Abstract

Scan statistics have become a popular approach used for detecting “hotspots” and “anomalies” in spatio-temporal and network data. This methodology involves maximizing a score function over all connected subgraphs and are NP-hard in general. A number of heuristics have been proposed for these problems, but they do not provide any quality guarantees. In this paper, we develop a unified framework for designing algorithms for optimizing a large class of parametric and non-parametric scan statistics for networks, subject to connectivity constraints. Our algorithms run in time that scales linearly on the size of the graph and depends on a parameter we call the “effective solution size”, while providing rigorous approximation guarantees. In contrast, most prior methods have super-linear running times in terms of graph size. Extensive empirical evidence demonstrates the effectiveness and efficiency of our proposed algorithms in comparison with state-of-the-art methods. Our contributions are:

1. **Rigorous algorithms**: We develop a unified framework for optimizing a large class of parametric and non-parametric scan statistics for networks with connectivity constraints, which scales linearly in the network size and is a function of a parameter defined as the “effective solution size”. We also give rigorous bounds on the solution quality (summarized in Theorem 3.1). In other words, our framework encompasses many different network scan statistics—this contrasts with all prior methods, which are developed for specific statistics; further, our approach also holds for the extensions of these functions with both node and edge weights, which generalize Steiner connectivity problems. In practice, the effective solution size parameter is very small (see Section 5.6), making the time complexity of our algorithms better than prior methods, which are super-linear in the network size.

2. **Scaling**: We develop a preprocessing and refinement technique that reduces the solution size without degrading the quality score beyond a provable constant factor (Section 3.4). The resulting algorithms are able to scale to graphs with over a million nodes in minutes and are significantly faster than state-of-the-art methods (Figure 1), which have only been run on graphs of up to $10^4$ nodes.

3. **Experimental results**: We show that our algorithms give significant improvement over the scores computed by different baselines, with over 25% improvement in some instances, compared to the best baseline method (Section 5.3 and Table 1 below). Better objective scores also translate to higher anomaly detection power with 3% improvement on accuracy and F1 score over state-of-the-art methods. Our algorithmic framework has the added advantage that different score func-
tions can be optimized by just modifying the specific objective function within the same implementation.

Table 1: Non-parametric scan statistics optimization. Our algorithm FastColCodeNP obtains higher scores than previous methods in real-world datasets. (Section 5.3)

<table>
<thead>
<tr>
<th>Graph</th>
<th>FastColCodeNP</th>
<th>GS</th>
<th>EL</th>
<th>GL</th>
<th>DFS</th>
<th>NPHGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CitHepPh</td>
<td>116.732</td>
<td>57.214</td>
<td>6.541</td>
<td>43.364</td>
<td>116.732</td>
<td></td>
</tr>
<tr>
<td>NEast</td>
<td>68.525</td>
<td>64.214</td>
<td>23.366</td>
<td>23.211</td>
<td>55.541</td>
<td>17.696</td>
</tr>
<tr>
<td>Twitter</td>
<td>128.740</td>
<td>128.722</td>
<td>116.732</td>
<td>125.824</td>
<td>14.412</td>
<td>121.632</td>
</tr>
<tr>
<td>BWSN</td>
<td>1722.790</td>
<td>1722.388</td>
<td>1722.388</td>
<td>1720.243</td>
<td>1457.410</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Scalability. Running time as a function of number of nodes. (Section 5.5)

For brevity, some of the results, including hardness of the problems, proof details, and pseudocode, are deferred to the supplementary material.

2 Preliminaries

We are given a graph \( G = (V, E) \), where \( V \) is a set of \( n \) vertices or nodes, and \( E \subseteq V \times V \) is a set of \( m \) edges. Each vertex \( v \in V \) has two values associated with it: (1) a population count, \( b(v) \), which indicates the count that we expect to see at the node \( v \), e.g., the number of people in a county, corresponding to node \( v \), and (2) an event count, \( c(v) \), which indicates how many occurrences of an event of interest are seen at the node, e.g., the number of cases of a disease in a county. These values vary over time, but we will not indicate the time in the notation in order to keep it simple. The notation used in this paper is summarized in Table 3.

2.1 Non-Parametric Scan Statistics. Non-parametric scan statistics do not assume an underlying distribution or process on the graph. Instead, they first estimate a \( p \)-value for each vertex based on empirical calibration by comparing the current feature \( (c(v) \text{ and } b(v)) \) of this vertex with its features in the historical data. The problem of anomaly detection has been formalized as a hypothesis testing problem for testing whether the empirical \( p \)-values are uniformly distributed on \([0, 1]\) [17, 19, 16]. Let \( \alpha \in [0, 1] \) be significance level and let \( w(v, \alpha) \) denote the weight of a node \( v \) as a function of \( \alpha \). For a set of nodes \( S \), let \( W(S, \alpha) = \sum_{v \in S} w(v, \alpha) \) and \( N(S) \) be a function of the cardinality of the set. Then, the score functions can be expressed in the following general form:

\[
(2.1) \quad F(S) = \max_{\alpha \leq \alpha_{max}} \phi(W(S, \alpha), N(S), \alpha),
\]

The significance level \( \alpha \) can be optimized between 0 and some constant \( \alpha_{max} \). We use \( w(v) \) and \( W(S) \) to denote \( w(v, \alpha) \) and \( W(S, \alpha) \), respectively, whenever \( \alpha \) is clear from the context. For clarity, in the rest of the paper, we consider the specific case when \( N(S) = |S| \) is the cardinality of \( S \) and \( w(v, \alpha) \) is 1 if the \( p \)-value of \( v \) is less than \( \alpha \), 0 otherwise — we say these nodes are significant at level \( \alpha \). Then, \( W(S, \alpha) \) is the number of significant nodes in \( S \). It is easy to generalize our results to other functions \( W(S, \alpha) \) and \( N(S) \). Table 2 shows examples of non-parametric scan statistics.

2.2 Parametric Scan Statistics. Parametric scan statistics assume that counts observed at each node are generated from some parameterized distribution and formalize anomaly detection as a hypothesis testing problem [11, 16]. Common choices are distributions from the exponential family, such as Poisson or Normal. The goal is to maximize an appropriate scan statistic function \( F(S) \), typically a likelihood ratio. These score functions can be expressed as

\[
(2.2) \quad F(S) = g(C(S), B(S)),
\]

where \( C(S) = \sum_{v \in S} c(v) \), \( B(S) = \sum_{v \in S} b(v) \), and the function \( g \) is defined depending on the score function considered. We refer to [11, 16, 17] for discussion on the strengths and limitations of parametric scan statistics.

2.3 Problem Formulation. Given a graph \( G = (V, E) \), a score function \( F(\cdot) \) and the associated counts for the model, the objective is to find a connected subset \( S \subseteq V \) that maximizes \( F(S) \).

Remark. As we show in the supplementary material, the above problem is NP-hard. In Section 3, we develop algorithms to maximize \( F(S) \), restricted to sets with \( |S| \leq k \), where \( k \) is a parameter that represents the solution size. In Section 3.4, we show that we can compress specific subsets of nodes into “supernodes”, using a process we refer to as refinement. The size of a set \( S \) computed in terms of these supernodes will be referred to as the effective solution size, and it becomes significantly smaller than the original size of \( S \). Our final algorithms will find solutions with effective solution size at most \( k \).

3 Algorithms for Non-Parametric Scan Statistics

In this section, we present an algorithm for non-parametric functions that are characterized by equation
Table 2: A common formulation for parametric and non-parametric scan statistics that can be optimized using our methods, along with their applications. Many other functions are described in the supplementary material.

<table>
<thead>
<tr>
<th>Non-Parametric Scan Statistics $F(S) = \max_{\alpha \leq \alpha_{\text{max}}} \phi(W(S, \alpha), N(S, \alpha))$</th>
<th>Original Form</th>
<th>General Form</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berk-Jones [5]</td>
<td>$F(S) = \max_{\alpha \leq \alpha_{\text{max}}}</td>
<td>S</td>
<td>KL(\frac{W(S, \alpha)}{N(\alpha)}), \alpha)$.</td>
</tr>
<tr>
<td>Higher Criticism [6]</td>
<td>$F(S) = \max_{\alpha \leq \alpha_{\text{max}}} \frac{W(S, \alpha) - E_s}{\sqrt{S N(\alpha(1-\alpha))}}$.</td>
<td>$\phi(a, b, \alpha) = b \cdot \sqrt{\frac{a \log(a/b)}{\alpha}} + b(1-a)$</td>
<td>In addition to the same applications above, it was also used in detection of rare and week effects in Genomics and Genetics [9] ($</td>
</tr>
<tr>
<td>Parametric Scan Statistics $F(S) = g(C(S), B(S))$, $C(S) = \sum_{v \in S} c(v)$, $B(S) = \sum_{v \in S} b(v)$</td>
<td>$F(S) = \sum_{i \in S} s_i/\sqrt{</td>
<td>S</td>
<td>}$</td>
</tr>
</tbody>
</table>

Table 3: Definitions and notation used in the paper

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b(v), c(v)$</td>
<td>population and event counts of node $v$</td>
</tr>
<tr>
<td>$\alpha, \alpha_{\text{max}}$</td>
<td>significance level, maximum significance level</td>
</tr>
<tr>
<td>Significant node (at level $\alpha$)</td>
<td>a node with $p$ value below $\alpha$</td>
</tr>
<tr>
<td>$\text{Nbr}(v)$</td>
<td>set of neighbors of $v$</td>
</tr>
<tr>
<td>$w(v), w(v, \alpha)$</td>
<td>weight of node $v$, based on its $p$-value and the significance level $\alpha$</td>
</tr>
<tr>
<td>$W(S), W(S, \alpha)$</td>
<td>denotes $\sum_{v \in S} w(v, \alpha)$</td>
</tr>
<tr>
<td>$F(S)$</td>
<td>any of the functions in Table 2</td>
</tr>
<tr>
<td>$K$</td>
<td>The set ${1, \ldots, k}$</td>
</tr>
<tr>
<td>$\text{col}(u)$</td>
<td>color of node $u$ from set $K$</td>
</tr>
<tr>
<td>$T$</td>
<td>Subset of $K$ (denotes colors)</td>
</tr>
<tr>
<td>$M(v, T)$</td>
<td>$\max_{S \subseteq V} W(S)$, where the maximization is over connected colorful sets $S \subseteq V$, such that $u \in S$ and ${\text{col}(u) : u \in S} = T$</td>
</tr>
<tr>
<td>$\psi_1, \psi_2(\alpha)$</td>
<td>$\max_{T:</td>
</tr>
<tr>
<td>$S^<em>_1, S^</em>_2(\alpha)$</td>
<td>Set with weight $\psi_1$ and $\psi_2(\alpha)$</td>
</tr>
<tr>
<td>$\text{OPT}(F, k)$</td>
<td>$\max_{S:</td>
</tr>
</tbody>
</table>

3.1 First idea: Monotonicity. A key observation is that the functions $\phi(W(S, \alpha), N(S, \alpha))$ are monotonically increasing functions of $W(S, \alpha)$ under some conditions, as described below.

Lemma 3.1. The non-parametric scan statistics functions characterized by equation (2.1) are increasing functions of $W(S, \alpha)$ if $\frac{W(S, \alpha)}{N(\alpha)} \geq \alpha$ and $N(S)$ is constant.

For example, in the Berk-Jones (BJ) statistic from Table 2, the function increases with the number of significant nodes—nodes with $p$-value less than $\alpha$. Further, given two node sets of the same size, the set with more significant nodes scores higher according to the BJ statistic. This provides us a way to optimize $F(S)$ by maximizing the function $\phi(\cdot)$ for sets of fixed size, as will be discussed next.

3.2 Second idea: Constraining the solutions. We introduce the idea of a coloring of the nodes and only consider connected subgraphs $S$, in which all nodes have distinct colors. Our approach builds on the color-coding technique of [2], but it involves several new techniques to scale the algorithm up to graphs with millions of nodes.

Let $K = \{1, 2, \ldots, k\}$ be a set of colors—where $k$ is a parameter—and let $\text{col}(v) \in K$ denote the color for node $v$. We say that a subgraph induced by set $S \subseteq V$ is colorful if $\text{col}(u) \neq \text{col}(v)$, for all $u, v \in S$. For a node $v$ and subset of colors $T \subseteq K$, we let $M(v, T) = \max_{S \subseteq V} W(S)$, where the maximization is over all connected and colorful sets $S \subseteq V$, such that $v \in S$, $|S| = |T|$, and $\{\text{col}(u) : u \in S\} = T$. In other words, we only consider a set $S$ if each node in the set has a distinct color from $T$. These definitions are illustrated in Figure 2. $M(v, T)$ can be computed by a dynamic program with a recurrence given in the lemma below.

Lemma 3.2. Let $M(v, T)$ be defined as above. For any node $v$ and color $s$, $M(v, \{s\}) = w(v)$ if $\text{col}(v) = s$, else $M(v, \{s\}) = -\infty$. If $|T| \geq 2$: $M(v, T) = \max_{u \in \text{Nbr}(v)} \{M(v, T_1) + M(u, T_2)\}$, where $T_1, T_2 \subseteq T$, the maximum is over all partitions $T_1 \cup T_2 = T$.

3.3 ColCodeNP In Algorithm 1, we present COLCODENP for optimizing non-parametric statistics. Re-
call the notation in Table 3. Let $F(S)$ denote any of the non-parametric functions in Table 2, and let $\text{OPT}(F, k) = \max_{|S| \leq k} F(S)$, where the maximum is over all connected subsets $S$ of size $\leq k$, for a given $\alpha_{\text{max}}$. Algorithm ColCODENP takes the size bound $k$ as input, and an error parameter $\epsilon$, which indicates the probability of not finding the optimum solution.

**Main steps.** We describe the main steps of ColCODENP connecting with the two ideas from above.

- The set $A$ in line 3 of ColCODENP denotes the set of distinct $p$-values of the nodes less than $\alpha_{\text{max}}$: it suffices to find the maximum of $\phi(W(S, \alpha), N(S), \alpha)$ for $\alpha \in A$. The for loop in lines 4—6 finds the best solution for each $i \in K$ and any given $\alpha$ (by calling MAXWEIGHT in line 6), and the maximum is computed in line 7.

- MAXWEIGHT finds the best solution $S_i^*$ of size $i$, for each $i \in K$ using the idea described in Section 3.2. We show an example in Figure 2.

- Each iteration of the outer for loop in lines 14—20 starts with a random coloring (line 15). This is Step 2 in Figure 2.

- The inner for loop in lines 16—17 computes the base case of the dynamic program from Lemma 3.2; then, we solve the program bottom-up in lines 18—19. These are Steps 3 and 4 in Figure 2.

- $\psi_i$ keeps track of the maximum weight solution restricted to size $i$, and it is updated if $M(v, T)$ denotes a better solution for size $|T|$.

**Theorem 3.1.** For any any non-parametric function $F(\cdot)$ in Table 2, algorithm ColCODENP returns solution $S_i^*$ satisfying $\Pr[\text{OPT}(F, k) \geq i - \epsilon]$, in time $O(2^k e^k |A| m \log (n^2 / \epsilon))$, and using space $O(2^k n)$, where $A$ is the set defined in line 3 of Algorithm 1.

*Proof. (Sketch)* We start with the proof of correctness of our algorithm, which involves three parts. The first observation is that within the outer loop for the procedure MAXWEIGHT, for each random coloring $\text{col}(\cdot)$, $\max_v M(v, \{1, \ldots, k\})$ is correctly computed. This follows because the algorithm is a dynamic program that computes all $M(v, T)$ for $T \subseteq K$ using the recurrence in Lemma 3.2.

Next, we observe that the algorithm correctly finds $\psi_i(\alpha)$ —the maximum weight among sets of size $i$ for a given $\alpha$—for each $i, \alpha$, with probability at least $1 - \epsilon / n^2$. The procedure MAXWEIGHT is called with parameter $\epsilon' = \epsilon / n^2$ and $e^k \log (1 / \epsilon')$ colorings. Let $X_{ij}$ be the maximum weight found over subsets of size $i$ in the $j$th random coloring. In the supplementary material, we show that $\Pr[\max_i X_{ij} \neq \psi_i(\alpha)] \leq \epsilon' = \epsilon / n^2$. The number of possible choices for $\alpha$ is $|A|$, which satisfies $|A| \leq n$. Therefore, by a union bound, it follows that for all $i, \alpha \in A$, we have $\Pr[\max_j X_{ij} \neq \psi_i(\alpha)] \leq n^2 \epsilon^i \leq \epsilon$, and the algorithm correctly computes $\psi_i(\alpha)$ for all $i, \alpha$ with probability $1 - \epsilon$.

Finally, for any fixed $i, \alpha$, by Lemma 3.1, $\phi(W(S), N(S), \alpha)$ is an increasing function of $W(S)$ when $N(S)$ is fixed. This implies that $\max_{S, N(S) = i} \phi(W(S), N(S), \alpha) = \psi_i(\alpha) = F(S_i^*(\alpha))$. Therefore, $\max_{i \in K, \alpha \in A} F(S_i^*(\alpha)) = \text{OPT}(F, k)$, and it follows that Algorithm ColCODENP correctly computes $\text{OPT}(F, k)$ with probability at least $1 - \epsilon$.

Next, we consider the space and time complexity. The algorithm maintains the array $M$, indexed by nodes and all possible color sets, which leads to the space complexity of $O(2^k n)$, since there are at most $2^k - 1$ possible non-empty color sets. The running time is the result of solving the recurrence for each node $v$, and for each color set $T$; this requires examining each possible partition $T_1 \cup T_2 = T$, and each neighbor $u \in \text{Nbr}(v)$, which requires time $O(|\text{Nbr}(v)| 2^{|T|})$. Therefore, the total running time for each coloring is $O(\sum_v |\text{Nbr}(v)| 2^k) = O(\sum_v |\text{Nbr}(v)| 2^k) = O(n^{2k} n)$. The algorithm considers $O(e^k \log (n^2 / \epsilon))$ colorings, so the running time follows.

**3.4 Techniques for Scaling.** ColCODENP has rigorous guarantees on the quality of the solution; however, if applied directly, it would only be feasible to discover small anomalies due to the exponential depen-
For a parameter \( \beta \) large solutions by setting we can usually discover In Section 5.6, we show that this refinement is very
The effective size of the subgraph 5 nodes of weight 1 for 2 nodes of weights 3 and 2.
We observe that neighboring significant nodes can
Graph refinement and effective solution size.
Graph refinement involves compressing subsets of nodes into “supernodes”. The size of a set \( S \) after refinement is determined in terms of the nodes and supernodes in it. This new size is called effective size, and, in practice, it is significantly smaller than the original size of \( S \).
We observe that neighboring significant nodes can be merged without loss in quality. We illustrate with an example in Figure 3a. In the figure, orange nodes are
Low radius subgraphs. Let \( B_G(v,r) \) (referred to the ball of radius \( r \) at \( v \)) denote the set of nodes at distance at most \( r \) from \( v \) in the graph \( G \). It suffices to run the algorithm restricted to the balls centered around significant nodes. The balls are smaller than the full graph, so they can be processed faster; furthermore, they can be processed in parallel.
Remark. We use these two techniques to reduce the size of the input graph before running MAXWEIGHT. Our algorithm, FASTCOLCODENP, with these addition is shown in the supplementary material.

4 Algorithms for Parametric Scan Statistics and Extensions

In parametric scan statistics, each node \( v \) of the input graph has two weights associated with it: \( c(v) \) and \( b(v) \), which makes the problem more challenging than for non-parametric functions. Furthermore, there exist other score functions for graph anomaly detection where both nodes and edges have weights [4, 18]. Optimizing such functions reduces to the Prize Collecting Steiner Tree (PCST) problem [10], which is NP-Hard. We can extend the methods described above to these settings by keeping additional information in the dynamic program. We propose algorithm FASTCOLCODENP for parametric scan statistics and algorithm COLCODENW for PCST. Details of these algorithms may be found in the supplementary material, but we show experimental results for both in the next section.

5 Experiments

Our experiments address the following questions.
1. **Optimization power.** Do our algorithms find high-scoring subgraphs in real networks and synthetic benchmarks? How do they compare with existing methods?
(Section 5.3)

2. Event detection power. Do our algorithms correctly identify anomalous subgraphs? How do precision and recall compare with baselines? (Section 5.4)

3. Scalability. How do our algorithms scale to networks with more than $10^9$ nodes? (Section 5.5)

4. Performance in real datasets. How does the performance in real datasets compare with the worst case bounds? (Section 5.6)

For brevity, we focus on one scan statistic from each class as illustrative examples: (1) Berk-Jones (BJ) statistic [3] with $\alpha_{\text{max}} = 0.15$, (2) positively-elevated mean statistic (EMS) [17], and (3) the Heaviest Subgraph (HS) [4] and EventTree+ [18] functions, as examples of non-parametric, parametric, and generalized functions with edge weights, respectively.

5.1 Datasets. We use datasets from different domains, including social networks, infrastructure networks, and standard synthetic benchmarks. For these datasets, we have multiple instances, corresponding to snapshots of the networks at different times, and we have data of events in each snapshot. Additionally, we use datasets with planted anomalies for our scalability experiments. A summary of the datasets is provided in Table 4. See the supplementary material for more details.

Table 4: Datasets used in our experiments

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Description</th>
<th>Nodes</th>
<th>Edges</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email-EuAll</td>
<td>Email Network</td>
<td>242,472</td>
<td>340,750</td>
<td>5</td>
</tr>
<tr>
<td>Higgs-Retweet</td>
<td>Higgs Network</td>
<td>223,635</td>
<td>307,884</td>
<td>1</td>
</tr>
<tr>
<td>RealNet-PA</td>
<td>Traffic Network of Pennsylvania</td>
<td>1,037,972</td>
<td>1,341,398</td>
<td>1</td>
</tr>
<tr>
<td>Twitter</td>
<td>Twitter network collected through Twitter API</td>
<td>2,645</td>
<td>17,708</td>
<td>152</td>
</tr>
<tr>
<td>BWSN</td>
<td>Battle of the Water Sensors</td>
<td>12,827</td>
<td>14,831</td>
<td>22</td>
</tr>
<tr>
<td>PCST</td>
<td>Benchmark for the Prize Collecting Steiner Tree Problem</td>
<td>100 to 1000</td>
<td>254 to 1,504</td>
<td>31</td>
</tr>
</tbody>
</table>

5.2 Baseline Methods. We compare our proposed algorithm with 6 state-of-the-art methods that are organized in three categories:

(1) **NPHGS** [5]: A local search heuristic for optimizing the BJ statistic. (2) **AdditiveGraphScan** [23] and **DepthFirstScan** [22]: The state-of-the-art algorithms for optimizing parametric scan statistics that satisfy the Linear-Time-Subset-Scanning (LTSS) property. The EMS statistic also belongs to this category.

(3) **GraphLaplacian** [21] and **EdgeLasso** [20]: The representative methods for anomalous subgraph detection that optimize their own specific score functions, but are often considered as baseline methods in connected subgraph detection papers [17].

(4) **EventTree+** [18] and **MEDEN** [4]: For optimizing anomaly score functions with node and edge weights.

Parameter Tuning. The methods under evaluation, including ours, depend on user-specified parameters. We set $k$ to 10 or below for our algorithms. We discuss how we calibrate other parameters in the supplementary material.

5.3 Optimization Power

5.3.1 Non-Parametric Scan Statistics. We compare **FastColCodeNP** to other algorithms on the BJ statistic. In Table 1, we report the average BJ score obtained by each method, where the average is taken over all the instances in each dataset. We observe that **FastColCodeNP** achieves higher scores than all other methods. The difference in score is more pronounced in the NEast dataset, where **FastColCodeNP** more than doubles the score of **EdgeLasso** (EL) and **GraphLaplacian** (GL). We also note that **AdditiveGraphScan** has performance close to our algorithm, which is reasonable, since this method uses a sophisticated heuristic for Steiner connectivity problems.

5.3.2 Parametric Scan Statistics. Next, we compare **FastColCodeNP** to other methods with respect to the EMS function. Table 5 shows the average score for different datasets. We find that **FastColCodeNP** has the best performance in all datasets, except for NEast, where **AdditiveGraphScan** (GS) scores slightly higher.

Table 5: Parametric scan statistics optimization. We evaluate different methods with respect to the EMS function. In almost all datasets, **FastColCodeNP** has better performance than existing methods.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Elevated Mean Scan Statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FastColCodeNP</td>
</tr>
<tr>
<td>CitHepPh</td>
<td>43.611</td>
</tr>
<tr>
<td>NEast</td>
<td>41.903</td>
</tr>
<tr>
<td>Traffic</td>
<td>11.763</td>
</tr>
<tr>
<td>Twitter</td>
<td>23.019</td>
</tr>
<tr>
<td>BWSN</td>
<td>109.097</td>
</tr>
</tbody>
</table>

5.3.3 Functions with Node and Edge Weights. We also test our algorithm on two objective functions for event detection that consider edge weights in addition to node weights: the Heaviest Subgraph (HS) [4] and EventTree+ [18] problems. The methods proposed in these two works are **MEDEN** [4] and **GreedyT** [18], respectively. Both problem formulations reduce to the Prize Collecting Steiner Tree (PCST) problem [10]. We use our framework to design an algorithm for the PCST objective; we call this algorithm **ColCODENW**, and...
we compare it in terms of objective score to MEDEN and GreedyT on the PCST benchmark of [10]. For MEDEN, we convert the PCST instances to HS instances and run the TopDown heuristic described in [4]. For GreedyT, we convert the instances to a complete graph where an edge between two nodes has weight equal to the shortest path between the nodes, as described in [18]. Figure 4 shows the scores of the two heuristics relative to the score of ColCodeNW. Our algorithm finds subgraphs of higher quality than the heuristics. The score is as much as 4 times higher compared to GreedyT and 1.5 times higher compared to MEDEN.

5.4 Event Detection Power. Now, we evaluate FASTColCodeNP in terms of event detection power. We use the ground truth provided with the BWSN dataset, and we evaluate in terms of accuracy, precision, recall, and the F1 score. Let \( R \) be the set of nodes in the anomalous subgraphs and let \( S \) be the detected subgraph; then, we define

1. Accuracy\((R, S) = \frac{|R \cap S|}{|R|}\),
2. Precision\((R, S) = \frac{|R \cap S|}{|S|}\),
3. Recall\((R, S) = \frac{|R \cap S|}{|R|}\), and
4. F1 score = \(2 \frac{\text{Precision}(R, S) \cdot \text{Recall}(R, S)}{\text{Precision}(R, S) + \text{Recall}(R, S)}\)

In order to assess the performance of our method under noise, we introduce a random percentage of uniform noise in each instance. For a given noise level \( l \), each non-significant node becomes significant with probability \( l \).

In Table 6, we compare the performance of FASTColCodeNP with other algorithms at different noise levels. As in the previous section, we observe that our algorithm achieves higher objective scores compared to other methods, even when noise is present. As for the event detection power, FASTColCodeNP has higher accuracy and recall for all the noise levels and higher F1 score at almost every level. Finally, we note that the results in this section provide evidence that better objective scores also lead to better detection power, thus the importance of algorithms with good theoretical bounds.

5.5 Scalability. With the techniques presented in Section 3.4, we are able to run FASTColCodeNP in networks with over one million nodes. We note that in previous work only networks of up to 80,000 nodes have been considered. In Table 7, we compare our algorithm to existing methods in terms of objective score and running time. First, we compare FASTColCodeNP to the best-performing heuristic: AdditiveGraphScan (GS). We note that both methods achieve the same score in all datasets, but the running time of the latter is one order of magnitude larger, and it didn’t complete after 10 hours for the road network. Second, we observe that GraphLaplacian (GL) does not scale to large networks due to the fact high time and space complexity of constrained quadratic programming methods. We also note that our algorithm is much faster on the road network than on the other two because nodes in this planar network have low degree, thus making our low-radius technique more effective. Finally, EdgeLasso, DFS, and NFHG are faster than FASTColCodeNP; however, the scores obtained are significantly lower than using our algorithm.

Table 7: Performance-runtime tradeoff in large datasets for non-parametric scan statistic evaluation.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Email-EnAll</th>
<th>Higgs-Boson</th>
<th>RoadNet-PA</th>
</tr>
</thead>
<tbody>
<tr>
<td>FASTColCodeNP</td>
<td>839.18 (1,011)</td>
<td>830.18 (32,540)</td>
<td>421.16 (5)</td>
</tr>
<tr>
<td>AdditiveGraphScan</td>
<td>830.18 (32,540)</td>
<td>421.16 (5,655)</td>
<td>421.16 (5)</td>
</tr>
<tr>
<td>Berk-Jones Scan Statistic</td>
<td>839.18 (32,540)</td>
<td>421.16 (5,655)</td>
<td>421.16 (5)</td>
</tr>
<tr>
<td>GraphLaplacian</td>
<td>839.18 (32,540)</td>
<td>421.16 (5,655)</td>
<td>421.16 (5)</td>
</tr>
<tr>
<td>EdgeLasso</td>
<td>839.18 (32,540)</td>
<td>421.16 (5,655)</td>
<td>421.16 (5)</td>
</tr>
<tr>
<td>DFS</td>
<td>24.66 (2)</td>
<td>421.16 (5,655)</td>
<td>421.16 (5)</td>
</tr>
<tr>
<td>NFHG</td>
<td>24.66 (2)</td>
<td>421.16 (5,655)</td>
<td>421.16 (5)</td>
</tr>
</tbody>
</table>

In Figure 1, we show the scalability of all the algorithms we consider as a function of graph size in the Random dataset. FASTColCodeNP is faster than other methods for graph of size \(10^4\) and above, and it was the only algorithm to run to completion on graphs with one million nodes within 24 hours.

5.6 Performance guarantees in real datasets.

Graph refinement. Our graph refinement operation reduces the number of significant nodes in real datasets to less than a third of the original number. In Table 8, we report the number of significant nodes before and after graph refinement for \( \beta \in \{1, 0.95, 0.90\} \). Each dataset initially contains hundreds of significant nodes, with Twitter being close to 1,000. However, after graph refinement, this number goes down to less than 100. With approximate refinement, we are able to further reduce the effective number of significant nodes, down to a single digit in most cases. Because there are only a few significant supernodes, solutions of high score are small. In fact, if we set \( k \) to 10 or less in FASTColCodeNP, we are able to discover solutions of higher scores than previous methods (Section 5.3).

Convergence in few iterations. Algorithm FASTColCodeNP uses \( \ell = e^k \log n^2 / c \) random colorings to
Table 6: Average precision, recall, F1 score, accuracy, and objective value at different levels of noise.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 Score</th>
<th>Accuracy</th>
<th>Objective Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS</td>
<td>0.964</td>
<td>0.960</td>
<td>0.957</td>
<td>0.953</td>
<td>0.903</td>
</tr>
<tr>
<td>EL</td>
<td>0.958</td>
<td>0.954</td>
<td>0.951</td>
<td>0.947</td>
<td>0.898</td>
</tr>
<tr>
<td>GL</td>
<td>0.976</td>
<td>0.972</td>
<td>0.970</td>
<td>0.968</td>
<td>0.936</td>
</tr>
<tr>
<td>FastColCodeNP</td>
<td>0.978</td>
<td>0.975</td>
<td>0.974</td>
<td>0.971</td>
<td>0.909</td>
</tr>
</tbody>
</table>

Table 8: Number of significant nodes with graph refinement.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>W(S, α) (α = 0.15)</th>
<th>Graph Refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CitHepPh</td>
<td>624</td>
<td>β = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>β = 0.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td>β = 0.90</td>
</tr>
<tr>
<td>NEast</td>
<td>65</td>
<td>20</td>
</tr>
<tr>
<td>Traffic</td>
<td>157</td>
<td>61</td>
</tr>
<tr>
<td>Twitter</td>
<td>991</td>
<td>80</td>
</tr>
<tr>
<td>BWSN</td>
<td>333</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 5: Score of the solution found (normalized by that of the optimum) as a function of the number of iterations.

6 Application

We use our methods for event detection in Twitter follower graphs of Mexico and Venezuela. We model interactions—i.e., retweets and replies—between users as Poisson counts. The weight of an edge is given by $-\log(p(n_e^t|n_e^{t-1})/\mu$, where $p(n_e^t|n_e^{t-1})$ is the posterior probability of seeing $n_e^t$ interactions given the counts in the previous days, and $\mu = 0.05$ is a significance threshold. This weighing function [14] is positive-increasing if the posterior probability is less than $\mu$ and negative-decreasing otherwise. After assigning weights to edges, we solve the Heaviest Dynamic Subgraph problem [4] using algorithm ColCodeNW (See Section 5.3.3). For Venezuela, the temporal anomalous subgraph spans the time period between January 4, 2014 and March 31, 2014, which was a time of nationwide protests against the central government of that country. We also extracted the tweets corresponding to the heavy subgraphs, and we find that these tweets contain chatter about important national events. Figure 6 shows one such event. The predominant terms of the tweets relate to a protest organized in the city of Tachira demanding the liberation of local students, who had been put in jail in the previous days for protesting. In the supplementary material, we show results for the Twitter follower network of Mexico.

7 Related Work

There is a very large body of work related to our paper because of the wide range of applications. For brevity, we only discuss some of the specific papers that deal with scan statistics in Table 9, and give more detailed comparison in the online supplementary material. We also refer to the comprehensive survey by Akoglu et al. [1] for general discussion on anomaly detection. As shown in Table 9, our method is the first to give rigorous guarantees for a large class of scan statistics.

8 Conclusions

We present a unified framework for optimizing a broad class of graph scan statistics with connectivity constraints, with the following novel characteristics: (1) it gives rigorous guarantees for a large class of parametric and non-parametric score functions, (2) it can be scaled to large graphs with over a million nodes, and our methods will lead to direct improvements in performance quality for other uses of graph scan statistics in different applications.

Acknowledgements. The work of Jose Cadena and Anil Vullikanti has been partially supported by the following grants: DTRA CNIMS Contract HDTRA1-11-D-0016-0010, NSF BIG DATA Grant IIS-1633028 and NSF DIBBS Grant ACI-1443054.

References
<table>
<thead>
<tr>
<th>Algorithms based on optimization of parametric scan statistics in networks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exact algorithms</strong></td>
</tr>
<tr>
<td>Simulated annealing</td>
</tr>
<tr>
<td>AdditiveGraphScan</td>
</tr>
<tr>
<td>EdgeLasso</td>
</tr>
<tr>
<td>GraphLaplacian</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithms based on optimization of nonparametric scan statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NPHS</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reduction to variants of Network Design and using methods for Prize-Collecting Steiner Tree (PCST)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EventTree</strong></td>
</tr>
<tr>
<td><strong>Meden</strong></td>
</tr>
</tbody>
</table>

| Our algorithm                                                 | gives optimal solution for a large number of parametric and non-parametric scan statistics with effective solution size parameter $\tau$, with probability at least $(1 - \epsilon)$, and runs in time $O((2\epsilon)^{-1} \cdot m \log \frac{n}{\epsilon})$ |

9 Supplementary Material

We present proof details, pseudocode, and extensions of the methods presented in Section 3. For brevity, some proofs have been omitted, but they can be found in an online extended version of this paper [6], along with the code and datasets used in the experiments.

9.1 Additional Details for Section 2

9.1.1 Scan Statistics Functions. Table 10 lists examples of parametric and non-parametric scan statistics that can be optimized using our framework. We list more functions that fall under our framework in the online version.

9.1.2 Hardness Because of connectivity constraints, optimizing scan statistics on graphs is challenging in general. Note that this contrasts with the case without any connectivity requirement, where the optimal value of the scan statistic can be computed in polynomial time because of a linear ordering property [16].

Lemma 9.1. Given an instance \((G = (V, E), \alpha_{max})\) of non-parametric scan statistics, finding a connected set \(S \subseteq V\) that maximizes the score \(F(S)\) is NP-complete.

Lemma 9.2. Given an instance \((G = (V, E), b(\cdot), c(\cdot))\) of parametric scan statistics, finding a connected set \(S \subseteq V\) that maximizes the score \(F(S)\) is NP-complete.

9.2 Additional details for Section 3

Proof. (of Lemma 3.2) Suppose \(M(v, T)\) is achieved for a connected set \(S\), such that \(|S| = |T|\) and \(\{col(u) : u \in \bar{S}\} = T\), with \(M(v, T) = \sum_{i \in S} w(i, \alpha)\). We claim that there exists \(u \in Nbr(v)\), and partitions \(T = T_1 \cup T_2\), and \(S = S_1 \cup S_2\), such that: (1) \(M(v, T) = M(v, T_1) + M(u, T_2)\), (2) \(\{col(i) : i \in S_1\} = T_1\) and \(\{col(i) : i \in S_2\} = T_2\), and (3) \(S_1\) and \(S_2\) are connected. Since \(S\) is connected, there exists a tree \(H\) that spans \(S\) and contains node \(v\). Further, there must exist a node \(u \in Nbr(v)\) such that \((u, v) \in E\), since \(|T| = |H| \geq 2\). Let \(H_1\) and \(H_2\) be the trees rooted at nodes \(u\) and \(v\), respectively, that result when edge \((u, v)\) is deleted in \(H\). Let \(S_1\) and \(S_2\) denote the sets of nodes in \(H_1\) and \(H_2\), respectively. Let \(T_1\) and \(T_2\) be the colors used by \(S_1\) and \(S_2\), respectively. By construction, we have \(M(v, T) = M(v, T_1) + M(u, T_2)\), so that the partitions \(S = S_1 \cup S_2\) and \(T = T_1 \cup T_2\) satisfy the requirements mentioned earlier. Therefore, the recurrence follows.

9.2.1 Scaling We take advantage of a locality property of non-parametric scan statistics. Namely, if we are given a set \(S\) with score \(F(S)\), we can increase the score of the set by adding any significant node that is a neighbor of a node already in \(S\).

Lemma 9.3. Let \(G(V, E)\) be a network, and let \(F(\cdot)\) be a non-parametric scan statistic function. Given a set \(S \subseteq V\), suppose there exists a significant node \(u \notin S\) and an edge \((u, v) \in E\), for some \(v \in S\). Then, \(F(S \cup \{u\}) \geq F(S)\).

Proof. Non-parametric scan statistics are increasing on \(W(S)\). Since \(u\) is significant, we have that
\[
\frac{W(S \cup \{u\})}{N(S \cup \{u\})} = \frac{W(S) + 1}{N(S)} \geq \frac{W(S)}{N(S)},
\]
and the proof follows.

Exact refinement. An implication of Lemma 9.3 is that we can collapse components of significant nodes into a single node prior to running ColCodeNP. We propose a graph refinement to reduce the total number of significant nodes in a graph. Given a network \(G(V, E)\), \(p\)-values for the nodes in \(V\), and a significance level \(\alpha\), we define \(V_1, \ldots, V_r\) as the connected components (or supernodes) of \(G\) induced by the set of significant nodes. We create a new graph \(H(V', E')\), whose node set consists of \(V_1, \ldots, V_r\) and the non-significant nodes in \(G, V \setminus \bigcup_{i=1}^r V_i\). Edges between non-significant nodes are preserved in \(H\), and we put an edge from a non-significant node \(u\) to \(V_i\) if the edge \((u, v)\) exists, for some \(v \in V_i\). Finally, we create a vector of weights, \(w'\). The weight of a node in \(H\) is \(|V_i|\) for each supernode \(V_i\), and 0 otherwise. This procedure is equivalent to removing all the significant nodes and replacing them with the respective supernode \(V_i\). Figure 3a in the main text illustrates the procedure. After this preprocessing step, we run procedure MAXWEIGHT for the instance \((H(V', E'), w')\).

Lemma 9.4. Let \(((\mathcal{G}(V, E), \alpha), k, \epsilon)\) be an input to COLCODENP, and let \(S_G^\epsilon\) be the solution returned by the algorithm. Similarly, let \(((H(V', E'), \alpha), k, \epsilon)\) with weights \(w'\) be the corresponding instance generated by graph refinement, and let \(S_H^\epsilon\) be the solution returned by COLCODENP in this instance. Then, \(F(S_H^\epsilon) = F(S_G^\epsilon)\). Furthermore, suppose \(|S_G^\epsilon| = k\); then, \(|S_H^\epsilon| = k'\), for some \(k' \leq k\), so it is possible to execute COLCODENP with parameter \(k'\) and still obtain \(F(S_H^\epsilon) = F(S_G^\epsilon)\).

Proof. The lemma follows from Lemma 9.3.

Approximate refinement. It is possible to further compress the graph by including non-significant nodes into the components described above. As before, we first obtain connected components of significant nodes, \(V_1, \ldots, V_r\), but now, we keep adding nodes to a component \(V_i\) as long as the number of significant nodes is at least \(\beta|V_i|\), where \(\beta\) is a parameter between 0 and 1.
Table 10: Scan Statistics Functions that can be optimized with our framework.

<table>
<thead>
<tr>
<th>Non-Parametric Scan Statistics</th>
<th>General Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berk-Jones [3] ( F(S) = \max_{a \leq a_{max}} N(S)K_L(\frac{W(S,a)}{N(S)}, a) )</td>
<td>( \phi(a, b, a) = b \cdot K_L(a/b, a) ), where ( K_L(x, a) = x \log \left( \frac{x}{a} \right) + (1 - x) \log \left( \frac{1}{1-x} \right) )</td>
</tr>
<tr>
<td>Higher Criticism [7] ( F(S) = \max_{a \leq a_{max}} \frac{W(S,a) \cdot N(S)}{\sqrt{N(S)}(1-\alpha)} )</td>
<td>( \phi(a, b, a) = (a - b \cdot a) / \sqrt{b \cdot (1-\alpha)} )</td>
</tr>
<tr>
<td>Kolmogorov-Smirnov [19] ( F(S) = \max_{a \leq a_{max}} \sqrt{N(S)} \cdot \left( \frac{W(S,a)}{N(S)} - a \right) )</td>
<td>( \phi(a, b, a) = \sqrt{b} \cdot (\frac{a}{\alpha} - 1) )</td>
</tr>
<tr>
<td>Anderson-Darling [8]</td>
<td>( F(S) = \max_{a \leq a_{max}} \sqrt{N(S)} \cdot \left( \frac{W(S,a)}{N(S)} - a \right) )</td>
</tr>
<tr>
<td>Stochastic Ordering of ( p )-Values [3]</td>
<td>( F(S) = N(S) \int_{a}^{a_{max}} \frac{W(S,a)N(S) - a}{N(S)} \cdot \frac{1}{\alpha(1-\alpha)} ) da ( N(S) = \sum_{a \leq a_{max}} I(p(v) \leq \alpha) )</td>
</tr>
<tr>
<td>Truncated Fisher’s Test</td>
<td>( F(S) = \max_{a \leq a_{max}} \frac{W(S,a) \cdot N(S) - a}{\sqrt{N(S)}(1-\alpha)} )</td>
</tr>
<tr>
<td>Stouffer’s Test [18]</td>
<td>( F(S) = \sum_{a \leq a_{max}} \frac{1 - \Phi^{-1}(1-p(v))}{\sqrt{N(S)}} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parametric Scan Statistics</th>
<th>General Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive Elevated Mean Scan Statistic [17]</td>
<td>( F(S) = \sum_{v \in S} x_i / \sqrt{N(S)} )</td>
</tr>
<tr>
<td>Elevated Mean Scan Statistic [17]</td>
<td>( F(S) = \left( \sum_{v \in S} x_i \right)^2 / N(S) )</td>
</tr>
<tr>
<td>Expectation-based Poisson Scan Statistic [16]</td>
<td>( F(S) = C(S) \log(C(S)/B(S)) + B(S) - C(S) )</td>
</tr>
<tr>
<td>Kulldorff Scan Statistic [11]</td>
<td>( F(S) = \frac{1}{C(S)} \log \left( \frac{(1-C(S))}{C(S)} \right) + \left( 1 - \frac{C(S)}{B(S)} \right) \cdot \left( 1 - \frac{C(S)}{B(S)} \right) )</td>
</tr>
<tr>
<td>Expectation-based Gaussian Scan Statistic [16]</td>
<td>( F(S) = \left( C(S) - B(S) \right)^2 / (2B(S)) )</td>
</tr>
</tbody>
</table>

We show an example in Figure 3b. By doing this, we are able to reduce the number of anomalous supernodes. We may not find the optimal solution now; however, we can control the error with the parameter \( \beta \).

**Lemma 9.5.** Denote the number of nodes significant at level \( \alpha \) in a set \( S \) as \( N_\alpha(S) \). Let \( S^* \) be the set that maximizes \( F \) and let \( r(S^*) = \frac{N_\alpha(S^*)}{N(S^*)} \). There is a solution on the instance \( H \) with ratio \( r(S) \geq \beta r(S^*) \).

**Proof.** We split the set \( S^* \) into significant nodes, \( N_\alpha(S^*) \), and non-significant nodes, \( N_{\alpha}(S^*) \), such that \( N(S) = N_\alpha(S^*) + N_{\alpha}(S^*) \). We now show that, in the instance \( H \), there exists a set \( S \) with ratio

\[
r(S) = \frac{N_\alpha(S)}{N(S)} \geq \frac{N_\alpha(S^*)}{N_\alpha(S^*) + N_{\alpha}(S^*) + \frac{1-\beta}{\beta} N_\alpha(S^*)}.
\]

We define \( S' \) as the set formed by the supernodes \( V_i \) corresponding to the significant nodes in \( S^* \), and we note that \( N_\alpha(S') \geq N_\alpha(S^*) \); for simplicity, we assume equality. By construction, the cardinality of \( S' \) is \( N(S') = N_\alpha(S') + \frac{1-\beta}{\beta} N_\alpha(S^*) \). Note that the nodes in \( S^* \) may be disconnected; however, we can connect them using a set of anomalous nodes \( S'' \) of size at most \( N^{-\alpha}_\alpha(S^*) \). Finally, we form a set \( S = S' \cup S'' \) that has the desired ratio.

To conclude the proof, we compute \( r(S)/r(S^*) \):

\[
\frac{r(S)}{r(S^*)} = \frac{N_\alpha(S)}{N_\alpha(S') + N_{\alpha}(S') + \frac{1-\beta}{\beta} N_\alpha(S^*)} \geq \frac{N_\alpha(S')}{N_\alpha(S'') + \frac{1-\beta}{\beta} N_\alpha(S^*)} = \frac{1}{1 + \frac{1-\beta}{\beta} N_\alpha(S')}.
\]

Noticing that \( N^{-\alpha}_\alpha(S^*) \leq 1 \), we obtain \( r(S) \geq \beta r(S^*) \).

**9.3 Extensions to Parametric Scan Statistics.**

In Algorithm 3, we describe CoLDEP for parametric scan statistics maximization. For these functions, each node \( v \) of the input graph has two weights associated with it: \( C(v) \) and \( B(v) \). Therefore, we need a more
Algorithm 2 \textsc{FastColCodenp}\((G(V,E), \alpha_{\text{max}}, k, \epsilon, \beta)\).

\textbf{Input:} Instance \((G(V,E), \alpha_{\text{max}})\), parameters \(k, \epsilon, \beta\)
\textbf{Output:} Set \(S^*\) with score \(OPT(F,k)\)

Let \(A\) be the set of \(p\)-values of nodes in \(V\) below \(\alpha_{\text{max}}\).

\textbf{for} \(i \in A\) \textbf{do}

Perform approximate refinement with parameter \(\beta\).

Let \(H = (V',E')\) be the refined graph with weights \(w'\) for each connected subset \(S\) of \(V\) such that \(\sum_{v \in S} w'(v)\) is constant.

\(S^* = \arg\max_{i \in [1,k]} F(S_i^*)\)

\textbf{return} \(S^*\)

\textbf{end for}\n

The general algorithm for \textsc{ColCodenp}. Analogous to Lemma 3.1, we make use of the following property:

\textbf{Lemma 9.6.} The parametric scan statistics functions characterized by equation \((2.2)\) are increasing functions of \(C(S)\) if \(C(S) > B(S)\) and \(B(S)\) is constant.

Given a graph \(G(V,E)\) and vectors \(C\) and \(B\), let \(M(v,T,j)\) be the maximum value of \(C(S)\) over all connected subsets \(S\), such that \(1) v \in S\), \(2) S\) is colorful with respect to \(T\), and \(3) B(S) = j\). Here \(j\) ranges from 1 to \(B(V)\). \(M(v,T,j)\) can be computed by a dynamic program with a constant factor in terms of the approximation. We define \(\mu = \delta B_{\text{max}}/n\), for some \(\delta > 0\). Then, we define a vector \(B'\), where \(b'(v) = \lfloor b(v)/\mu \rfloor\), for each node \(v\). By invoking \textsc{ColCodenp} on the instance \((G(V,E), C', B')\), we obtain a \((1 + \delta)\) approximation on the weight of \(S^*\).

\textbf{3.2 Scaling} There is a notion of graph refinement for parametric scan statistics analogous to the one presented above. We note that the parametric functions in Table 10 are increasing on the ratio \(r(S) = C(S)/B(S)\). Therefore, if we are given a set \(S\) with score \(F(S)\), we can increase the score of the set by adding any node that is a neighbor of a node already in \(S\) as long as \(r(S)\) does not decrease. This idea is formalized in the following lemma.

\textbf{Lemma 9.8.} Let \(G(V,E)\) be a network, and let \(F()\) be a parametric scan statistic function. Given a set \(S \subseteq V\), suppose there exists an node \(u \in S\) and an edge \((u,v)\) \(v \in E\), for some \(v \in S\), such that \(C(S \cup \{u\})/B(S \cup \{u\}) \geq C(S)/B(S)\); then, \(F(S \cup \{u\}) \geq F(S)\).

\textbf{Exact refinement.} For parametric scan statistics, the exact refinement consists of merging nodes into components as long as the ratio \(r(S)\) of the component does not decrease. Given a network \(G(V,E)\) and event \((c(v))\) and population \((b(v))\) counts for every \(v \in V\), we maintain a list of components or supernodes \(Y = V_1, \ldots, V_r\) and the graph \(H(V',E')\) induced by those.
There is an edge between \( V_i \) and \( V_j \) if there exists nodes \( v_i \in V_i \) and \( v_j \in V_j \), such that \( v_i \) and \( v_j \) are neighbors in \( G \). Initially, every node is its own component. The exact refinement iterates through the list of current components trying to merge them until no more merges are possible. In each iteration, we first sort the current components in descending order of ratio \( r(S) \). Then, in that order, we merge a component with its neighbors if the ratio does not decrease. After this exact refinement, we run Algorithm 3 for the instance \((H(V', E'), C', B')\), where \( C' \) and \( B' \) are the event and population counts of the supernodes, respectively.

**Approximate refinement.** We can obtain larger components by allowing merges that decrease the ratio of the component. Our approximate refinement procedure takes two parameters: \( \beta \in [0, 1] \) and \( \delta > 1 \). We allow a component \( V_i \) to grow as long as two constraints are not violated:

1. **Population size constraint.** The population size of \( V_i \) is at most \( \delta \) times the smallest population size in the component: \( B(V_i) \leq \delta \min_{v \in V_i} b(v) \).

2. **Event size constraint.** The event size of \( V_i \) is at least \( \beta \delta \) times the largest event size in the component: \( \frac{C(V_i)}{s} \geq \beta \max_{v \in V_i} c(v) \).

**Lemma 9.9.** Let \( S^* \) be the set that maximizes a parametric scan statistic \( G \) and let \( r(S^*) = \frac{C(S^*)}{B(S^*)} \). There is a solution \( S \) on the instance \( H \) constructed as above with ratio \( r(S) \geq \beta r(S^*) \).

**Proof.** Analogous to Lemma 9.5.

### 9.3.3 Functions with Node and Edge Weights

Both the Heaviest Subgraph [4] and EventTree+ problems [18] reduce to Prize Collecting Steiner Tree (PCST) with NetWorth objective [10]. In PCST, we are given a graph \( G(V, E) \) with non-negative node prizes, \( \pi \), and non-negative edge costs, \( w \), and the goal is to find a tree \( S(V(S), E(S)) \) that maximizes the NetWorth objective:

\[
W(S) = \sum_{v \in V(S)} \pi(v) - \sum_{e \in E(S)} w(e).
\]

Using our framework, we can design an algorithm to find a tree with maximal NetWorth and size up to \( k \), where \( k \) is a parameter. This implies an algorithm for HS and EventTree+.

Let \( M(v, T) = \max_S W(S) \), where the maximization is over all connected and colorful sets \( S \subseteq V \), such that \( v \in S \), \( |S| = |T| \), and \( \{\text{col}(u) : u \in S\} = T \). \( M(v, T) \) can be computed by a dynamic program:

**Lemma 9.10.** Let \( M(v, T) \) be defined as above. For any node \( v \) and color \( s \), \( M(v, \{s\}) = \pi(v) \) if \( \text{col}(v) = s \), else \( M(v, \{s\}) = -\infty \). If \(|T| \geq 2\),

\[
M(v, T) = \max_{u \in \text{nbr}(v)} \{M(v, T_1) + M(u, T_2)\},
\]

where the maximum is over all partitions \( T_1 \cup T_2 \) of the set \( T \) and all neighbors \( u \) of \( v \).

**Proof.** Analogous to Lemma 3.2.

### 9.4 Additional details for Section 5

#### 9.4.1 Datasets

For the optimization event detection power experiments, we use the following datasets, for which we have temporal event data:

**CitHepPh\(^1\).** This is a network of scientific collaborations between authors of papers submitted to the High Energy Physics - Phenomenology category of arXiv. The p-value of each node \( v \) for a specific snapshot was calculated as the ratio of nodes in the current graph snapshot whose citations are greater than or equal to the citations of this node.

**NEast [12].** The Northeastern USA Benchmark is a well-known dataset in the spatial scan statistics community. The benchmark contains census information of 245 counties as well as synthetically generated cases of a disease.

**Traffic\(^{2}\).** The highway network of Los Angeles County, California and its activity on May, 2014. Nodes in the graph are sensors that record traffic statistics, such as average speed and the number of vehicles passing through. We assume a normal distribution for the average speed recorded by each sensor. In each snapshot \( t \), the p-value of a node \( v \) is the cumulative distribution function of a normal distribution with mean \( x_v^{[1, t-1]} \) and standard deviation \( \sigma_v^{[1, t-1]} \), where \( x_v^{[1, t-1]} \) and \( \sigma_v^{[1, t-1]} \) are, respectively, the sample mean and standard deviation for node \( v \) from snapshots 1 to \( t-1 \).

**Twitter.** A sample of the follower graph of Venezuela collected between July 1, 2013 and December 31, 2013. We assign p-values based on the tweeting behavior of the users. Formally, let \( x_u^t \) be the number of tweets generated by node \( u \) at time \( t \); we model \( x_u^t \) as a draw from a Poisson distribution with parameter \( \lambda_u \). We take a Bayesian approach and consider \( \lambda_u \) to be drawn from a Gamma distribution with parameters \( \alpha_u \) and \( \beta_u \). These parameters are updated as we see new data every snapshot. The p-value of a node at time \( t \) is its posterior probability \( p(x_u^t | x_u^{[1, t-1]} \) \), which follows a negative binomial distribution by our choice of prior.

**Battle of the Water Sensor Networks (BWSN) [15].** This dataset is a benchmark originally used to

\(^1\)https://snap.stanford.edu/data/cit-HepPh.html

\(^2\)http://pems.dot.ca.gov/
evaluate different sensor network designs in terms of early detection of contaminants in a water system. The dataset includes “ground truth” subgraphs representing parts of the network that are contaminated, which we use for evaluation in Section 5.4.

PCST [10]. A standard benchmark to evaluate algorithms for the Prize Collecting Steiner Tree Problem. We use the “K” instances of the benchmark to evaluate methods that reduce to PCST (Section 5.3.3).

For the scalability experiments (Section 5.5), we consider three large networks (i.e., over $10^9$ nodes) from the SNAP repository [13]. Since we do not have data of events in these networks, we plant events according to the statistical assumptions of the BJ scan statistic. First, we assign $p$-values uniformly at random to all the nodes in the graph. Then, we select 5 seed nodes and their neighbors in the graph, and we assign a $p$-values less than 0.15 to these nodes. We also generated a set of Erdos-Renyi graphs (Random in Table 4) with number of nodes $n \in \{10^2, 10^3, 10^4, 10^5, 10^6\}$ and number of edges $m = O(n)$. We first assign random $p$-values to all the nodes. Then, we plant an anomaly by selecting a node and its ego network at random and setting their $p$-values below 0.15.

9.4.2 Parameter tuning for baseline methods. When possible, we use the values prescribed by the authors of the method; this is the case with NPHGS, EventTree+, and MEDEN. In the case of GraphLaplacian and EdgeLasso, we tune the parameters separately for each dataset. In particular, we take a sample of 20 instances for each dataset and choose a parameter from \{0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1\} that maximizes the average score. Notice that the parameter for each dataset may be different.

9.4.3 Additional Details for Section 6. For Mexico, the two most common words in the extracted tweets form the phrase “energy reform” referring to a bill recently proposed by Mexican president Enrique Pena Nieto that would have a significant impact in the economy of the country (Figure 7).

Figure 7: Examples of events found in the heavy subgraphs for Venezuela (top) and Mexico (bottom)

References