CS 6604: Data Mining Large Networks and Time-series

B. Aditya Prakash

Lecture #10: Immunization in Networks
Research Theme – Public Health

ANALYSIS
Will an epidemic happen?

DATA
Modeling # patient transfers

POLICY/ACTION
How to control out-breaks?
NODE RESILIENCY
Network Resilience

- Remove nodes one-by-one

- What is resilience of a graph?
  - The remaining nodes are in a CC
  - Average distance does not increase
  - ...
  - Other metrics
Random Graphs

- $p = 0$: Empty Graph
- $p = 1/n$: GCC emerges
- $p = \log n/n$: No isolated nodes
- $p = 1$: Complete Graph
Random Graphs

- Start with an ER graph with $p > \ln n/n$
- Remove a random fraction $f$ of edges

- When will the remaining graph be connected?
  - Now we have an ER graph with $p' = p(1-f)$
  - So connected if $p(1-f) > \ln n/n$
  - Typically there is a threshold for removal. [Bollobas, 1985].
The graph still has a GCC after $f$ fraction of nodes are removed if:

$$f < f_c = 1 - \frac{1}{\frac{E[k^2]}{E[k]} - 1}$$

Special case: In ER graphs?

$$f < f_c = 1 - \frac{1}{E[k]}$$

Higher the original average degree, more resilient the network.
Breakdown threshold of PL random graphs

- $P(k) = C k^{-a}$ (K $\geq k \geq k_m$)
- If $a > 3$
  \[ f_c = 1 - \frac{1}{\frac{a-2}{a-3} k_m - 1} \]
- If $2 < a < 3$
  \[ f_c = 1 - \frac{1}{\frac{a-2}{3-a} k_m^{a-2} K^{3-a} - 1} \]

Infinite scale-free networks with $a < 3$ do not break down under random node failure. [Cohen+, 2000]
Flip side: PL networks are error-tolerant but vulnerable to attacks

BA model, degree exponent = 3

Error [Squares]: remove a random node

Attacks [Circles]: remove node with highest degree

FAST breakdown with targeted attacks

[Albert+, 2000]
Real PL Networks

- Blue: random failures ;; Red: targeted attacks

- Break down if 5% of the nodes are attacked
- Resilient to random failure of 50% of nodes
Generalization to arbitrary networks

- Reliability Polynomial
  - The probability that no connected component of \( G \) is disconnected due to random edge removals, denoted \( C(p) \)
  - Related to the Tutte polynomial
    - generating function for the number of edge sets of a given size and connected components

- Hard to compute in general!
  - Known for some simple classes
NODE IMMUNIZATION
Given: a graph $A$, virus prop. model and budget $k$; Find: $k$ ‘best’ nodes for immunization (removal).

$k = 2$

We WANT to break the network!
A possible approach

- Remove the high degree nodes
  - We know in PL networks, it breaks networks fast (if breaking == decompose into smaller components)

- How to do this in a decentralized fashion??
Acquaintance Immunization [Cohen+, 2003]

**First:** randomly choose a node

**Second:** pick a random neighbor of the previous node

Immunize the second node (the ‘acquaintance’)
Why it works?

- Remember first lecture
  - Your friends have more friends than you have 😊
Average number of friends
= ( 1 + 3 + 2 + 2 ) / 4
= 2

Average number of friends of friends
= (3 + 1 + 2 + 2 + 3 + 2 + 3 + 2)/8
= ((1x1) + (3x3) + (2x2) + (2x2))/8
= 2.25!

Source: S. Strogatz, NYT 2012
Why it works?

- Remember first lecture
  - Your friends have more friends than you have 😊

- Your friends are more likely to have a higher degree than you
In particular....

- For random PL graphs, using acquaintance immunization ~ picking nodes by degree
Can we do better?

**Given:** a graph $A$, virus prop. model and budget $k$;

**Find:** $k$ ‘best’ nodes for immunization (removal).

We WANT to break the network!
Challenges

- Given a graph $A$, budget $k$,

**Q1 (Metric)** How to measure the ‘shield-value’ for a set of nodes $(S)$?

**Q2 (Algorithm)** How to find a set of $k$ nodes with highest ‘shield-value’?
Metric

- Is the size of the GCC the best metric for immunization?
  - again we need a measure of connectivity more relevant to spreading
Vulnerability measure $\lambda$

$\lambda$ is the epidemic threshold

"Safe" 
(a) Chain($\lambda = 1.73$) 
"Vulnerable" 
(b) Star($\lambda = 2$) 
(c) Clique($\lambda = 4$) 

Increasing $\lambda$ 
Increasing vulnerability
A1: “Eigen-Drop”: an ideal shield value

Eigen-Drop(\(S\))

\[ \Delta \lambda = \lambda - \lambda_S \]

Original Graph

Without \{2, 6\}
(Q2) - Direct Algorithm too expensive!

- Immunize $k$ nodes which maximize $\Delta \lambda$

$$S = \text{argmax} \Delta \lambda$$

- Combinatorial!

- Complexity: \[ O\left(\binom{n}{k} \cdot m\right) \]

  - Example:
    - 1,000 nodes, with 10,000 edges
    - It takes 0.01 seconds to compute $\lambda$
    - It takes 2,615 years to find 5-best nodes!
Solution: Part 1 [Tong+, 2010]

- **Approximate Eigen-drop (Δ λ)**

\[ Δ \lambda \approx \hat{SV}(S) = \sum_{i \in S} 2\lambda u(i)^2 - \sum_{i,j \in S} A(i,j)u(i)u(j) \]

- Result using Matrix perturbation theory

\[-u(i) == ‘eigenscore’ \]

\[ \sim pagerank(i) \]
\[
\sum_{i \in S} 2 \lambda u(i)^2 - \sum_{i,j \in S} A(i,j)u(i)u(j)
\]

**P1:** node importance  
**P2:** set diversity

Original Graph  
Select by P1  
Select by P1+P2
Solution: Part 2: NetShield

- NetShield: Greedily add best node at each step

1. NetShield is near-optimal (w.r.t. max $\overline{SV(S)}$)
2. NetShield is $O(nk^2+m)$

Footnote: near-optimal means $\overline{SV(S)} \geq (1-1/e) \overline{SV(S_{\text{Opt}})}$
Why *Netshield* is Near-Optimal?

Sub-Modular (i.e., Diminishing Returns)

Marginal benefit of deleting \{5,6\}  Marginal benefit of deleting \{5,6\}

Benefit of deleting \{1,2\}  Benefit of deleting \{1,2, 3,4\}

\[ \Delta \geq \Delta' \]  \[ \iff \] Sub-Modular (i.e., Diminishing Returns)

Source: H. Tong

Prakash 2015  CS 6604:DM Large Networks & Time-Series
M1: Why $SV(S)$ is sub-modular?

Source: H. Tong

Prakash 2015 CS 6604:DM Large Networks & Time-Series 30
M1: Why $SV(S)$ is sub-modular?

$$Br(S_{blue} \cup S_{green}) - Br(S_{green}) =$$

$$\sum_{i \in S_{blue}} 2\lambda_i u(i)^2 - \sum_{i, j \in S_{blue}} A(i, j)u(i)u(j) -$$

$$\sum_{i \in S_{blue}, j \in S_{green}} A(i, j)u(i)u(j) - \sum_{j \in S_{blue}, i \in S_{green}} A(i, j)u(i)u(j)$$

Only purple term depends on $\{1, 2\}$!

Source: H. Tong
M1: Why $SV(S)$ is sub-modular?

Marginal Benefit $= \text{Blue} - \text{Purple}$

More Green $\iff$ More Purple $\iff$ Less Red

Marginal Benefit of Left $\geq$ Marginal Benefit of Right

Footnote: greens are nodes already deleted; blue $\{5,6\}$ nodes are nodes to be deleted
Why *Netshield* is Near-Optimal?

Sub-Modular (i.e., Diminishing Returns) \( \geq \)

Theorem: *k*-step greedy alg. to maximize a sub-modular function guarantees (1-1/e) optimal [Nemhauster+ 78]

Source: H. Tong
Quality of Netshield

Quality of Netshield

Optimal

Netshield

(1-1/e) x Optimal

(k)

Eig-Drop

(better)
Experiment: Immunization quality

Log(fraction of infected nodes)

- PageRank
- Betweenness (shortest path)
- Degree
- Acquaintance
- Eigs (=HITS)

Lower is better

Prakash 2015
SUBMODULAR FUNCTIONS
Submodular Functions

- Set Function $f(S)$ is sub-modular if
  
  $\forall \ S \subseteq T$
  
  $f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T)$

  Gain of adding a node to a small set  Gain of adding a node to a large set

  - Sort of like convexity, sort of like concavity
  - Continuous Optimization: Convex Functions ::
  
  Discrete Optimization: Submodular Functions

- Useful fact

  If $f_1(x), f_2(x), \ldots f_k(x)$ are sub modular, then for any $c_1, c_2, \ldots, c_k \geq 0$, $F(x) = \sum_i c_i f_i(x)$ is submodular as well.
Example

- Function \( f(S) = \left| \bigcup_{i \in S} X_i \right| \)
  - For sets \( X_1, X_2, \ldots, X_m \)

- Is it Submodular?
  - Yes!

- Many others
  - Influence maximization, Facility Locations, Entropy etc...
Submodular Monotone Functions

- If $\forall S \subseteq T, \quad f(S) \geq f(T)$
Efficient Optimization

- How to solve for submodular, monotone f:
  \[ S^* = \arg\max_{S} f(S), \text{ s.t. } |S| = k \]

- BAD news
  - Typically NP-Hard

- GOOD news
  - Greedy gradient based method (‘hill climbing’) gives an approximation algorithm!
Greedy Hill Climbing

- At each iteration, pick the element which gives the best marginal gain

\[ S_i = S_{i-1} \cup \text{arg max}_u f(S_{i-1} \cup u) \]

- Start with \( S_0 = \{\} \) (empty set)

- Guarantee [Nemhauser+ 78, Kempe+ 03]: Greedy Hill-Climbing gives a \((1-1/e \sim 0.63)\) approximation \( S \) i.e. \( f(S) \geq (1-1/e) \times \text{OPT} \)
Diminishing Returns

\[ f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T) \]

Gain of adding a node to a small set \quad \geq \quad Gain of adding a node to a large set

Adding \( u \) to \( T \) helps less than adding it to \( S \)!
Why Hill-Climbing works?

- Claim 1: \( (B = \{b_1, b_2, \ldots, b_k\}) \)

\[
f(A \cup B) - f(A) \leq \sum_{i}^{k} (f(A \cup \{b_i\}) - f(A))
\]

- Proof:

Let \( B_i = \{b_1, \ldots, b_i\} \), then \( B_k = B \)

\[
f(A \cup B) - f(A) = \sum_{i}^{k} (f(A \cup B_i) - f(A \cup B_{i-1}))
\]

\[
= \sum_{i}^{k} (f(A \cup B_{i-1} \cup \{b_i\}) - f(A \cup B_{i-1}))
\]

\[
\leq \sum_{i}^{k} (f(A \cup \{b_i\}) - f(A))
\]
Marginal gain at Step i:
\[ \delta_i = f(S_i) - f(S_{i-1}) \]

- We know

\[
    f(T) \leq f(S_i \cup T)
    = f(S_i \cup T) - f(S_i) + f(S_i)
    \leq \sum_{j=1}^{k} (f(S_i \cup \{t_j\}) - f(S_i)) + f(S_i)
    \leq \sum_{j=1}^{k} \delta_{i+1} + f(S_i)
\]

Using the hill climbing assumption:
\( \delta_{i+1} \) corresponds to the best element
Marginal gain at Step i:
\[ \delta_i = f(S_i) - f(S_{i-1}) \]

- We know

\[
\begin{align*}
  f(T) & \leq f(S_i \cup T) \\
  & = f(S_i \cup T) - f(S_i) + f(S_i) \\
  & \leq \sum_{j=1}^{k} (f(S_i \cup \{t_j\}) - f(S_i)) + f(S_i) \\
  & \leq \sum_{j=1}^{k} \delta_{i+1} + f(S_i) \\
  & = f(S_i) + k \delta_{i+1} \\
  \Rightarrow \delta_{i+1} & \geq \frac{1}{k} (f(T) - f(S_i))
\end{align*}
\]
What is \( f(S_{i+1}) \)?

- From previous slide
  \[
  \delta_{i+1} \geq \frac{1}{k} (f(T) - f(S_i))
  \]
  \[
  f(S_{i+1}) = f(S_i) + \delta_{i+1}
  \]
  \[
  \geq f(S_i) + \frac{1}{k} (f(T) - f(S_i))
  \]
  \[
  = (1 - \frac{1}{k}) f(S_i) + \frac{1}{k} f(T)
  \]

So what happens when \( i+1 = k \)? i.e. when the hill-climbing terminates?
What is $f(S_k)$?

- **Claim 2:** 
  
  $$f(S_i) \geq (1 - (1 - \frac{1}{k})^i) f(T)$$

- **Proof:** (by induction)

For $i=0$

$$f(S_0) = f(\emptyset) = 0 = (1 - (1 - \frac{1}{k})^0) f(T)$$
What is $f(S_k)$?

- **Claim 2:** $f(S_i) \geq (1 - (1 - \frac{1}{k})^i) f(T)$

- **Proof:** (by induction)

  $f(S_{i+1}) = (1 - \frac{1}{k}) f(S_i) + \frac{1}{k} f(T)$

  $\geq (1 - \frac{1}{k})(1 - (1 - \frac{1}{k})^i) f(T) + \frac{1}{k} f(T)$

  $= (1 - (1 - \frac{1}{k})^{i+1}) f(T)$  

  **Inductive claim**
What is $f(S_k)$?

- So we get:

\[ f(S) = f(S_k) \geq (1 - (1 - \frac{1}{k})^k) f(T) \leq \frac{1}{e} \]

\[ f(S_k) \geq (1 - \frac{1}{e}) f(T) \]

QED
Many extensions

- Without cardinality constraints
  - Like knapsack constraints
- Robust online optimization
- Using multi-linear extensions
- .....
IMMUNIZATION ON TIME-VARYING GRAPHS
Immunization

- Our solution
  - Recall Theorem
  - Simple: reduce

- goal: max eigendrop \( \Delta \lambda \)
  \[
  \Delta \lambda = \lambda_{\text{before}} - \lambda_{\text{after}}
  \]

- Comparison - But: No competing policy
- We propose and evaluate many policies
Performance of policies

Footprint after k=6 immunizations

- Greedy-DmaxA
- Greedy-DavgA
- Greedy-AavgA
- Greedy-S
- Optimal

Lower is better

Footprint

MIT Reality Mining
Q2: Immunization Policies

- Optimal
- Greedy-DmaxA
- Greedy-DavgA
- Greedy-AavgA
- Greedy-S

Footprint after k=6 immunizations
Optimal – exhaustive enumeration
  – Test each set of k nodes
  – Remove & Re-compute
  – pick best ‘eigendrop’ ($\Delta \lambda$)
  – $\binom{n}{k}$ sets, hence combinatorial, $O\left(\frac{n^3}{k}\right)$

– Super-expensive!
**Immunization Policies**

- **Greedy-S**
  - Node which causes largest $\Delta \lambda$
  - Remove node
  - Repeat till $k$ nodes are removed

- Can be expensive: $O(k.n.n^3)$

- But performs well!
Immunization Policies

- Greedy-DavgA
  - Node with highest degree in
  - Remove node
  - Repeat

- Cheap – $O(k \cdot n \cdot E)$

- Other policies in the paper

---

Footprint after $k=6$ immunizations
Lower is better
Discussion

- Why does Greedy-DavgA perform well?
Greedy-DavgA (details)

Lemma 3. \((1 - 2\delta)I + 2\beta A_{\text{avg}}\) is a first-order approximation of the \(S\) matrix.

Proof. Note that \((T = 2)\),

\[
S = S_1 \times S_2 = ((1 - \delta)I + \beta A_1) \times ((1 - \delta)I + \beta A_2) = (1 - \delta)^2 I + (1 - \delta)\beta(A_1 + A_2) + \beta^2 A_1 A_2 \approx (1 - 2\delta)I + \beta(A_1 + A_2) = (1 - 2\delta)I + 2\beta \left(\frac{A_1 + A_2}{2}\right)
\]

- Static case with \(A_{\text{avg}} \approx\) Original time-varying case (upto first order)
  - Can use any immunization technique for static graphs
EDGE IMMUNIZATION
Q: How to guild propagation by opt. link structure?
   - Q1: Understand tipping point
   - Q2: Minimize the propagation
   - Q3: Maximize the propagation

Recap: Flu/Virus Propagation

1: Sneeze to neighbors
2: Some neighbors → Sick
3: Try to recover

Healthy  Sick

Contact
Recap: Vulnerability measure $\lambda$ [ICDM 2011, PKDD 2010]

$\lambda$ is the epidemic threshold

Increasing $\lambda$

Increasing vulnerability

"Safe" (a)Chain($\lambda = 1.73$)  "Vulnerable" (b)Star($\lambda = 2$)  "Deadly" (c)Clique($\lambda = 4$)
Minimizing Propagation: Edge Deletion

- Given: a graph $A$, virus prop model and budget $k$;
- Find: delete $k$ ‘best’ edges from $A$ to minimize $\lambda$

Challenge: We need $O\left(\binom{m}{k}m\right)$ time for Naïve method!
Q: How to find k best edges to delete efficiently? [Tong+ 2012]

• Our Sol: By 1st order perturbation, we have

$$\lambda - \lambda_s \approx Mv(S) = c \sum_{e \in S} u(i_e)v(j_e)$$

Left eigen-score of source

Right eigen-score of target

• Observations:
  • Only need eigen-computation once
  • Impact of different edges are de-coupled
Minimizing Propagation: Evaluations

Log (Infected Ratio)

Data set: Oregon Autonomous System Graph (14K node, 61K edges)
Discussions: Node Deletion vs. Edge Deletion

• Observations:
  • Node or Edge Deletion $\rightarrow \lambda$ Decrease
  • Nodes on $A$ = Edges on its line graph $L(A)$

• Questions?
  • Edge Deletion on $A$ = Node Deletion on $L(A)$?
  • Which strategy is better (when both feasible)?
Discussions: Node Deletion vs. Edge Deletion

• Q: Is Edge Deletion on $A = \text{Node Deletion on } L(A)$?
• A: Yes!

Theorem: Line Graph Spectrum.
Eigenvalue of $A \rightarrow$ Eigenvalue of $L(A)$
Discussions: Node Deletion vs. Edge Deletion

Q: Which strategy is better (when both feasible)?

A: Edge Deletion > Node Deletion

Green: Node Deletion (e.g., shutdown a twitter account)

Red: Edge Deletion (e.g., un-friend two users)
Maximizing Propagation: Edge Addition

• Given: a graph $A$, virus prop model and budget $k$;
• Find: add $k$ ‘best’ new edges into $A$.

• By 1st order perturbation, we have

$$\lambda_s - \lambda \approx Gv(S) = c \sum_{e \in S} u(i_e)v(j_e)$$

• So, we are done $\rightarrow$ need $O(n^2-m)$ complexity
Maximizing Propagation: Edge Addition

\[ \lambda_s - \lambda \approx G_v(S) = c \sum_{e \in S} u(i_e)v(j_e) \]

- **Q:** How to Find k new edges w/ highest \( G_v(S) \) ?

- **A:** Modified Fagin’s algorithm

Time Complexity: \( O(m + nt + kt^2) \), \( t = \max(k, d) \)

:existing edge
Detour
Problem Definition: Fagin’s Algorithm 2003

- Given a collection of objects, our goal is to find Top-k objects, whose scores are greater than the remaining objects.

Source: Ronald Fagin, IBM
A sample set of Databases

<table>
<thead>
<tr>
<th>Object</th>
<th>Area ($x_3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object</th>
<th>Roundness ($x_2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object</th>
<th>Redness ($x_1$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0.67</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Every subsystem is sorted by the grade it holds.
Top-k Object Problem

- Naïve Algorithm
- Fagin’s Algorithm
Naïve Algorithm

- Basic Idea:
  - For each object, use the aggregation function to get the score
  - According to the scores, get the top $k$. 
Questions

- Do we need to count the score for every object in the database?
- Can we SAFELY ignore some objects whose scores are lower than what we already have?
Fagin’s Algorithm

- Do Sorted access in parallel at all the lists
- Stop when we have k objects which appear in all the lists
- Calculate score value of all the objects
- Compute Top-k objects
Example: Fagin’s Algorithm

Objects appear in every list:

\{ \}

Objects seen so far:

\{\text{\ding{85}}, \text{\ding{86}}\}

\begin{array}{c|c|c|c}
\text{Object} & \text{Redness}\ (x_1) & \text{Object} & \text{Roundness}\ (x_2) & \text{Object} & \text{Area}\ (x_3) \\
\hline
\text{\textcolor{red}{\ding{85}}} & 1 & \text{\textcolor{red}{\ding{85}}} & 1 & \text{\textcolor{red}{\ding{86}}} & 0.95 \\
\hline
\text{\textcolor{red}{\ding{86}}} & 1 & \text{\textcolor{red}{\ding{86}}} & 1 & \text{\textcolor{red}{\ding{85}}} & 0.85 \\
\hline
\text{\textcolor{red}{\ding{85}}} & 0.67 & \text{\textcolor{red}{\ding{85}}} & 0.5 & \text{\textcolor{red}{\ding{86}}} & 0.75 \\
\hline
\text{\textcolor{red}{\ding{85}}} & 0.6 & \text{\textcolor{red}{\ding{85}}} & 0.2 & \text{\textcolor{red}{\ding{86}}} & 0.3 \\
\hline
\text{\textcolor{red}{\ding{85}}} & 0.5 & \text{\textcolor{red}{\ding{85}}} & 0 & \text{\textcolor{red}{\ding{86}}} & 0.1 \\
\hline
\end{array}

\[ k = 3 \]

Source: Ronald Fagin, IBM
Example: Fagin’s Algorithm

Objects appear in every list:

\{ \} 

Objects seen so far:

\{ \text{●}, \text{□}, \text{□}, \text{□}, \text{○} \} 

<table>
<thead>
<tr>
<th>Object</th>
<th>Redness ($x_1$)</th>
<th>Roundness ($x_2$)</th>
<th>Area ($x_3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>●</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>□</td>
<td>1</td>
<td>1</td>
<td>0.95</td>
</tr>
<tr>
<td>□</td>
<td>0.67</td>
<td>0.5</td>
<td>0.85</td>
</tr>
<tr>
<td>□</td>
<td>0.6</td>
<td>0.2</td>
<td>0.75</td>
</tr>
<tr>
<td>□</td>
<td>0.5</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>□</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
</tbody>
</table>

$k = 3$
Example: Fagin’s Algorithm

Objects appear in every list:

\{ \text{..} \}

Objects seen so far:

\{ \text{..} \}

$k = 3$
Example: Fagin’s Algorithm

Objects appear in every list:

\{\text{oval}, \text{rectangle}, \text{circle}\}

We got enough objects

Objects seen so far:

\{\text{circle}, \text{oval}, \text{rectangle}, \text{circle}, \text{oval}, \text{circle}\}

\[k = 3\]
Example: Fagin’s Algorithm

Objects appear in every list:

\{ \text{red}, \text{red}, \text{red} \}

We got enough objects

For all these, calculate the score and get the Top-k

\[ k = 3 \]

<table>
<thead>
<tr>
<th>Object</th>
<th>Redness ((x_1))</th>
<th>Object</th>
<th>Roundness ((x_2))</th>
<th>Object</th>
<th>Area ((x_3))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>0.67</td>
<td></td>
<td></td>
<td></td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td></td>
<td></td>
<td></td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>0.1</td>
</tr>
</tbody>
</table>
End Detour
Maximizing Propagation: Evaluation

Log (Infected Ratio)

Our Method

better

Time Ticks

Original

CompDelete

Rand

CompPage

CompEigs

CompDeg

Our Method
Fractional Asymmetric Immunization
[Prakash+ 2013]

- Fractional Effect \[ f(x) = 0.5^x \]
- Asymmetric Effect

\# antidotes = 3
Now: Fractional Asymmetric Immunization

- Fractional Effect \([ f(x) = 0.5^x ]\)
- Asymmetric Effect

\# antidotes = 3
Fractional Asymmetric Immunization

- Fractional Effect \[ f(x) = 0.5^x \]
- Asymmetric Effect

# antidotes = 3
Fractional Asymmetric Immunization

Drug-resistant Bacteria (like XDR-TB)

Hospital → Another Hospital

Prakash 2015
Fractional Asymmetric Immunization

Drug-resistant Bacteria (like XDR-TB)

Hospital

\[ f \]

Another Hospital

Prakash 2015

CS 6604:DM Large Networks & Time-Series
Fractional Asymmetric Immunization

**Problem:** Given $k$ units of disinfectant, how to distribute them to maximize hospitals saved?
Our approach (1)

- Upper bound the ‘cost’ (number of infected nodes)

**Lemma 1.** In the SI virus spreading model on a graph:

\[ \sigma(t) \leq (1 + \lambda_A)^t \sigma(0) \]

where \( \sigma(t) \) is the expected number of infected nodes at time \( t > 0 \) and \( \sigma(0) \) is a scalar depending just on the initial conditions (independent of \( t \)).
Our Algorithm “SMART-ALLOC”

~6x fewer!

[US-MEDICARE NETWORK 2005]
- Each circle is a hospital, ~3000 hospitals
- More than 30,000 patients transferred

CURRENT PRACTICE

SMART-ALLOC

Prakash 2015
Running Time

Wall-Clock Time

Lower is better

> 1 week

> 30,000x speed-up!

14 secs

Simulations

SMART-ALLOC

Lower is better
Experiments

Lower is better

PENN-NETWORK

SECOND-LIFE

\[ \approx 5 \times \]

\[ \approx 2.5 \times \]

\( K = 200 \)

\( K = 2000 \)