Fast Graph Scan Statistics Optimization Using Algebraic Fingerprints

Jose Cadena, Saliya Ekanayake, and Anil Vullikanti
Network Dynamics and Simulation Science Laboratory (NDSSL), Virginia Tech
Email: {jcadena,esaliya,vsakumar}@vt.edu

***Abstract*—Graph scan statistics have become popular for event detection in networks. This methodology involves finding connected subgraphs that maximize a certain anomaly function, but maximizing these functions is computationally hard in general. We develop a novel approach for graph scan statistics with connectivity constraints. Our algorithm APPROX-MULTILINEARSCAN relies on an algebraic technique called multilinear detection, and it improves over prior methods for large networks. We also develop a Pregel-based parallel version of this algorithm in Giraph, MULTILINEARSCANGIRAPH, that allows us to solve instances with over 40 million edges, which is more than one order of magnitude larger than existing methods.**

I. INTRODUCTION

Many methods have been proposed for detecting anomalies or "hotspots" in a graph; among those, *graph scan statistics* has become one popular approach in recent years [1], [2], [3]. At a high level, a scan statistic is a function that assigns an "anomalousness" score to a subgraph, and thus the anomaly detection problem corresponds to maximizing a suitably defined function over all *connected subgraphs*—informally, we want to find the most "interesting" subgraph. However, finding subgraphs of this kind generalizes NETWORK DESIGN problems [4], which are very challenging constrained optimization problems. To cope with the complexity, several heuristics have been developed over the years, and, while heuristics perform well for certain types of graphs and particular functions, they do not give any guarantees on the quality of the solution in general, which is important in practice because suboptimal solutions affect the detection power [5]. A different approach recently proposed in [5] is based on designing *fixed parameter tractable* algorithms, which is becoming a popular way of handling NP-complete problems [6]. This gives optimal solutions for a large class of scan statistics, but constrained to an "effective" size of k, in time $O((2e)^k m)$, where m is the number of edges. *All previous methods have only been shown to scale to networks with less than a few million edges*. There is only one parallel algorithm [7], to the best of our knowledge, but even that only scales to similar size networks.

We significantly improve these results by designing novel sequential and parallel algorithms for optimizing graph scan statistics. Our contributions are:

1. Efficient algorithms for graph scan statistics with guarantees. We propose APPROX-MULTILINEARSCAN, which adapts an algebraic technique for detecting multilinear terms in multivariate polynomials. Our method can maximize a large class of scan statistics with connectivity constraints, while having a much lower memory overhead than the methods of [5], We also develop a Pregel-based distributed algorithm MULTILINEARSCANGIRAPH, which allows us to scale to very large instances with over 40 million edges.

2. Experimental results. We evaluate our algorithms on different real and synthetic networks, and we observe significant improvement over all prior sequential and parallel algorithms. MULTILINEARSCANGIRAPH improves over FASTCOLCODENP [5] by *more than three orders of magnitude in running time and 1-2 orders of magnitude in memory usage.*

3. Case study: finding unvaccinated clusters. We illustrate our methods by applying them to the task of finding clusters of low-vaccination in Minnesota.

A preview of our results is presented in Figure 1. Complete details of our algorithms, including missing proofs, and additional results are presented in the full version [8].

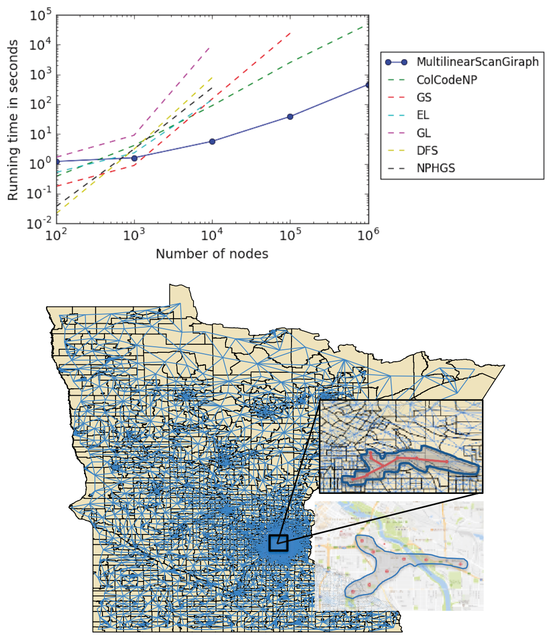


Fig. 1: Summary of results. (Top) Comparison of the running times for different graph sizes (Section IV-B). Our algorithm, MULTILINEARSCANGIRAPH, is a Pregel algorithm implemented in Giraph, has rigorous theoretical guarantees (Theorem 3), and scales to graphs with millions of nodes, in contrast with most previous methods. (Bottom) Using APPROX-MULTILINEARSCAN, we discover low-vaccination clusters in the census block group graph of Minnesota (Section IV-E).

II. Preliminaries

A. Graph Scan Statistics

We are given an undirected graph $G = (V, E)$, where V is a set of n vertices or nodes, and E is a set of m edges. Each vertex $v \in V$ has two values associated with it: (1) a *baseline count*, $b(v)$—for instance, the number of people in a county corresponding to node v—and (2) an *event count* or *weight*, $w(v)$—for instance, the number of cases of a disease of interest in county v.

The methodology of graph scan statistics formalizes anomaly detection as a hypothesis testing problem. Under the null hypothesis, it is *business as usual*, and the event counts for all nodes are generated proportionally to their baseline counts. Under the alternative hypothesis, there exists a small connected subset $S \subseteq V$ of vertices for which the counts are generated at a higher rate than outside S. Then, the goal is to find a set S that maximizes an appropriate scan statistic function $F(S)$, typically a log-likelihood ratio that compares event counts to baseline counts. We define a scan statistic in terms of the event and baseline counts of a node set:

$$F(S) = F(W(S), B(S), \theta),$$

where $W(S) = \sum_{v \in S} w(v)$ is the total event count or *weight* of S, $B(S) = \sum_{v \in S} b(v)$ is the baseline count of the set, and θ represents possible additional arguments to F.

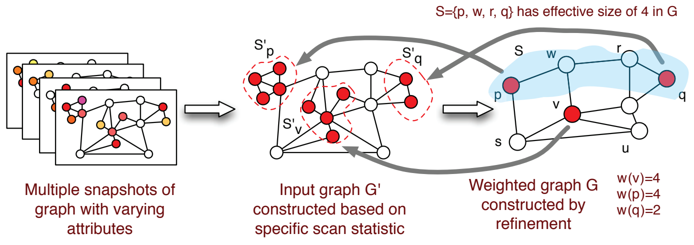
Depending on the assumptions that are satisfied by the data, there are two broad types of scan statistics: parametric and non-parametric. A well-known example of the former is the Kulldorff scan statistic commonly used in disease surveillance [9], [10], which assumes baseline counts drawn from a simple distribution, such as Poisson. See [8] for details of this statistic. For simplicity, we will focus on non-parametric functions in this paper; an example is the Berk-Jones scan statistic (BJ) [11] used for civil unrest events and network intrusion detection [12], [13]. Each node v has a p-value $p(v) \in [0, 1]$, and, for a significance level α the event count $w(v)$ is 1 if $p(v) < \alpha$ (i.e., the node is significant) and 0 otherwise, and the baseline count is $b(v) = 1$ for all nodes. This scan statistic is defined as

$$\max_{\alpha \le \alpha_{\max}} B(S) \left[\frac{W(S)}{B(S)} \log \left(\frac{W(S)/B(S)}{\alpha} \right) + \left(1 - \frac{W(S)}{B(S)}\right) \log \left(\frac{1 - W(S)/B(S)}{1 - \alpha} \right) \right].$$

B. Problem Formulation

From the discussion above, the graph anomaly detection task can be posed as the following constrained optimization problem: Given a graph $G^0 = (V^0, E^0)$, a scan statistic $F(\cdot)$, and the associated counts for vertices—represented by vectors w and b—find a connected subset $S^0 \subseteq V^0$ that maximizes $F(S^0) = F(W(S^0), B(S^0), \theta)$. For brevity, we will focus on non-parametric scan statistics in the rest of the paper, using the BJ statistic as a working example (our methods extend to all the other scan statistics described in [8]). As discussed above, for this function, $B(S^0) = |S^0|$ is the size of the set.

Finding solutions with constraints on the effective size. Following [5], we will consider connected subgraphs S^0 that maximize $F(S^0)$, with $|S^0| \le k^0$, where k^0 is a parameter. The



S={p, w, r, q} has effective size of 4 in G
S'p
S'q
S'v
S
w
r
p
q
v
s
u
Multiple snapshots of graph with varying attributes
Input graph G' constructed based on specific scan statistic
Weighted graph G constructed by refinement
w(v)=4
w(p)=4
w(q)=2

Fig. 2: Example of an input graph G^0 and the reduced graph G constructed using the refinement process of [5]. The anomalous nodes in G^0 are shown in red. (Super-)node p in G represents a set of four anomalous nodes in G^0, denoted by the set S^0_p of weight $\mathsf{W(p) = 4}$. The subset $\mathsf{S = \{p, w, r, q\}}$ in the graph G corresponds to a subset of size 8 in G^0, but it has effective size 4 in G.

graph G^0 can be transformed $G^0 \rightarrow G$ by merging connected subsets S^0_v of anomalous nodes in G^0 to form "supernodes" v in G using the refinement method discussed in [5]; we use $w(v)$ to denote the number of anomalous nodes in S^0_v. This is illustrated in Figure 4. We refer to $k_{\text{eff}} = |S|$ as the "effective solution size" for the set $\bigcup_{v \in S} S^0_v$, which has size $|\bigcup_{v \in S} S^0_v| \ge k_{\text{eff}}$. In the rest of the paper, we assume that we are given the preprocessed graph G as input, and the goal is to find a connected subset S that maximizes $F(S)$, with $|S| \le k_{\text{eff}}$; this will be denoted by k, in order to simplify the notation. Formally, the focus of the paper is the following problem:

Problem 1: Given a graph $G = (V, E)$, scan statistic $F(\cdot)$, node counts w and b, and parameter $k \ll |V|$, find a connected subset $S \subseteq V$ that maximizes $F(S)$ with $B(S) \le k$.

C. Multilinear Detection

Let $X = x_1, \ldots, x_n$ be a set of variables, and let $P(X)$ be a polynomial, which is a sum of monomials on X. An example of a polynomial on 4 variables is $P(x_1, x_2, x_3, x_4) = x_1^2 x_2 + x_1 x_2 x_3 + x_2 x_4^2$. A monomial is called *multilinear* or *square-free* if all its variables have exponent 1; its *degree* is the sum of the exponents of all its variables. For instance, $x_1 x_2 x_3$ is a multilinear monomial of degree 3. Given variables $X = x_1, \ldots, x_n$ and a polynomial $P(X)$, the goal in the k-Multilinear Detection (k-MLD) problem is to decide whether or not $P(X)$ has a multilinear monomial of degree exactly k. Note that $P(X)$ may have an arbitrary number of terms—i.e., exponential on n. Therefore, the problem is not as trivial as writing the polynomial explicitly and checking each term. Rather, we assume that $P(X)$ is given succinctly in a recursive form, and the decision has to be made without unrolling this recursion. For brevity, we assume some familiarity with group algebras; a more detailed discussion of the algebraic concepts used here is presented in the full version [8].

Algorithm for Multilinear Detection. The main idea in the algorithm of [14] is that, if we evaluate a polynomial over the "right" algebra, monomials that have square terms evaluate to $\bar{0}$ (which is the additive identity in the algebra), and the remaining terms, which are multilinear, do not cancel out, with high probability. Then, a polynomial $P(X)$ has a k multilinear term if $P(X) \neq \bar{0}$. Let $\mathbb{Z}_2^k$ be the group formed by all the k-dimensional binary vectors, and define the group multiplication operation as entry-wise XOR. For example, $\mathbb{Z}_2^2$ consists of the vectors $v_0 = (0, 0), v_1 = (0, 1), v_2 = (1, 0), v_3 = (1, 1)$.

Now, we define a group algebra $\mathbb{Z}_2[\mathbb{Z}_2^k]$. Each element in the group algebra is a sum of elements from $\mathbb{Z}_2^k$ with coefficients from $\mathbb{Z}_2$ (i.e., either 1 or 0): $\sum_{v \in \mathbb{Z}_2^k} a_v v$, where $a_v \in \{0,1\}$. The addition operator of the group algebra is

$$\sum_{v\in \mathbb{Z}_2^k} a_v v + \sum_{v\in \mathbb{Z}_2^k} b_v v = \sum_{v\in \mathbb{Z}_2^k} (a_v + b_v) v,$$

where the addition of the coefficients is modulo 2, and the multiplication is defined as

$$\left(\sum_{v\in \mathbb{Z}_2^k} a_v v\right)\left(\sum_{u\in \mathbb{Z}_2^k} b_u u\right) = \sum_{v\in \mathbb{Z}_2^k} (a_v \cdot b_u)(v \cdot u).$$

The key insight in [14] is that, for any $v_i \in \mathbb{Z}_2^k$, the square of the term $(v_0 + v_i) \in \mathbb{Z}_2[\mathbb{Z}_2^k]$ evaluates to $\bar{0}$:

$$(v_0+v_i)^2 = v_0^2 + 2(v_0\cdot v_i) + v_i^2 = v_0 + (0 \mod 2)v_i + v_0 = 2v_0 = \bar{0}.$$

Then, the algorithm of [14] is roughly as follows:

1) For each variable x_i, sample a vector v_i uniformly at random from $\mathbb{Z}_2^k$ and assign $x_i = (v_0 + v_i) \in \mathbb{Z}_2[\mathbb{Z}_2^k]$.
2) Evaluate the polynomial $P(x_1, \ldots, x_n)$ on this random assignment.
3) If $P(x_1, \ldots, x_n) \neq \bar{0}$ return "yes"; else, return "no".

The algorithm was later refined in [15] by using the group algebra $GF(2^{3+\log_2 k})[\mathbb{Z}_2^k]$, where $GF(p)$ is the finite field of order p [16]. A polynomial $P(x_1, \ldots, x_n)$ with variables from $GF(2^{3+\log_2 k})[\mathbb{Z}_2^k]$ can be evaluated in time $O(2^k poly(n))$ and space $O(k poly(n))$, resulting in the following theorem.

Theorem 1 (Koutis [14] and Williams [15]) *There exists an algorithm that, given an instance $P(x_1, \ldots, x_n)$ of the k-MLD problem, correctly returns "no" if $P(X)$ does not contain a k multilinear term. Otherwise, it returns "yes" with probability at least 1/5. The algorithm has time complexity $O(2^k poly(n))$ and space complexity $O(k poly(n))$.*

III. Scan Statistics Using Multilinear Detection

A. Subgraphs as Monomials

We define the sets $K = \{1, 2 \ldots, k\}$ where k is a *size parameter*, and $R = \{0, 1, 2, \ldots, W(V)\}$, where $W(V)$ is the weight of the entire node set—this is an upper bound on the weight of any subgraph. After the graph transformation of [5] illustrated in Figure 3, the input graph will have a weight $w(v)$ for each node $v \in V$.

We now define a set of variables $\{x_v : v \in V\}$, and we construct a polynomial over these variables. Every term in the polynomial will represent a connected subgraph of size at most k and weight at most $W(V)$. For $i \in K$ and $j \in R$, let $M_v(i,j)$ be the polynomial corresponding to a subgraph (1) containing node v, (2) of size i, and (3) weight j. The polynomials $M_v(i,j)$ will be constructed and evaluated recursively. As a result, there can be monomials in the polynomial representation for $M_v(i,j)$ with squared terms, e.g., the monomial $x_1^2 x_2$ in Figure 4—these terms correspond to the same node being considered multiple times during the recursive construction.

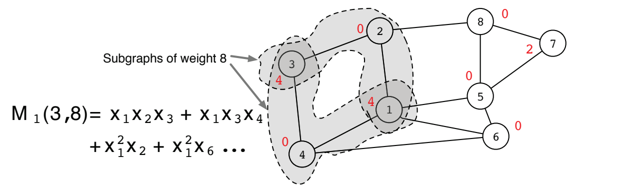

Subgraphs of weight 8
$M_1(3,8)= x_1x_2x_3 + x_1x_3x_4$
$+ x_1^2x_2 + x_1^2x_6 \ldots$
1
2
3
4
5
6
7
8
0
0
2
0
4
4
0
0

Fig. 4: Example showing the graph $G(V,E)$ constructed in Figure 3, with $|V| = 8$, and the node weights shown in red next to the node ID. $M_1(3,8)$ is the polynomial consisting of node 1 and two other nodes, with weight 8. Two of the terms in the polynomial, $x_1x_2x_3$ and $x_1x_3x_4$, are multilinear, but there are other terms, such as $x_1^2x_2$ and $x_1^2x_6$ that arise because they also satisfy the weight constraint. These will be canceled out when the polynomial is evaluated.

B. Basic-MultilinearScan: *Optimal, but slow*

We first give the intuition behind the algorithm Basic-MultilinearScan and the high level steps.

- *Preprocessing step*: The algorithm starts by performing the graph refinement of [5] (Section II). This corresponds to the **for** loop in lines 4–6 in Algorithm 2, which is run for each possible p-value $\alpha \in A$.
- *Evaluating the polynomials $M_v(i,j)$*: Algorithm 1, ConnectedSubgraphSearch, constructs and evaluates the polynomial $M_v(i,j)$ for each node v, size $i \in K$ and weight $j \in R$ in the graph G (line 6 of Algorithm 2). These polynomials are constructed recursively through a dynamic program. Only the results of their evaluation are stored and not the complete representation of each $M_v(i,j)$.
- *Finding the best solution*: The algorithm evaluates $F(j, i, \alpha)$ for each i, j such that $M(i,j) \neq \bar{0}$, and for each α, and returns the best solution in line 7. Let $OPT(F,k) = \max_{S: i \le k} F(W(S), i, \alpha)$ be the optimal solution over connected subgraphs of H with size $i \le k$; our algorithm returns $OPT(F,k)$ with constant probability.
- *Reducing space with matrix representations*: The results of the polynomial evaluations are elements of $Q[\mathbb{Z}_2^k]$, and we require $O(2^k)$ bits for storing them. Instead, we use the idea of matrix representations of the elements of $\mathbb{Z}_2^k$ from [14]. This allows us to evaluate the polynomial requiring only $O(1)$ space for each $M_v(i,j)$. We describe this method in the full version [8].

Algorithm 1 ConnectedSubgraphSearch($G(V,E), w, k$).

```
1: Input: Instance $(G(V,E), w)$ and parameter $k$
2: Output: Polynomial $M$, such that $M(i,j)$ is non-zero if $G$ has
   a subgraph $S$ with size $i \le k$ and weight $j \le W(V)$
3: For each node $v$, pick a random vector $x_v \in Q[\mathbb{Z}_2^k]$
4: $M_v(i,j) = \bar{0}$ for $i \in K$, $j \in R$
5: for $v \in V$ do
6:     $M_v(1, w(v)) = x_v$
7: for $v \in V$, $i = 2$ to $k$, $j = 0$ to $W(V)$ do
8:     $M_v(i,j) = \sum_{u \in Nbr(v)} \sum_{i'=1}^{i-1} \sum_{j'=0}^{j} (M_v(i',j') \cdot M_u(i-i', j-j'))$
9: $M(i,j) = \sum_v M_v(i,j)$ for $i \in K$, $j \in R$
10: return $M$
```

Theorem 2: Let $F(\cdot)$ be a non-parametric scan statistic, as defined in Section II. Algorithm Basic-MultilinearScan

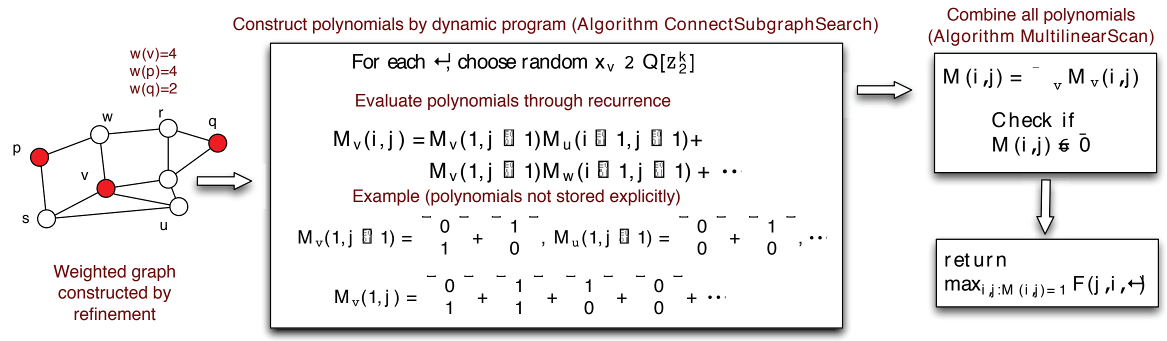

Construct polynomials by dynamic program (Algorithm ConnectSubgraphSearch)
Combine all polynomials (Algorithm MultilinearScan)
w(v)=4
w(p)=4
w(q)=2
p
w
r
q
v
s
u
Weighted graph constructed by refinement
For each α choose random $x_v \in Q[Z_2^k]$
Evaluate polynomials through recurrence
$M_v(i,j) = M_v(1,j-1)M_u(i-1,j-1)+$
$M_v(1,j-1)M_w(i-1,j-1)+\cdots$
Example (polynomials not stored explicitly)
$M_v(1,j-1) = \begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix}$, $M_u(1,j-1) = \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix}, \cdots$
$M_v(1,j) = \begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix} + \cdots$
$M(i,j) = \sum_v M_v(i,j)$
Check if $M(i,j) \neq \bar{0}$
return $\max_{i,j:M(i,j)=1} F(j,i,\alpha)$

Fig. 3: Overview of graph scan statistics using Algorithm BASIC-MULTILINEARSCAN. The input is a weighted graph G, which is constructed as in Figure 2. The subroutine CONNECTEDSUBGRAPHSEARCH constructs and evaluates the polynomials $M_v(i,j)$ for each node v, size $i \leq k$ and weight in the interval $[(1+\epsilon)^{j-1}, (1+\epsilon)^j]$. More terms of the polynomial $M_v(3,4)$ are shown in Figure 4, where node 1 corresponds to node v here. Q denotes the field $GF(2^{3+\log_2 k})$, and $\bar{0}$ denotes the additive identity of $Q[Z_2^k]$.

Algorithm 2 BASIC-MULTILINEARSCAN($(G^0(V^0,E^0), p, \alpha_{max}), k$).

```
1: Input: Instance $(G^0(V^0,E^0), p, \alpha_{max})$, parameter $k$
2: Output: Score $OPT(F,k)$
3: Let $A$ be the set of $p$-values of nodes in $V$ below $\alpha_{max}$
4: for $\alpha \in A$ do
5:     Run refinement step of [5] to create a weighted graph $G(V,E)$
       on "super nodes". Let $w(v)$ denote the weight of node $v$.
6:     $M^{\alpha}$ = CONNECTEDSUBGRAPHSEARCH$(G(V,E), w, k)$
7: return $\max_{\alpha \in A} \max_{i,j:M^{\alpha}(i,j) \neq \bar{0}} F(j,i,\alpha)$
```

returns $OPT(F,k)$ defined above with probability at least $1/5$, in time $O(2^k|A|mk^2W(V)^2)$, and using space $O(knW(V))$, where A and is defined in line 3 of Algorithm 2.

C. *Scaling Algorithm 2 with logarithmic binning*

The quadratic dependence on $W(V)$ creates a bottleneck in CONNECTEDSUBGRAPHSEARCH. To make the computation scalable, we group the weights of the input graph by considering powers of $(1+\epsilon)$, where $\epsilon > 0$ is an error parameter: if $w(v) \in [(1+\epsilon)^{j-1}, (1+\epsilon)^j)$, we say that v is in the weight group j; this definition is extended to the weight of a subgraph. For instance, if $\epsilon = 1$, nodes with weight in $[8,16)$ are in group 4. The goal is to scale the weights of the input graph down by a logarithmic factor and run Algorithm 1 on this much smaller set of weights.

With a slight abuse of notation, let us redefine R as $\{0,1,2,\ldots,r\}$, where r is a *weight parameter*. Now the polynomial $M_v(i,j)$ corresponds to a subgraph (1) containing node v, (2) of size i, and (3) total weight in $[(1+\epsilon)^{j-1}, (1+\epsilon)^j)$. $M_v(i,0)$ represents subgraphs of weight 0.

We need to modify CONNECTEDSUBGRAPHSEARCH to satisfy condition (3). In the base case, we set $M_v(1,j)$ to be x_v if node v is in group j and 0 otherwise. For $i \geq 2$, we consider the following cases:

Case 1. For $j = 0$, a polynomial $M_v(i,0)$ represents connected subgraphs of weight 0. For a graph to have weight 0, its two parts must both have weight 0 as well.

Case 2. For $j = 1$, a polynomial $M_v(i,1)$ represents connected subgraphs of weight 1. For a graph to have weight 1, one of its two parts must have weight 1, and the other must have weight 0.

Case 3. For $j \geq 2$, a polynomial $M_v(i,j)$ represents connected subgraphs of weight between $(1+\epsilon)^{j-1}$ and $(1+\epsilon)^j$. We could obtain a subgraph with this weight in one of two ways. The first is to combine two subgraphs, each of weights between $(1+\epsilon)^{j-2}$ and $(1+\epsilon)^{j-1}$. The second way is to combine a subgraph with weight between $(1+\epsilon)^{j-1}$ and $(1+\epsilon)^j$ with one of weight 0.

We propose APPROX-MULTILINEARSCAN, which takes an extra parameter, ϵ, and sets the weight parameter to $r = \lceil \log_{(1+\epsilon)}(kw_{max}+1) \rceil$, where $w_{max} = \max_v w(v)$ is the maximum weight of a node. That way, r is the maximum (scaled down) weight of a subgraph of size k. Let $W_{1+\epsilon}(S) = (1+\epsilon)^{\lceil \log_{1+\epsilon}(W(S)+1) \rceil}$ be the approximate weight of S, and let $OPT(F,k,\epsilon) = \max_{S:i \leq k} F(W_{1+\epsilon}(S), i, \theta)$ be the optimal solution over connected subgraphs of G^0 with size $i \leq k$ and rounded weights. Then, we obtain the following result.

Theorem 3: Algorithm APPROX-MULTILINEARSCAN returns $OPT(F,k,\epsilon)$ with probability at least $1/5$, in time $O(2^k|A|mk^2 \log_{1+\epsilon}(kw_{max}))$ and space $O(kn \log_{1+\epsilon}(kw_{max}))$, for A and w_{max} defined above.

D. APPROX-MULTILINEARSCAN *in Pregel*

We implement parallel versions of CONNECTEDSUBGRAPHSEARCH and BASIC-MULTILINEARSCAN in Giraph. We have also explored a GraphX [17] version of the algorithm, but it performs poorly, and we omit the results.

Our parallel algorithm, MULTILINEARSCANGIRAPH, calls subroutine PARCONNECTEDSUBGRAPHSEARCH; this is a parallel version of CONNECTEDSUBGRAPHSEARCH. As in the sequential version, this is described without the more efficient matrix representation, which requires a for loop with 2^k steps around the Pregel call. PARCONNECTEDSUBGRAPHSEARCH exploits two levels of parallelism in the sequential implementation: (1) the outer **for** loop, which corresponds to the matrix representation; (2) the **for** $v \in V$ loop in line 7 of CONNECTEDSUBGRAPHSEARCH happens in parallel.

IV. EXPERIMENTS

A. *Experimental Setup*

Datasets. A summary of the datasets is provided in Table I. See the full version [8] for more details.

TABLE I: Datasets used in our experiments

Dataset	Description	Nodes	Edges	Snap-shots
CitHepPh	Citation network	11,895	76,284	4
NEast	Network of counties in Northeastern USA	245	683	10,000
Traffic	Traffic Network of Los Angeles Country, CA	1,870	1,993	1,488
Twitter	Follower network collected through Twitter API	2,645	17,108	182
BWSN	Battle of the Water Sensors	12,527	14,831	22
Random	Erdos-Renyi graphs	100 to 10^6	⇠ 100 to ⇠ 10^6	5
soc-Live-Journal1	The largest connected subgraph of soc-Live-Journal data from SNAP	4,843,864	42,843,302	1
as-Skitter	The largest connected subgraph of as-Skitter data from SNAP	1,694,538	11,093,792	1
MN-blkgrp	Block group network of Minnesota	4,084	12,660	1

Baseline methods. We compare our proposed algorithms to FASTCOLCODENP [5], which, to the best of our knowledge, is the only method for scan statistics optimization in graphs with approximation guarantees. Additionally, we compare scalability with graph size to 5 state-of-the-art heuristic methods: (1) `NPHGS` [12], (2) `AdditiveGraphScan` (GS) [18], (3) `DepthFirstScan` (DFS) [1], (4) `GraphLaplacian` (GL [19], (5) `EdgeLasso` (EL) [20].

B. Scalability of our algorithms

In Figure 1 (top), we show the running time of MULTILINEARSCANGIRAPH and the baseline methods as a function of the size of the graph. MULTILINEARSCANGIRAPH is the only one besides FASTCOLCODENP that is able to process the instance of 10^6 nodes. All the other algorithms run out of memory or do not finish running within 24 hours.

1) Scalability with k: We compare the time and space scalability of APPROX-MULTILINEARSCAN to FASTCOLCODENP [5] in terms of the size parameter k. In Figure 5 a–b, we show the ratio of FASTCOLCODENP to APPROX-MULTILINEARSCAN with respect to time (a) and memory usage (b). Higher is better, and points above the dashed black line (ratio of 1) indicate that our proposed method improves over FASTCOLCODENP, which is the case for k [illegible] 6. Our algorithm is up to 1,000 times faster for a solution size of 12 and uses as little as a tenth of memory compared to FASTCOLCODENP. The total running time (c) and memory usage (d) for $k = 7$ are also shown.

C. Approximation Error and Solution Quality

We now study the effect of the ✏ parameter, which controls the approximation guarantee of APPROX-MULTILINEARSCAN. Figure 6 shows the objective score for different values of ✏, for $k = 5$. We compare this score to that obtained by FASTCOLCODENP, which is optimal with high probability. The solutions discovered by our algorithm are much better than the worst case approximation bound from Theorem 3, and they are usually close to the optimal solution. Even for ✏ = 2 (i.e, approximation guarantee of 3), APPROX-MULTILINEARSCAN yields a good estimate while keeping the running time low. We observe similar trends in the remaining datasets.

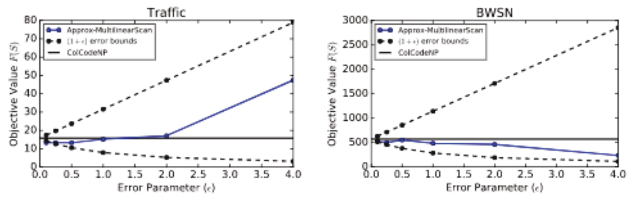


Fig. 6: Effect of the error parameter (✏) on the quality of the solution discovered. BASIC-MULTILINEARSCAN obtains close-to-optimal solutions (i.e., close to the horizontal line) despite the worst-case error bound (black dashed lines).

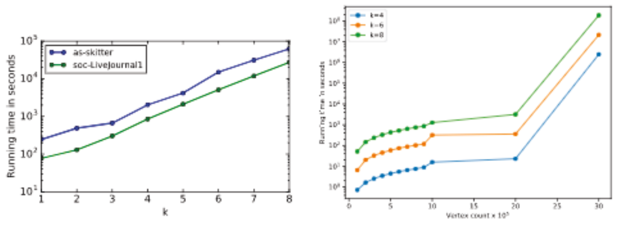


Fig. 7: Running time with (a) k for soc-LiverJournal1 and as-Skitter, and (b) number of nodes in random graphs

We also evaluate APPROX-MULTILINEARSCAN in terms of event detection power in the BWSN dataset. Table II reports the average performance of FASTCOLCODENP and our algorithm on all the snapshots of the dataset. We improve on the precision, recall, and F1 score of the color-coding based method, while having similar performance in terms of the objective score, *but being almost 3 times faster.*

TABLE II: Detection performance in the BWSN dataset.

	FASTCOLCODENP	**APPROX-MSCAN**
Precision	0.977	**0.985**
Recall	0.955	**0.957**
F1 Score	0.952	**0.961**
Accuracy	**0.973**	0.945
Obj. Score	**602.164**	601.938
Running Time	987	**366**

D. Parallel evaluation

Figure 7(a) shows parallel runtime for two of the largest graphs found in SNAP. Figure 7(b) shows the variation of the running time with the number of nodes for random networks, which is much more gradual. The communication cost in Giraph and GraphX is the main bottleneck in our algorithm, which has a very different computing pattern from standard benchmarks for distributed systems. For instance, PageRank algorithms stop sending messages to specific nodes as the algorithm proceeds, when certain criteria are met, leading to reduced communication over time. In contrast, in our algorithm, in each super step, vertices always send the same amount of data to neighbors, which contributes to significant overheads. In as-Skitter, for instance, we observed over 80% overhead with 256 and 512 parallel tasks over 16 machines.

E. Application: finding undervaccinated clusters

As an application, we study undervaccinated clusters in Minnesota[1], which have become a concern for public health

[1]//www.health.state.mn.us/divs/idepc/immunize/stats/school/

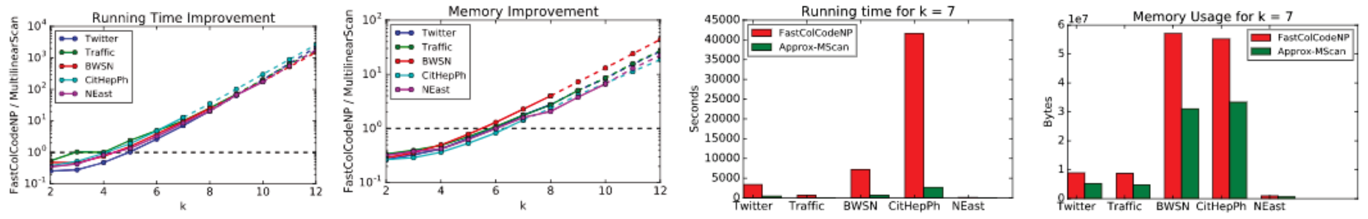


Fig. 5: Running time (a) and memory improvement (b) when using APPROX-MULTILINEARSCAN over FASTCOLCODENP. For $k \geq 6$, our algorithm is orders of magnitude faster than FASTCOLCODENP; thus, we can discover larger anomalous subgraphs in a fraction of the time. Memory usage also improves by up to an order of magnitude. We also show total running time (c) and memory usage (d) for $k = 7$.

officials in recent years. For each school, the dataset provides the number of students who received the MMR vaccine, which has a 97.5% average vaccination rate statewide in Minnesota. Our goal is to discover geographical clusters where the vaccination rate for MMR is much lower than the statewide average. Figure 1 shows the block group network and a connected subgraph discovered by our method. The unvaccination rate in the cluster is 6%, 2.5 times higher than expected. All prior work on finding undervaccinated clusters has only considered well-rounded shapes [21], which would not find such a cluster.

V. RELATED WORK

There is a large body of work on anomaly detection that relates to our paper (see [22] for a comprehensive discussion on graph anomaly detection). For brevity, we only discuss results for scan statistics optimization; an extended related work is presented in the full version of the paper [8]. *Algorithms for parametric scan statistics* include exhaustive search [23], `AdditiveGraphScan` [18], `EdgeLasso` [20], `GraphLaplacian` [19], and FASTCOLCODEP [5]; (2) *Algorithms for non-parametric scan statistics* include `NPHGS` [12] and FASTCOLCODENP [5]. To the best of our knowledge, the latter is the only method with approximation guarantees for graph scan statistics. FASTCOLCODENP is a fixed-parameter tractable algorithm with running time and space complexity $O((2e)^k m)$ and $O(2^k m)$, respectively. Our algorithm improves both the time and space bounds to $O(2^k m)$ and $O(km)$, respectively. As discussed above, these improved bounds lead to orders of magnitude of improvement on scalability.

Finally, we note that there has been limited work on parallel algorithms for network problems, especially in the emerging frameworks, such as Spark, Giraph, and Graphlab. Most of this work has been restricted to simple problems like PageRank and shortest paths [24]. The only prior work on parallel algorithms for scan statistics is by Zhao et al. [7]. However, they show results only for datasets with up to 12,000 nodes. Our algorithms improve on the running time of [7] for these instances and also scale to very large networks.

VI. CONCLUSIONS

We present a novel approach for graph scan statistics based on algebraic techniques for multilinear detection, which gives provable tradeoffs between running time and approximation guarantee. Our method leads to significant improvement both in time and space over the state-of-the-art methods. Further, our algorithm can be used without any modification for a broad class of parametric and non-parametric functions.

Acknowledgements. This work was partially supported by the following grants: DTRA CNIMS Contracts HDTRA1-11-D-0016-0010, HDTRA1-17-0118, and NSF grants IIS-1633028, ACI-1443054.

REFERENCES

[1] S. Speakman *et al.*, "Scalable detection of anomalous patterns with connectivity constraints," *Jl Comp Graphical Stat*, 2015.
[2] M. Leiserson *et al.*, "Pan-cancer network analysis identifies combinations of rare somatic mutations across pathways and protein complexes," *Nature genetics*, vol. 47, no. 2, pp. 106–114, 2015.
[3] T. Hansen and F. Vandin, "Finding mutated subnetworks associated with survival in cancer," *arXiv preprint arXiv:1604.02467*, 2016.
[4] D. Williamson and D. Shmoys, *The Design of Approximation Algorithms*. Cambridge University Press, 2011.
[5] J. Cadena, F. Chen, and A. Vullikanti, "Near-optimal and practical algorithms for graph scan statistics," in *SIAM Data Mining (SDM)*, 2017.
[6] R. Downey and M. Fellows, *Parameterized Complexity*. Springer, 2012.
[7] J. Zhao *et al.*, "Parallel algorithms for anomalous subgraph detection," *Concurrency and Computation: Practice and Experience*, 2016.
[8] J. Cadena *et al.*, "Fast graph scan statistics optimization using algebraic fingerprints," http://tinyurl.com/y76veog5, 2017.
[9] M. Kulldorff, "A spatial scan statistic," *Communications in Statistics: Theory and Methods*, 1997.
[10] D. B. Neill, "Fast subset scan for spatial pattern detection," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2012.
[11] R. H. Berk and D. H. Jones, "Goodness-of-fit test statistics that dominate the kolmogorov statistics," *Z. Wahrsch. Verw. Gebiete*, 1979.
[12] F. Chen and D. Neill, "Non-parametric scan statistics for event detection and forecasting in heterogeneous social media graphs," in *KDD*, 2014.
[13] E. McFowland, S. Speakman, and D. B. Neill, "Fast generalized subset scan for anomalous pattern detection," *JMLR*, vol. 14(1), 2013.
[14] I. Koutis, "Faster algebraic algorithms for path and packing problems," in *Proc. ICALP*, 2008.
[15] R. Williams, "Finding paths of length k in o(k2) time," *Information Processing Letters*, vol. 109, no. 6, pp. 315–318, 2009.
[16] G. Mullen and C. Mummert, "Finite fields and applications," *American Mathematical Society*, vol. 3, pp. 19–20, 2007.
[17] J. Gonzalez *et al.*, "Graphx: Graph processing in a distributed dataflow framework," in *Proc OSDI*, 2014.
[18] S. Speakman, Y. Zhang, and D. B. Neill, "Dynamic pattern detection with temporal consistency and connectivity constraints," in *ICDM*, 2013.
[19] J. Sharpnack, A. Singh, and A. Rinaldo, "Changepoint detection over graphs with the spectral scan statistic." in *AISTATS*, 2013.
[20] ——, "Sparsistency of the edge lasso over graphs." in *AISTATS*, 2012.
[21] T. A. Lieu *et al.*, "Geographic clusters in underimmunization and vaccine refusal," *Pediatrics*, vol. 135, no. 2, pp. 280–289, 2015.
[22] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data Mining and Knowledge Discovery*, 2014.
[23] K. Takahashi *et al.*, "A flexibly shaped space-time scan statistic for disease outbreak detection and monitoring." *Int. Jl. of Health Geographics*, 2008.
[24] J. Wei *et al.*, "Benchmarking of distributed computing engines: Spark and graphlab for big data analytics," in *Proc. IEEE BigData*, 2016.