Anomalous and Significant Subgraph Detection in Attributed Networks

Feng Chen \textsuperscript{1}, Petko Bogdanov \textsuperscript{1}, Daniel B. Neill \textsuperscript{2}, and Ambuj K. Singh \textsuperscript{3}

\textsuperscript{1} Department of Computer Science  
College of Engineering and Applied Sciences  
University at Albany - SUNY

\textsuperscript{2} Event and Pattern Detection Laboratory  
H.J. Heinz III College  
Carnegie Mellon University

\textsuperscript{3} Department of Computer Science &  
Biomolecular Science and Engineering  
University of California at Santa Barbara
Roadmap

• Introduction and motivation
• Part 1: Subgraph detection in static attributed networks
• Part 2: Subgraph detection in dynamic attributed networks
• Conclusion and future directions
Real-world networks

Internet map

Biological networks

Road networks

Blog networks

Food web

Terrorist networks
Real-world networks

Protein-protein interaction networks

Power grid networks

Dating networks

Facebook friends’ networks

Retail networks

Water distribution networks
Anomalous & significant subgraphs

Anomalous and significant subgraphs refer to subgraphs, in which the behaviors (attributes) of the nodes or edges are significantly different from the behaviors of those outside the subgraphs.

This tutorial mainly reviews methods on detection of anomalous and significant subgraphs with connectivity constraint.
Anomalous & significant subgraphs

• Detection of subnetwork biomarkers

(Chuang et al. 2007)
Anomalous & significant subgraphs

• Detection of road traffic congestion events

https://mikethemadbiologist.com/2015/08/08/the-ripple-effects-of-mass-transit/
Anomalous & significant subgraphs

• Detection of abnormally high breakage in a distribution network

(de Oliveira et al., 2010)
Anomalous & significant subgraphs

• Detection of disease outbreaks

Other applications

Societal events in social media
Malicious cargo

Image/video surveillance
New business discovery

Auction fraud, fake reviews, email spams, false advertising

Extreme weather events
Crime hotspots

Brain activities
Disease diagnosis
Animal activities

New chemical structures
New knowledge discovery
Subgraph detection: definition

• **Univariate static networks**

Network topology $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

Attributes (w)

$s \in \mathcal{S}$ satisfies a predefined topological constraint (e.g. connectivity).

$\max_{S \subseteq \mathcal{V}} F(S)$

$F(S)$ characterizes the level of anomalousness of $S$ based on attributes.

Constraint is defined based on network topology.
Subgraph detection: definition

- **Univariate** static networks

Network topology

Attributes (w)

\[ S = \{1, 2, 4, 5\}, \quad F(S) = 3 + 4 + 2 + 3 = 12 \]

\[
\max_{S \subseteq \mathbb{V}} F(S) = \sum_{i \in S} w(i)
\]

s. t. \( S \) is connected
Subgraph detection: definition

- **Multivariate static networks**

Constraint is defined based on network topology.

\[
S = \{1, 2, 4, 5\}
\]

\[
R = \{1, 2, 4\}
\]

\[
\max_{S,R} F(S, R)
\]

s.t. \(S\) satisfies a predefined topological constraint
Subgraph detection: definition

- **Multivariate static networks**

Constraint is defined based on network topology.

\[ S = \{1, 2, 4, 5\} \]

\[ R = \{1, 2, 4\} \]

\[
\max_{S, R} F(S, R) \]

s. t. \( S \) satisfies a predefined topological constraint
Subgraph detection: definition

- **Multivariate static networks**

  Network topology

  Constraint is defined based on network topology.

  Attributes (w)

  \[ S = \{1, 2, 4, 5\} \]

  \[ R = \{1, 2, 4\} \]

  \[ \max_{S,R} F(S, R) \]

  s. t. S satisfies a predefined topological constraint
Subgraph detection: definition

• Multivariate **dynamic** networks

**Network topology**

- Constraint is defined based on network topology.

Mathematical expression:

\[
\max_{S, R, W} F(S, R, W)
\]

**s. t.** \( S \) satisfies a predefined topological constraint
Subgraph detection: definition

Constraint is defined based on network topology.

Network topology

\[ S = \{1, 2, 4, 5\} \]

\[
\max_{S,R,W} F(S, R, W)
\]

s. t. \( S \) satisfies a predefined topological constraint
Subgraph detection: definition

Network topology

Constraint is defined based on network topology.

\[ S = \{1,2,4,5\} \]

\[ R = \{1,2,4\} \]

\[ \max F(S, R, W) \]

\[ s.t. \ S \text{ satisfies a predefined topological constraint} \]
Subgraph detection: definition

Network topology

Constraint is defined based on network topology.

\[ \text{satisfies a predefined topological constraint} \]

\[ S = \{1, 2, 4, 5\} \]

\[ R = \{1, 2, 4\} \]

\[ \max F(S, R, W) \]

\[ s.t. \] S satisfies a predefined topological constraint
Subgraph detection: definition

Network topology

Constraint is defined based on network topology.

$R = \{1, 2, 4\}$

$W = \{1, 2, 3\}$

$S = \{1, 2, 4, 5\}$

$max \ F(S, R, W)$

$s.t. \ S$ satisfies a predefined topological constraint
Subgraph detection: definition

Network topology

Constraint is defined based on network topology.

\[ R = \{1,2,4\} \]
\[ W = \{1,2,3\} \]
\[ S = \{1,2,4,5\} \]

\[ \max F(S, R, W) \]
\[ S, R, W \]
\[ s.t. \ S \text{ satisfies a predefined topological constraint} \]
Score function & constraints

• Score functions
  • Parametric scan statistics
    • Kulldorff’s statistic, Expectation-based statistic
  • Nonparametric scan statistics
    • Higher Criticism (HC) statistic, Berk-Jones’s statistic
  • Network design based functions
    • Prize Collecting Steiner Tree (PCST) objective

• Topological constraints
  • Regular shapes, such as circles and rectangles.
  • Connectivity *(the focus of this tutorial)*
  • Compactness
Computational Challenges

• Exponentially many possible subsets, $O(2^N \cdot 2^M)$, where $N$ and $M$ refer to the total numbers of nodes and attributes, respectively: computationally infeasible for naïve search.

• Given a score function and a topological constraint (e.g. connectivity) predefined by a user, how we can identify the highest scoring subgraphs efficiently and effectively?
Comparisons with related topics

• The unique aspect of this tutorial is that the focus is on detection of subgraph patterns that optimize certain structural and attribute properties (or constraints) in large attributed networks.

• In comparison, most relevant tutorials were focused on analysis of graph-level or node-level patterns in networks without attributes.

• Community detection and node embedding methods will not be reviewed in this tutorial.
Part 1: Subgraph Detection in Static Attributed Networks
Taxonomy

Anomalous & significant subgraph detection

Static attributed networks

Dynamic attributed networks

Spatial networks

Complex networks

Fast subset scan

Graph scan

Nonparametric graph scan

Submodular optimization methods

Graph-structured Sparse optimization methods
Detection in Spatial Networks

- Each graph node corresponds to the centroid of a small area (e.g., zip code or census tract), with corresponding lat/long coordinates.
- Edges are defined by spatial adjacency between areas.
- Some quantities (e.g., number of crimes or disease cases) are monitored for each area → attributes of that node.
- **Goal**: find connected subgraph with collectively anomalous attribute values.
- Graph sizes tend to be relatively small (hundreds-thousands) but still far too large for exhaustive search over subgraphs.
Multivariate event detection

Spatial time series data from spatial locations $s_i$ (e.g. zip codes)

Outbreak detection
- $d_1 =$ respiratory ED
- $d_2 =$ constitutional ED
- $d_3 =$ OTC cough/cold
- $d_4 =$ OTC anti-fever
  (etc.)

Time series of counts $c_{i,m,t}$ for each zip code $s_i$ for each data stream $d_m$.

Main goals:

- **Detect** any emerging events.
- **Pinpoint** the affected subset of locations and time duration.
- **Characterize** the event by identifying the affected streams.

Compare hypotheses:

$H_1(D, S, W)$

$D =$ subset of streams
$S =$ subset of locations
$W =$ time duration

vs. $H_0$: no events occurring
Expectation-based scan statistics

(Kulldorff, 1997; Neill and Moore, 2005)

We search for spatial regions (subsets of locations) where the recently observed counts for some subset of streams are significantly higher than expected.

We perform time series analysis to compute expected counts (“baselines”) for each location and stream for each recent day.

We then compare the actual and expected counts for each subset (D, S, W) under consideration.
We find the subsets with highest values of a likelihood ratio statistic, and compute the \( p \)-value of each subset by randomization testing.

\[
F(D, S, W) = \frac{\Pr(\text{Data} \mid H_1(D, S, W))}{\Pr(\text{Data} \mid H_0)}
\]

To compute \( p \)-value
Compare subset score to maximum subset scores of simulated datasets under \( H_0 \).

Maximum subset score = 9.8

Not significant \((p = .098)\)

2nd highest score = 8.4

Significant! \((p = .013)\)

\( F_1^* = 2.4 \)
\( F_2^* = 9.1 \)
\( F_{999}^* = 7.0 \)

(Kulldorff, 1997; Neill and Moore, 2005)
Which regions to search?

Typical approach: “spatial scan” (Kulldorff, 1997)

Each search region $S$ is a sub-region of space.
- Choose some region shape (e.g. circles, rectangles) and consider all regions of that shape and varying size.
- Low power for true events that do not correspond well to the chosen set of search regions (e.g. irregular shapes).

Our approach: “subset scan” (Neill, 2012)

Each search region $S$ is a subset of locations.
- Find the highest scoring subset, subject to some constraints (e.g. spatial proximity, connectivity).
- For multivariate, also optimize over subsets of streams.
- Exponentially many possible subsets, $O(2^N \times 2^M)$: computationally infeasible for naïve search.
Fast subset scan

• In certain cases, we can optimize $F(S)$ over the exponentially many subsets of the data, while evaluating only $O(N)$ rather than $O(2^N)$ subsets.

• Many commonly used scan statistics have the property of linear-time subset scanning:
  • Just sort the data records (spatial locations, etc.) from highest to lowest priority according to some function…
  • … then search over groups consisting of the top-$k$ highest priority records, for $k = 1..N$.

The highest scoring subset is **guaranteed** to be one of these!

Sample result: we can find the *most anomalous* subset of Allegheny County zip codes in 0.03 sec vs. $10^{24}$ years.
Fast subset scan with spatial proximity constraints

• Maximize a likelihood ratio statistic over all subsets of the “local neighborhoods” consisting of a center location $s_i$ and its $k-1$ nearest neighbors, for a fixed neighborhood size $k$.

• Naïve search requires $O(N \cdot 2^k)$ time and is computationally infeasible for $k > 25$.

• For each center, we search over all subsets of its local neighborhood in $O(k)$ time using LTSS, thus requiring a total time of $O(Nk) + O(N \log N)$ for sorting the locations.

• In Neill (2012), we show that this approach dramatically improves the timeliness and accuracy of outbreak detection for irregularly-shaped disease clusters.
Incorporating connectivity constraints

Proximity-constrained subset scans may return a disconnected subset of the data.

In some cases this may be undesirable, or we might have non-spatial data so proximity constraints cannot be used.

Example: tracking disease spread from person-to-person contact.

Example: identifying a connected subset of zip codes (Allegheny County, PA)
Taxonomy

Anomalous & significant subgraph detection

Static attributed networks
- Spatial networks
  - Fast subset scan
  - Graph scan
- Complex networks
  - Nonparametric graph scan

Dynamic attributed networks
- Submodular optimization methods
- Graph-structured Sparse optimization methods
Incorporating connectivity constraints

Proximity-constrained subset scans may return a disconnected subset of the data. In some cases this may be undesirable, or we might have non-spatial data so proximity constraints cannot be used.

Our **GraphScan** algorithm* can efficiently and exactly identify the highest-scoring connected subgraph:

- Can incorporate multiple data streams
- With or without proximity constraints
- Graphs with several hundred nodes

We can use the LTSS property to rule out subgraphs that are provably suboptimal, dramatically reducing our search space.

Incorporating connectivity constraints

We represent groups of subsets as strings of 0’s, 1’s, and ?’s.

Assume that the graph nodes are sorted from highest priority to lowest priority.

<table>
<thead>
<tr>
<th>Priority Ranking</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit String</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

The above bit string represents four possible subsets: \( \{1,4\} \), \( \{1,4,5\} \), \( \{1,4,6\} \), and \( \{1,4,5,6\} \).

LTSS property **without** connectivity constraints:
“If node \( x \in S \) and node \( y \notin S \), for \( x > y \), then subset \( S \) cannot be optimal.”

We can use the LTSS property to rule out subgraphs that are provably suboptimal, dramatically reducing our search space.
Incorporating connectivity constraints

We represent groups of subsets as strings of 0’s, 1’s, and ?’s.

Assume that the graph nodes are sorted from highest priority to lowest priority.

<table>
<thead>
<tr>
<th>Priority Ranking</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit String</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

The above bit string represents four possible subsets: \{1,4\}, \{1,4,5\}, \{1,4,6\}, and \{1,4,5,6\}.

LTSS property with connectivity constraints:
“If node x ∈ S and node y ∉ S, for x > y, and S \ {x} and S ∪ \{y\} are both connected, then subset S cannot be optimal.”

We can use the LTSS property to rule out subgraphs that are provably suboptimal, dramatically reducing our search space.
Incorporating connectivity constraints

We represent groups of subsets as strings of 0’s, 1’s, and ?’s.

Assume that the graph nodes are sorted from highest priority to lowest priority.

<table>
<thead>
<tr>
<th>Priority Ranking</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit String</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

The above bit string represents four possible subsets: {1,4}, {1,4,5}, {1,4,6}, and {1,4,5,6}.

LTSS property with connectivity constraints:
“If node \( x \in S \) and node \( y \notin S \), for \( x > y \), and \( S \setminus \{x\} \) and \( S \cup \{y\} \) are both connected, then subset \( S \) cannot be optimal.”

We can use the LTSS property to rule out subgraphs that are provably suboptimal, dramatically reducing our search space.
Incorporating connectivity constraints

We represent groups of subsets as strings of 0’s, 1’s, and ?’s.

Assume that the graph nodes are sorted from highest priority to lowest priority.

<table>
<thead>
<tr>
<th>Priority Ranking</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit String</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

The above bit string represents four possible subsets: {1,4}, {1,4,5}, {1,4,6}, and {1,4,5,6}.

LTSS property with connectivity constraints:
“If node x ∈ S and node y ∉ S, for x > y, and S \ {x} and S U {y} are both connected, then subset S cannot be optimal.”

Additional speedups can be gained by branch-and-bounding: we use the unconstrained subset score as an upper bound on the connected subgraph score, and rule out subsets which cannot be higher-scoring than the best subset found so far.
Evaluation: run times

Average Runtime for a Single Day of ED data

- GraphScan
- GraphScan w/ no Branch & Bounding
- Flexscan

Seconds vs. Neighborhood Size (k)
Evaluation: detection power

Comparison of detection power for outbreaks along highways

Percent of outbreaks detected vs. k

GraphScan
- circles
- All subsets
Extensions of GraphScan

What if we want to allow for events which spread dynamically over the (static) graph structure?

Based on a new variant of the LTSS property\(^1\), we can search for dynamic patterns while enforcing soft constraints on temporal consistency.

We have applied this method for accurate detection, tracking, and source-tracing of contaminants spreading through a water distribution network.\(^2\)

What if the underlying graph structure is unknown?

We can accurately **learn** the graph structure from unlabeled outbreak data, and use the learned structure for detection.

Often, the learned graph enables even faster detection of events than the true graph!\(^3\)


\(^3\)Somanchi and Neill, submitted.
Variants of GraphScan

Previous exact approaches are very slow…

FlexScan (Tango & Takahashi, 2005): exhaustive search over connected subgraphs within each spatial neighborhood, infeasible for $k > 25$.

Contiguous Max-LLR model (Murray et al., 2014): requires solving many mixed integer linear programs, exponentially many in worst case.

… but a variety of heuristic approaches exist.

Duczmal et al.: simulated annealing, genetic algorithms

Assuncao et al.: spanning trees

Chen and Neill: greedy growth

Speakman et al.: additive GraphScan

1) Construct conditionally additive score function.

2) Optimizing $F(S)$ reduces to maximum weight connected subgraph problem.
Anomalous & significant subgraph detection

Static attributed networks

Spatial networks

Fast subset scan

Graph scan

Complex networks

Nonparametric graph scan

Dynamic attributed networks

Submodular optimization methods

Graph-structured Sparse optimization methods
Social media is a real-time “sensor” of large-scale population behavior, and can be used for early detection of emerging events…

… but it is very complex, noisy, and subject to biases.

We have developed a new event detection methodology:
“Non-Parametric Heterogeneous Graph Scan” (NPHGS)

Applied to: civil unrest prediction, rare disease outbreak detection, and early detection of human rights events.
Technical Challenges

Integration of multiple heterogeneous information sources!
Technical Challenges

One week before Mexico’s 2012 presidential election:

- Hashtag “#Megamarch” mentioned 1,000 times
- Influential user “Zeka” posted 10 tweets
- Mexico City has 5,000 active users and 100,000 tweets
- Tweets that have been re-tweeted 1,000 times
- A specific link (URL) was mentioned 866 times
- Keyword “Protest” mentioned 5,000 times
Technical Challenges

One week before Mexico’s 2012 presidential election:

- Hashtag “#Megamarch” mentioned 1,000 times
- Influential user “Zeka” posted 10 tweets
- Mexico City has 5,000 active users and 100,000 tweets
- Keyword “Protest” mentioned 5,000 times
- Tweets that have been re-tweeted 1,000 times
- A specific link (URL) was mentioned 866 times
Twitter Heterogeneous Network

- Location
  - Geographic Neighbors
  - Geographical Relationship
- Tweet
  - Geographical Relationship
  - User Mention
  - Owner
- Hashtag
- Link
  - Part of
  - Part of
  - Part of

Co-occurrence

Followers
Veracruz, Jalapa, Mérida, Tepoztlan add to the #MegaMarcha vs imposición. Tambien Los Ángeles. Who else says

See you on Saturday at 15:00 in the #MegaMarcha

Ready to march, tweeting or filming tomorrow #MegaMarcha vs imposición. Hopefully many say #Vamon

"#MexicoExigeDemocracia" http://t.co/MdG5T3z0 Twitterers help me with a RT?. See you on Saturday at 15:00 in the #MegaMarcha.

"#MexicoExigeDemocracia" http://t.co/MdG5T3z0 Twitterers help me with a RT?. See you on Saturday at 15:00
Twitter Heterogeneous Network
1) We model the heterogeneous social network as a sensor network. Each node senses its local neighborhood, computes multiple features, and reports the overall degree of anomalousness.

2) We compute an empirical p-value for each node:
   - Uniform on [0,1] under the null hypothesis of no events.
   - We search for subgraphs of the network with a higher than expected number of low (significant) empirical p-values.

3) We can scale up to very large heterogeneous networks:
   - Heuristic approach: iterative subgraph expansion ("greedy growth" to subset of neighbors on each iteration).
   - We can efficiently find the best subset of neighbors, ensuring that the subset remains connected, at each step.
Sensor network modeling

Each node reports an empirical p-value measuring the current level of anomalousness for each time interval (hour or day).

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td># tweets, # retweets, # followers, #followees, mentioned_by, replied_by, diffusion graph depth, diffusion graph size</td>
</tr>
<tr>
<td>Tweet</td>
<td>Klout, sentiment, replied_by_graph_size, reply_graph_size, retweet_graph_size, retweet_graph_depth</td>
</tr>
<tr>
<td>City, State, Country</td>
<td># tweets, # active users</td>
</tr>
<tr>
<td>Term</td>
<td># tweets</td>
</tr>
<tr>
<td>Link</td>
<td># tweets</td>
</tr>
<tr>
<td>Hashtag</td>
<td># tweets</td>
</tr>
</tbody>
</table>

Features \(\xrightarrow{\text{empirical calibration}}\) Individual p-value for each feature \(\xrightarrow{\text{min}}\) Minimum empirical p-value for each node \(\xrightarrow{\text{empirical calibration}}\) Overall p-value for each node
Nonparametric scan statistics

Subgraph

\[ F(S) = \max_{\alpha \leq \alpha_{\text{max}}} \quad F_\alpha(S) = \max_{\alpha \leq \alpha_{\text{max}}} \phi(\alpha, N_\alpha(S), N(S)) \]

Significance level

Berk-Jones (BJ) statistic:

\[ \phi_{BJ}(\alpha, N_\alpha(S), N(S)) = N(S)K \left( \frac{N_\alpha}{N}, \alpha \right) \]

Kullback-Liebler divergence:

\[ K(x, y) = x \log \left( \frac{x}{y} \right) + (1 - x) \log \left( \frac{1 - x}{1 - y} \right) \]
We propose an approximate algorithm with time cost $O(|V| \log |V|)$. 

$$S^* = \arg\max_{S \in V: S \text{ is connected}} F(S)$$
NPHGS evaluation - civil unrest

<table>
<thead>
<tr>
<th>Country</th>
<th># of tweets</th>
<th>News source*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argentina</td>
<td>29,000,000</td>
<td>Clarín; La Nación; Infobae</td>
</tr>
<tr>
<td>Chile</td>
<td>14,000,000</td>
<td>La Tercera; Las Últimas Noticias; El Mercurio</td>
</tr>
<tr>
<td>Colombia</td>
<td>22,000,000</td>
<td>El Espectador; El Tiempo; El Colombiano</td>
</tr>
<tr>
<td>Ecuador</td>
<td>6,900,000</td>
<td>El Universo; El Comercio; Hoy</td>
</tr>
</tbody>
</table>

**Gold standard dataset:** 918 civil unrest events between July and December 2012.

**Example of a gold standard event label:**

PROVINCE = “El Loa”
COUNTRY = “Chile”
DATE = “2012-05-18”
LINK = “http://www.pressenza.com/2012/05/…”
DESCRIPTION = “A large-scale march was staged by inhabitants of the northern city of Calama, considered the mining capital of Chile, who demanded the allocation of more resources to copper mining cities”

We compared the detection performance of our NPHGS approach to homogeneous graph scan methods and to a variety of state-of-the-art methods previously proposed for Twitter event detection.
NPHGS outperforms existing representative techniques for both event detection and forecasting, increasing detection power, forecasting accuracy, and forecasting lead time while reducing time to detection.

Similar improvements in performance were observed on a second task:

Early detection of rare disease outbreaks, using gold standard data about 17 hantavirus outbreaks from the Chilean Ministry of Health.

<table>
<thead>
<tr>
<th>Method</th>
<th>FPR (FP/Day)</th>
<th>TPR (Forecasting)</th>
<th>TPR (Forecasting &amp; Detection)</th>
<th>Lead Time (Days)</th>
<th>Lag Time (Days)</th>
<th>Run Time (Hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST Burst Detection</td>
<td>0.65</td>
<td>0.07</td>
<td>0.42</td>
<td>1.10</td>
<td>4.57</td>
<td>30.1</td>
</tr>
<tr>
<td>Graph Partition</td>
<td>0.29</td>
<td>0.03</td>
<td>0.15</td>
<td>0.59</td>
<td>6.13</td>
<td>18.9</td>
</tr>
<tr>
<td>Earthquake</td>
<td>0.04</td>
<td>0.06</td>
<td>0.17</td>
<td>0.49</td>
<td>5.95</td>
<td>18.9</td>
</tr>
<tr>
<td>RW Event</td>
<td>0.10</td>
<td>0.22</td>
<td>0.25</td>
<td>0.93</td>
<td>5.83</td>
<td>16.3</td>
</tr>
<tr>
<td>Geo Topic Modeling</td>
<td>0.09</td>
<td>0.06</td>
<td>0.08</td>
<td>0.01</td>
<td>6.94</td>
<td>9.7</td>
</tr>
<tr>
<td>NPHGS (FPR=.05)</td>
<td>0.05</td>
<td>0.15</td>
<td>0.23</td>
<td>0.65</td>
<td>5.65</td>
<td>38.4</td>
</tr>
<tr>
<td>NPHGS (FPR=.10)</td>
<td>0.10</td>
<td>0.31</td>
<td>0.38</td>
<td>1.94</td>
<td>4.49</td>
<td>38.4</td>
</tr>
<tr>
<td>NPHGS (FPR=.15)</td>
<td>0.15</td>
<td>0.37</td>
<td>0.42</td>
<td>2.28</td>
<td>4.17</td>
<td>38.4</td>
</tr>
<tr>
<td>NPHGS (FPR=.20)</td>
<td>0.20</td>
<td>0.39</td>
<td>0.46</td>
<td>2.36</td>
<td>3.98</td>
<td>38.4</td>
</tr>
</tbody>
</table>

Table 3: Comparison between NPHGS and Existing Methods on the civil unrest datasets
Taxonomy

Anomalous & significant subgraph detection

Static attributed networks

Dynamic attributed networks

Spatial networks

Complex networks

Fast subset scan

Graph scan

Nonparametric graph scan

Submodular optimization methods

Graph-structured Sparse optimization methods
Subgraph detection via submodular optimization (Rozenshtein et al., KDD 2014)

• A class of subgraph detection problems can be framed as a general submodular (but not monotone) maximization problem:

$$\max_S F(S) + \lambda D(S)$$

A submodular score function that characterizes the level of anomalousness of the subset of nodes $S$.

A submodular compactness function that gives a higher score if the subset of nodes $S$ is more compact.
\( \frac{1}{2} \)-approximation for submodular maximization (Buchbinder et al., 2012)

Network topology \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \)

Node-level attributes

Initialize \( A = \emptyset, B = \text{everything} \)

In each step, grow \( A \) or shrink \( B \)

Invariant: \( A \subseteq B \)
\(1/2\)-approximation for submodular maximization (Buchbinder et al., 2012)

Network topology \(G = (V, E)\)

Node-level attributes

> Initialize \(A = \emptyset, B = \text{everything}\)

In each step, grow \(A\) or shrink \(B\)

Invariant: \(A \subseteq B\)
½-approximation for submodular maximization (Buchbinder et al., 2012)

**Network topology** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

- Node-level attributes

  - Random shuffling

  - Initial values:
    - $1, 2, 3, 4, 5, 6, 7$
    - Red: $1, 2, 5, 6$
    - Blue: $3, 4$
    - Green: $7$

  - After random shuffling:
    - $1, 7, 2, 5, 3, 6, 4$

- Initialize $A = \emptyset$, $B = $ everything

- In each step, **grow** $A$ or **shrink** $B$

- Invariant: $A \subseteq B$
$\frac{1}{2}$-approximation for submodular maximization  

(Buchbinder et al., 2012)

Network topology $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

Node-level attributes

Initialize $A = \emptyset$, $B = $ everything

In each step, grow $A$ or shrink $B$

Invariant: $A \subseteq B$
\( \frac{1}{2} \)-approximation for submodular maximization (Buchbinder et al., 2012)

**Network topology** \( G = (V, E) \)

**Node-level attributes**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Random shuffling

<table>
<thead>
<tr>
<th>1</th>
<th>7</th>
<th>2</th>
<th>5</th>
<th>3</th>
<th>6</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Initialize \( A = \emptyset, B = \text{everything} \)

In each step, grow \( A \) or shrink \( B \)

Invariant: \( A \subseteq B \)
$\frac{1}{2}$-approximation for submodular maximization (Buchbinder et al., 2012)

Network topology $\mathcal{G} = (\mathbb{V}, \mathbb{E})$

Node-level attributes

Initialize $A = \emptyset$, $B = \text{everything}$

In each step, grow $A$ or shrink $B$

Invariant: $A \subseteq B$
\(\frac{1}{2}\)-approximation for submodular maximization (Buchbinder et al., 2012)

**Network topology** \( \mathcal{G} = (\mathbb{V}, \mathbb{E}) \)

**Node-level attributes**

Initialize \( A = \emptyset, B = \text{everything} \)

In each step, **grow** \( A \) or **shrink** \( B \)

**Invariant:** \( A \subseteq B \)
\( \frac{1}{2} \)-approximation for submodular maximization (Buchbinder et al., 2012)

Network topology \( G = (V, E) \)

Node-level attributes

Random shuffling

Initialize \( A = \emptyset, B = \text{everything} \)

In each step, grow \( A \) or shrink \( B \)

Invariant: \( A \subseteq B \)

When \( A = B \), we return \( A \) (or \( B \)) as the final subset of nodes.
Case studies: event detection

- Bicing sensor networks

(Rozenshtein et al., KDD 2014)
Case studies: event detection

15.11.2012
ordinary day, no events

11.09.2012
Catalunya national day

(Rozenshtein et al., KDD 2014)
Case studies: event detection

• Events discovered with bicing data

Figure 4: Public holiday city-events discovered using the SDP algorithm.

(a) Barcelona: 11.09.12 National Day of Catalonia  
(b) Minneapolis: 4.07.12 Independence Day  
(c) Washington, DC: 27.05.13 Memorial Day  
(d) Los Angeles: 31.05.10 Memorial Day  
(e) New York: 6.09.10 Labor Day

(a) 01.06.12 Primavera sound music festival  
(b) 18.09.12 festival of the Poblenou neighborhood  
(c) 31.10.12 Halloween

(Rozenshtein et al., KDD 2014)
Taxonomy

Anomalous & significant subgraph detection

Static attributed networks

Dynamic attributed networks

Spatial networks

Complex networks

Fast subset scan

Graph scan

Nonparametric graph scan

Submodular optimization methods

Graph-structured Sparse optimization methods
Graph structured sparse optimization

The problem of subgraph detection

$$\max_{S \subseteq V} F(S) \quad \text{s.t. } S \text{ satisfies a predefined topological constraint.}$$

can be reformulated as

$$\max_{y \subseteq \{0,1\}^n} f(y) \quad \text{s.t. } \text{supp}(y) \text{ satisfies a predefined topological constraint.}$$

where $\text{supp}(y) = \{i \mid y_i > 0\}$ and $S$ can be identified as

$$S = \text{supp}(y), \quad \text{and } f(y) = F(S)$$
Graph structured sparse optimization

• This approach solves the relaxed problem

\[ \max_{y \in [0,1]^n} f(y) \quad s.t. \quad \text{supp}(y) \text{ satisfies a predefined topological constraint.} \]

• Three novel sparse optimization algorithms
  • Graph-structured iterative hard thresholding (Graph-IHT). (Zhou and Chen, ICDM, 2016)
  • Graph-structured gradient hard thresholding Pursuit (Graph-GHTP). (Zhou and Chen, ICDM, 2016)
  • Graph-structured matching pursuit (Graph-MP) (Chen and Zhou, IJCAI, 2016)
Interpretation of projection oracle

- A projection oracle $P(b)$ is defined as

$$P(b) = \arg \min_{y \in \mathbb{R}^n} \|y - b\|_2^2 \quad s.t. \quad \text{supp}(y) \text{ is a connected subset of size at most 4.}$$
Interpretation of projection oracle

- A projection oracle $P(b)$ is defined as

$$P(b) = \arg \min_{y \in \mathbb{R}^n} ||y - b||_2^2 \text{ s.t. supp}(y) \text{ is a connected subset of size at most 4.}$$

Network topology $G = (V, E)$

Projection of relaxed vector $\hat{y}$
Interpretation of projection oracle

• A projection oracle $P(b)$ is defined as

$$P(b) = \arg\min_{y \in \mathbb{R}^n} \|y - b\|^2_2 \text{ s.t. supp}(y) \text{ is a connected subset of size at most } 4.$$ 

Network topology $G = (V, E)$
Description of the Graph-IHT algorithm

(Chen and Zhou, ICDM, 2016)

**Algorithm: Graph-IHT**

1. **Input:** Instance $G$;
2. **Output:** The subset $S$;
3. $i \leftarrow 0$, $y^i \leftarrow$ an initial vector;
4. repeat
5. \[ b \leftarrow y^i + \eta \cdot P(\nabla f(y^i)) \]
6. \[ y^{i+1} \leftarrow P(b) \]
7. \[ i = i + 1; \]
8. until halting condition holds;
9. $S = \text{supp}(y^{i+1});$
10. return $S$

Projection on the gradient $\nabla f(y^i)$

Projection on an intermediate solution $b$
Illustration of the Graph-IHT algorithm

(Zhou and Chen, ICDM, 2016)

\[ \mathcal{M}(G, k = 5) = \{ y \mid y \in [0,1]^n, \text{supp}(y) \in \mathcal{M}(G, k = 5) \} \]

\[ \mathcal{M}(G, k = 5) \] represents the space of connected subsets of size at most 5.

Network instance \( G \)
Illustration of the Graph-IHT algorithm
(Zhou and Chen, ICDM, 2016)

Network instance $G$

$\mathcal{M}(G, k = 5) = \{ y | y \in [0,1]^n, \text{supp}(y) \in \mathcal{M}(G, k = 5) \}$
Illustration of the Graph-IHT algorithm

(Zhou and Chen, ICDM, 2016)

\[ M(\mathcal{G}, k = 5) = \{ \mathbf{y} | \mathbf{y} \in [0,1]^n, \text{supp}(\mathbf{y}) \in M(\mathcal{G}, k = 5) \} \]
Illustration of the Graph-IHT algorithm

(Zhou and Chen, ICDM, 2016)

Network instance $\mathbb{G}_t$

$f(y)$

$\mathcal{M}(\mathbb{G}, k = 5) = \{y \mid y \in [0,1]^n, \text{supp}(y) \in \mathbb{M}(\mathbb{G}, k = 5)\}$
Illustration of the Graph-IHT algorithm

(Zhou and Chen, ICDM, 2016)

\[ \mathcal{M}(G, k = 5) = \{y \mid y \in [0,1]^n, \text{supp}(y) \in \mathcal{M}(G, k = 5)\} \]
Illustration of the Graph-IHT algorithm

(Zhou and Chen, ICDM, 2016)

\[ \mathcal{M}(\mathbb{G}, k = 5) = \{ \mathbf{y} \mid \mathbf{y} \in [0,1]^n, \text{supp}(\mathbf{y}) \in \mathcal{M}(\mathbb{G}, k = 5) \} \]
Illustration of the Graph-IHT algorithm

(Zhou and Chen, ICDM, 2016)

$\mathcal{M}(\mathcal{G}, k = 5) = \{\mathbf{y} \mid \mathbf{y} \in [0,1]^n, \text{supp}(\mathbf{y}) \in \mathcal{M}(\mathcal{G}, k = 5)\}$
Illustration of the Graph-IHT algorithm
(Zhou and Chen, ICDM, 2016)

\[ \mathcal{M}(G, k = 5) = \{ y | y \in [0,1]^n, \text{supp}(y) \in \mathcal{M}(G, k = 5) \} \]
Illustration of the Graph-IHT algorithm

(Zhou and Chen, ICDM, 2016)

Network instance $\mathcal{G}$

$\mathcal{M}(\mathcal{G}, k = 5) = \{y \mid y \in [0,1]^n, \text{supp}(y) \in \mathcal{M}(\mathcal{G}, k = 5)\}$
Illustration of the Graph-IHT algorithm

(Zhou and Chen, ICDM, 2016)

Network instance $G$

$M(G, k = 5) = \{y | y \in [0,1]^n, \text{supp}(y) \in M(G, k = 5)\}$
Illustration of the Graph-IHT algorithm

(Zhou and Chen, ICDM, 2016)

\[
\mathcal{M}(G, k = 5) = \{y \mid y \in [0,1]^n, \text{supp}(y) \in \mathcal{M}(G, k = 5)\}
\]
Illustration of the Graph-IHT algorithm

(Zhou and Chen, ICDM, 2016)

Network instance $\mathbb{G}$

$\mathcal{M}(\mathbb{G}, k = 5) = \{y | y \in [0,1]^n, \text{supp}(y) \in \mathcal{M}(\mathbb{G}, k = 5)\}$
Illustration of the Graph-IHT algorithm

(Zhou and Chen, ICDM, 2016)

\[ \mathcal{M}(G, k = 5) = \{ y | y \in [0,1]^n, \text{supp}(y) \in \mathcal{M}(G, k = 5) \} \]
Illustration of the Graph-IHT algorithm
(Zhou and Chen, ICDM, 2016)

Network instance $\mathcal{G}$

$\mathcal{M}(\mathcal{G}, k = 5) = \{y | y \in [0,1]^n, \text{supp}(y) \in \mathcal{M}(\mathcal{G}, k = 5)\}$
Theoretical Guarantees

(Zhou and Chen, ICDM, 2016)

- The proposed algorithms have the following nice theoretical properties
  - Nearly-linear time complexity.
  - Let \( \mathbf{y}^* \) be the optimal solution of the relaxed problem. Under practical assumptions, we have the tight error bound

    \[
    \| \mathbf{y}^* - \mathbf{y}_i \|_2 \leq c \cdot \| \nabla I f(\mathbf{y}^*) \|_2
    \]

    where

    - \( c \) is a constant value, and
    - \( I = \arg \max_S \| \nabla_S f(\mathbf{y}^*) \|_2 \) s.t. \( S \) satisfies the predefined topological constraint.
Experiments

- Four real datasets for anomalous subgraph detection

Comparison on scores of the identified subgraphs

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of Nodes</th>
<th># of Edges</th>
<th># of snapshots</th>
</tr>
</thead>
<tbody>
<tr>
<td>BWSN</td>
<td>12,527</td>
<td>14831</td>
<td>hourly: 8</td>
</tr>
<tr>
<td>CiteHepPh</td>
<td>11,895</td>
<td>76,284</td>
<td>yearly: 11</td>
</tr>
<tr>
<td>RoadTraffic</td>
<td>1,723</td>
<td>5,301</td>
<td>per-15-min: 68×304</td>
</tr>
<tr>
<td>ChicagoCrime</td>
<td>46,357</td>
<td>168,020</td>
<td>yearly: 15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>BWSN</th>
<th>CitHepPh</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Kulldorff</td>
<td>EMS</td>
</tr>
<tr>
<td>GRAPH-GHTP</td>
<td>1097.15</td>
<td>21.56</td>
</tr>
<tr>
<td>GraphLaplacian</td>
<td>474.96</td>
<td>14.89</td>
</tr>
<tr>
<td>EventTree</td>
<td>834.59</td>
<td>20.25</td>
</tr>
<tr>
<td>DepthFirstGraphScan</td>
<td>735.85</td>
<td>20.41</td>
</tr>
<tr>
<td>NPHGS</td>
<td>541.13</td>
<td>16.90</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Traffic</th>
<th>ChicagoCrime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EMS</td>
<td>Run Time</td>
</tr>
<tr>
<td>GRAPH-GHTP</td>
<td>20.45</td>
<td>22.25</td>
</tr>
<tr>
<td>GraphLaplacian</td>
<td>5.40</td>
<td>291.75</td>
</tr>
<tr>
<td>EventTree</td>
<td>12.40</td>
<td>5.02</td>
</tr>
<tr>
<td>DepthFirstGraphScan</td>
<td>8.13</td>
<td>47.73</td>
</tr>
<tr>
<td>NPHGS</td>
<td>6.28</td>
<td>0.22</td>
</tr>
</tbody>
</table>
Review of other methods

• Scalable anomaly ranking of attributed neighborhoods  (Perozzi and Akoglu, SDM, 2016)
  • Rank a predefined set of neighborhoods (subgraphs) based on internal connectivity, boundary, and node-level attributes in quadratic time in the neighborhood size.

• Focused cluster or subgraph outlier detection  (Perozzi et al., KDD, 2016)
  • Given an initial set of nodes provided by a user
  • Step 1: Identify a subset of attributes that the given nodes agree on (called “focus attributes”)
  • Step 2: Find densely connected subgraphs that also agree on these attributes (called “focused clusters”)
Focused subgraph outlier detection

(Perozzi et al., KDD, 2016)

- Finding nodes to cluster around

\[
\text{for each } (i, j) \in E \text{ do}
\]
\[
w(i, j) = \frac{1}{1 + \sqrt{(f_i - f_j)^T \text{diag}(\beta)(f_i - f_j)}}
\]

- Highly weighted edges are reserved

- The connected components are considered as seeds
Focused subgraph outlier detection

(Perozzi et al., KDD, 2016)

1. Clustering objective: subgraph conductance weighted by focus

\[ F(S) = \frac{\text{WeightedOutDegree}(S)}{\text{WeightedDensity}(S)} \]

2. At each edge in subgraph expansion
   1. Examine boundary nodes
   2. Add node with the best marginal gain
Disney: amazon co-purchase network
(Perozzi et al., KDD, 2016)

The detected subgraphs focus on attributes related to popularity
(sales rank, number of reviews, etc)
Political blogs citation network

(Perozzi et al., KDD, 2016)

A focused cluster of liberal blogs in Pol-Blogs with a focus on Iraq ware debate
Part I: References


Part I: References


Part I: References


Part I: References


5 minutes break: Q/A