbor information: while the Dynamic method achieves similar performance to Static during the early stages of the contamination event and much better performance in the later stages, the performance of Dynamic Alt. (which does not use neighbor information) does not surpass Static until the ninth hour. Finally, we note the substantial increase in performance of the Independent method at hour 12, though its overall performance is still low. This is an artifact of Independent preferring to return 12-hour regions.

V. Conclusions

This work introduced the Dynamic Subset Scan for detecting, tracking, and source-tracing *dynamic* patterns that change the affected subset over time. This novel extension of the well-known spatial and subset scan statistics is composed of two main contributions. First is the incorporation of *temporal consistency constraints* that may be enforced on temporally adjacent, spatial subsets. These constraints are a fruitful compromise between traditional spatial-temporal scan statistics that do not allow the detected region to change over time (Static) and the other extreme where temporal information is ignored (Independent). The key insight to enforcing temporal consistency constraints is recognizing that the expectation-based binomial scoring function may be written as an additive function over the data records. This allows for additional terms (constraints) to be included in the penalized log likelihood ratio while remaining efficient to optimize. Critically, these temporal consistency constraints were derived to allow temporal information to be shared both forward and backward in time.

Our second novel contribution is the Additive GraphScan algorithm, which allows the Dynamic Subset Scan to enforce both soft temporal consistency constraints *and* hard connectivity constraints while scaling to large, real world networks. Additive GraphScan is a fast, heuristic alternative to GraphScan. However, our results demonstrate

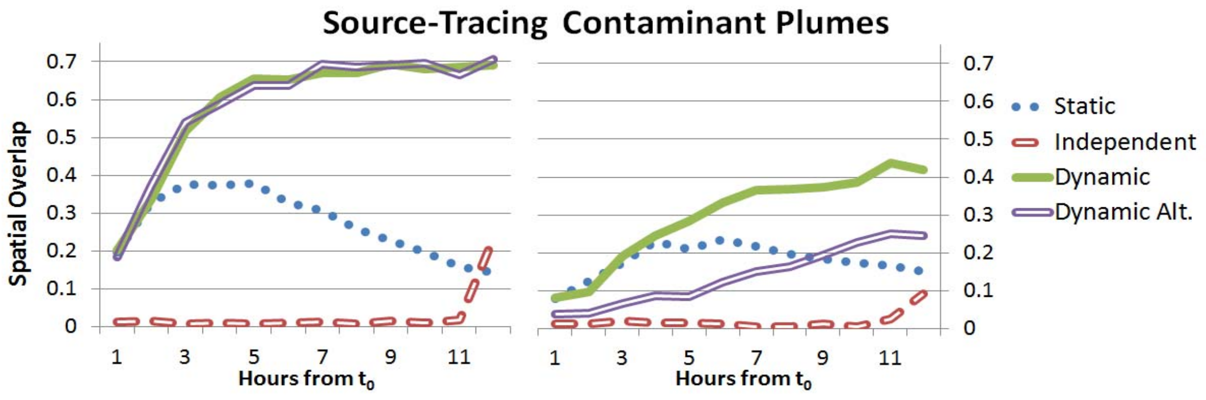

Figure 6. Source-tracing results for FPR = 0.1 and TPR = 0.9 sensors on the left and FPR = 0.2 and TPR = 0.8 sensors on the right. We measure source-tracing ability by reporting the spatial overlap between the *earliest detected* spatial region and the original affected node(s).

an approximation ratio of over 99%, suggesting a very small sacrifice for dramatic gains in speed and scalability.

The Dynamic Subset Scan was evaluated on data provided through the "Battle of the Water Sensor Networks" [1]. Dynamic scan succeeded in detecting contamination events sooner and tracking these events more accurately compared to other competing methods. The gains were due to Dynamic Scan's constrained flexibility: competing methods either failed to capture the dynamics of the spreading plume (Static) or were susceptible to over-fitting from lack of constraints (Independent). In scenarios with a weaker signal to be detected, incorporating information from a node's neighbors in the Dynamic Scan proved worthwhile, leading to substantial gains in performance on the detection, tracking, and source-tracing tasks.

In summary, relaxing constraints on spatial-temporal region shape must be done carefully. Strict temporal constraints work well when the affected subset of the data does not change over time. However, removing them completely in order to track dynamic patterns performs poorly as shown by the Independent method results. Dynamic Subset Scan with temporal consistency and connectivity constraints provides a scalable solution for future work in dynamic pattern detection in graph-based or sensor network data.

Acknowledgment

A special thanks to R. Ravi for providing insights on the use of Steiner points. This work was partially supported by NSF grants IIS-0916345, IIS-0911032, and IIS-0953330.

References

[1] Avi Ostfeld et al. The battle of water sensor networks: A design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management*, 134(6):556–568, 2008.

[2] J. Berry, R. D. Carr, W. Hart, V. J. Leung, C. A. Phillips, and J. P. Watson. Designing contamination warning systems for municipal water networks using imperfect sensors. *Journal of Water Resources Planning and Management*, 135(4):253–263, 2009.

[3] J. Berry, L. Fleischer, W. Hart, and C. Phillips. Sensor placement in municipal water networks. *J. Water*, 131:237–243, 2003.

[4] T. Fawcett and F. Provost. Activity monitoring: noticing interesting changes in behavior. In *Proc. 5th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 53–62, 1999.

[5] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In *Proc. 16th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, pages 1019–1028, 2010.

[6] M. W. Koch and S. A. Mckenna. Distributed sensor fusion in water quality event detection. *Journal of Water Resources Planning and Management*, 137:10–19, 2011.

[7] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526, November 2008.

[8] M. Kulldorff. A spatial scan statistic. *Communications in Statistics: Theory and Methods*, 26(6):1481–1496, 1997.

[9] S. Myers and J. Leskovec. On the convexity of latent social network inference. In *Advances in Neural Information Processing Systems 23*, pages 1741–1749. 2010.

[10] D. B. Neill. Expectation-based scan statistics for monitoring spatial time series data. *International Journal of Forecasting*, 25:498–517, 2009.

[11] D. B. Neill. Fast subset scan for spatial pattern detection. *Journal of the Royal Statistical Society (Series B: Statistical Methodology)*, 74(2):337–360, 2012.

[12] D. B. Neill, A. W. Moore, M. R. Sabhnani, and K. Daniel. Detection of emerging space-time clusters. In *Proc. 11th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*, 2005.

[13] S. Speakman and D. B. Neill. Fast graph scan for scalable detection of arbitrary connected clusters. In *Proc. International Society for Disease Surveillance Annual Conf*, 2010.

[14] R. Viswanathan and P. K. Varshney. Distributed detection with multiple sensors: Part I Fundamentals. *Proceedings of the IEEE*, pages 54–63, 1997.