Controlling diffusion processes on networks

Anil Vullikanti
Department of Computer Science
Biocomplexity Institute & Initiative
University of Virginia

November 12, 2020

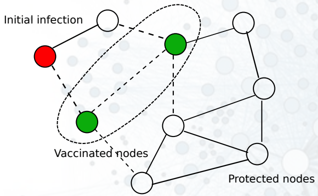
Recap

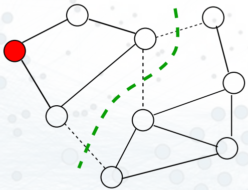
- Stochastic diffusion models, independent cascades
- Influence maximization problem, $F(S)$
- Submodularity
- Greedy algorithm
- Monte-Carlo sampling to approximate $F(S)$
- Greedy gives a $(1 - 1/e)$-approximation to a submodular function

Outline for lecture

- $F(S)$ is submodular
- Minimizing diffusion
- Summary

Interventions to control epidemic spread





Pharmaceutical interventions (PI)

- use of prophylactic vaccinations and anti-viral drugs
- modeled as node deletions

Non Pharmaceutical interventions (NPI)

- Reducing contacts by social distancing, school or work place closure, or isolation.
- Modeled as edge deletions

Intervention design problems

Given limited budget B for node/edge removal, minimize the epidemic outbreak

SIS model

- Recall
  - Nodes in Susceptible (S) or Infectious (I) states
  - Prob infected node causes susceptible neighbor to become infected: β
  - Prob infected node becomes susceptible: δ
- Limiting state: all nodes in S
- Natural quantity to consider: how long does the outbreak last
  - Die out time τ

Characterizing τ (informal)

- $A = A(G)$: adjacency matrix of G with eigenvalues $\lambda_1(G) \geq \lambda_2(G) \geq \dots \lambda_n(G)$
  - We will sometimes just refer to these as A, λ_1
- λ_1 referred to as the *spectral radius* of G
- Let $T = \delta/\beta$

Sufficient condition [A. Ganesh, L. Massoulie and D. Towsley, *IEEE INFOCOM*, 2005]

If $\lambda_1 < T$: epidemic dies out "fast"

Formally

Lemma (Sufficient condition for fast recovery)

Suppose $\lambda_1 < T$. Then, the time to extinction τ satisfies

$$E[\tau] \leq \frac{\log n + 1}{1 - \lambda_1 / T}$$

Formally

Lemma (Sufficient condition for fast recovery)

Suppose $\lambda_1 < T$. Then, the time to extinction τ satisfies

$$E[\tau] \leq \frac{\log n + 1}{1 - \lambda_1 / T}$$

- What if $\lambda_1 > T$: not a "if and only if" result

Formally

Lemma (Sufficient condition for fast recovery)

Suppose $\lambda_1 < T$. Then, the time to extinction τ satisfies

$$E[\tau] \leq \frac{\log n + 1}{1 - \lambda_1/T}$$

- What if $\lambda_1 > T$: not a "if and only if" result
- Generalized isoperimetric constant: $\eta(m) = \inf_{S \subset V, |S| \leq m} \frac{E(S,\bar{S})}{|S|}$

Formally

Lemma (Sufficient condition for fast recovery)

Suppose $\lambda_1 < T$. Then, the time to extinction τ satisfies

$$E[\tau] \leq \frac{\log n + 1}{1 - \lambda_1 / T}$$

- What if $\lambda_1 > T$: not a "if and only if" result
- Generalized isoperimetric constant: $\eta(m) = \inf_{S \subset V, |S| \leq m} \frac{E(S, \bar{S})}{|S|}$

Lemma (Sufficient condition for lasting infection)

Formally

Lemma (Sufficient condition for fast recovery)

Suppose $\lambda_1 < T$. Then, the time to extinction τ satisfies

$$E[\tau] \leq \frac{\log n + 1}{1 - \lambda_1 / T}$$

- What if $\lambda_1 > T$: not a "if and only if" result
- Generalized isoperimetric constant: $\eta(m) = \inf_{S \subset V, |S| \leq m} \frac{E(S, \bar{S})}{|S|}$

Lemma (Sufficient condition for lasting infection)

If $\eta(m)$ is "large", then the epidemic lasts for "long"

Formally

Lemma (Sufficient condition for fast recovery)

Suppose $\lambda_1 < T$. Then, the time to extinction τ satisfies

$$E[\tau] \leq \frac{\log n + 1}{1 - \lambda_1 / T}$$

- What if $\lambda_1 > T$: not a "if and only if" result
- Generalized isoperimetric constant: $\eta(m) = \inf_{S \subset V, |S| \leq m} \frac{E(S,\bar{S})}{|S|}$

Lemma (Sufficient condition for lasting infection)

If $r = \frac{\delta}{\beta \eta(m)} < 1$, then

$$\Pr[\tau > r^{-m+1}/(2m)] \geq \frac{1-r}{e}(1 + O(r^m))$$

Implications for different network models

- Hypercube: $\lambda_1 = \log_2 n$, and $\eta(m) = (1-a)\log_2 n$ for $m = n^a$
  - Fast die out if $\beta < \frac{1}{\log_2 n}$, slow die out if $\beta > \frac{1}{(1-a)\log_2 n}$
- Erdős-Rényi model: $\lambda_1 = (1+o(1))np = (1+o(1))d$ and $\eta(m) = (1+o(1))(1-\alpha)d$ where $m/n \to \alpha$
  - Fast die out if $\beta < \frac{1}{(1+o(1))d}$, slow die out if $\beta > \frac{1}{(1+o(1))(1-\alpha)d}$
- Power law graphs (Chung-Lu model): assume degree distribution with power law exponent $\gamma > 2.5$
  - $E[\tau] = O(\log n)$ if $\beta < (1-u)/\sqrt{m}$ and $E[\tau]$ exponential if $\beta > m^\alpha/\sqrt{m}$ for some $u, \alpha \in (0,1)$ and $m = n^\lambda$, for $\lambda \in (0, \frac{1}{\gamma-1})$
- In general, gap between necessary and sufficient conditions for epidemic to last long

Note on derivations

There exist three different approaches for deriving spectral radius characterization

- Continuous time approximation [A. Ganesh, L. Massoulie and D. Towsley, *IEEE INFOCOM*, 2005]
  - Gives both upper and lower bounds on τ (in terms of λ_1 and $\eta(m)$)
- Independence assumption [D. Chakrabarti, et al., *ACM TISS*, 2008]
  - Only gives condition in terms of λ_1
  - Extended to other models beyond SIS [BA Prakash, et al., *KAIS*, 2012]
- Mean-field assumption [P. Van Mieghem, J. Omic, and R. Kooij. *IEEE/ACM ToN*, 2009]

Implications

- Low spectral radius $\Rightarrow$ epidemic dies out faster
- Strategy to control outbreak: reduce spectral radius
- λ_1 is monotone: decreases if nodes or edges are removed

Implications

- Low spectral radius $\Rightarrow$ epidemic dies out faster
- Strategy to control outbreak: reduce spectral radius
- λ_1 is monotone: decreases if nodes or edges are removed

Spectral Radius Minimization (SRM) problem

- *Given*: graph G=(V, E), threshold T and cost $c(v)$ for each node v
- *Objective*: choose cheapest set $S \subseteq V$ of nodes to delete (i.e., vaccinate) so that $\lambda_1(G[V - S]) \leq T$.

Similarly, edge version

Approximation algorithms

- SRM is NP-hard, in general
- Therefore, approximation algorithms
- Let $V_{opt}(T)$ be an optimum solution for a given T
- α-approximation if the algorithm picks set S with cost $c(S) \leq \alpha \cdot c(V_{opt}(T))$, and $\lambda_1(G[V - S]) \leq T$
- (α, μ)-approximation if the algorithm picks set S with cost $c(S) \leq \alpha \cdot c(V_{opt}(T))$, and $\lambda_1(G[V - S]) \leq \mu T$

Many heuristics

- Pick nodes in non-increasing order of:

Many heuristics

- Pick nodes in non-increasing order of:
  - *Degree* [Van Mieghem et al., 2011]

Many heuristics

- Pick nodes in non-increasing order of:
  - *Degree* [Van Mieghem et al., 2011]
  - *Eigenscore* $x_1(i)$, where $\mathbf{x}_1$ is the first eigenvector [Tong et al., 2012], [Van Mieghem et al., 2011]

Many heuristics

- Pick nodes in non-increasing order of:
  - *Degree* [Van Mieghem et al., 2011]
  - *Eigenscore* $x_1(i)$, where $\mathbf{x}_1$ is the first eigenvector [Tong et al., 2012], [Van Mieghem et al., 2011]
  - *Pagerank* [Tong et al., 2012]

Many heuristics

- Pick nodes in non-increasing order of:
  - *Degree* [Van Mieghem et al., 2011]
  - *Eigenscore* $x_1(i)$, where $\mathbf{x}_1$ is the first eigenvector [Tong et al., 2012], [Van Mieghem et al., 2011]
  - *Pagerank* [Tong et al., 2012]
- Hybrid rule: pick node based on either degree or eigenscore, depending on reduction in λ_1

Many heuristics

- Pick nodes in non-increasing order of:
  - *Degree* [Van Mieghem et al., 2011]
  - *Eigenscore* $x_1(i)$, where $\mathbf{x}_1$ is the first eigenvector [Tong et al., 2012], [Van Mieghem et al., 2011]
  - *Pagerank* [Tong et al., 2012]
- Hybrid rule: pick node based on either degree or eigenscore, depending on reduction in λ_1

Worst case performance

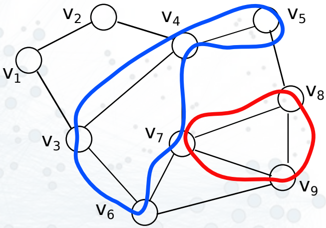
There exist instances, such that for all the above heuristics, the solution is $\Omega\left(\frac{n}{\sqrt{T}}\right)$ times the optimal

Our results[1]

- Algorithm GreedyWalk: gives an $(O(\log^2 n), 1+\epsilon)$-approximation
- Primal-dual version: gives an $(O(\log n), 1+\epsilon)$-approximation
- Constant factor approximation by semidefinite programming based rounding

Some notation and properties

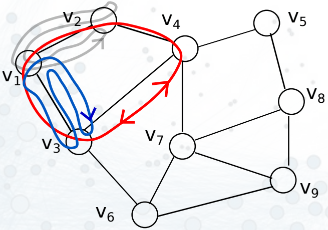
- Consider closed walks of length k
  - Start and end at same node
  - Length k of a walk: #edges in it



- Walk $v_3, v_4, v_5, v_4, v_7, v_6, v_3$ of length 6
  - Nodes can be repeated
- Walk v_7, v_8, v_9, v_7 of length 3

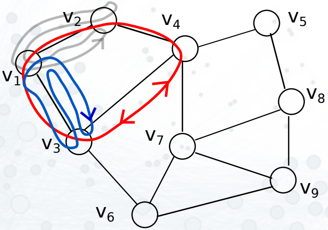
Some notation and properties

- Consider closed walks of length k
  - Start and end at same node
  - Length k of a walk: #edges in it



Some notation and properties

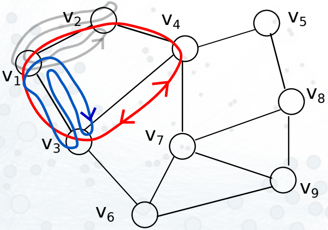
- Consider closed walks of length k
  - Start and end at same node
  - Length k of a walk: #edges in it
- Node v "hits" walk w if $v \in w$.



- Node v_1 hits the walks $w_1 = (v_1, v_2, v_1, v_2, v_1)$, $w_2 = (v_1, v_3, v_1, v_3, v_1)$, $w_3 = (v_1, v_2, v_3, v_4, v_1)$, $w_4 = (v_1, v_4, v_3, v_2, v_1)$, and four other walks

Some notation and properties

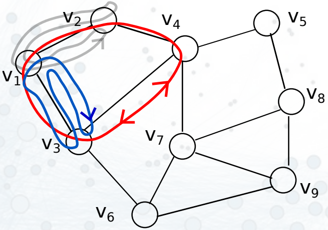
- Consider closed walks of length k
  - Start and end at same node
  - Length k of a walk: #edges in it
- Node v "hits" walk w if $v \in w$.
- $n(v, G)$: #walks (of length k) containing node v



- Node v_1 hits the walks $w_1 = (v_1, v_2, v_1, v_2, v_1)$, $w_2 = (v_1, v_3, v_1, v_3, v_1)$, $w_3 = (v_1, v_2, v_3, v_4, v_1)$, $w_4 = (v_1, v_4, v_3, v_2, v_1)$, and four other walks
- $n(v_1, G) = 8$

Some notation and properties

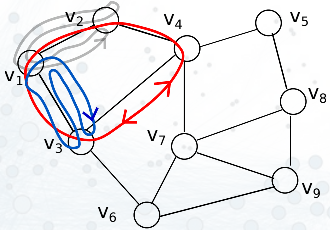
- Consider closed walks of length k
  - Start and end at same node
  - Length k of a walk: #edges in it
- Node v "hits" walk w if $v \in w$.
- $n(v, G)$: #walks (of length k) containing node v
- $d(w)$: #distinct nodes in walk w



- Node v_1 hits the walks $w_1 = (v_1, v_2, v_1, v_2, v_1)$, $w_2 = (v_1, v_3, v_1, v_3, v_1)$, $w_3 = (v_1, v_2, v_3, v_4, v_1)$, $w_4 = (v_1, v_4, v_3, v_2, v_1)$, and four other walks
- $n(v_1, G) = 8$
- $d(w_1) = 2$, $d(w_4) = 4$

Some notation and properties

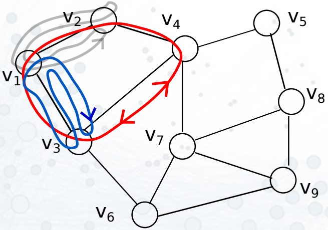
- Consider closed walks of length k
  - Start and end at same node
  - Length k of a walk: #edges in it
- Node v "hits" walk w if $v \in w$.
- $n(v, G)$: #walks (of length k) containing node v
- $d(w)$: #distinct nodes in walk w
- Let $\mathcal{W}_k(G)$ denote the set of closed walks of length k
- Let $W_k(G) = |\mathcal{W}_k(G)|$



- Node v_1 hits the walks $w_1 = (v_1, v_2, v_1, v_2, v_1)$, $w_2 = (v_1, v_3, v_1, v_3, v_1)$, $w_3 = (v_1, v_2, v_3, v_4, v_1)$, $w_4 = (v_1, v_4, v_3, v_2, v_1)$, and four other walks
- $n(v_1, G) = 8$
- $d(w_1) = 2$, $d(w_4) = 4$

Some notation and properties

- Consider closed walks of length k
  - Start and end at same node
  - Length k of a walk: #edges in it
- Node v "hits" walk w if $v \in w$.
- $n(v, G)$: #walks (of length k) containing node v
- $d(w)$: #distinct nodes in walk w
- Let $\mathcal{W}_k(G)$ denote the set of closed walks of length k
- Let $W_k(G) = |\mathcal{W}_k(G)|$
- Let $c(v)$ denote the cost of picking node v



- Node v_1 hits the walks $w_1 = (v_1, v_2, v_1, v_2, v_1)$, $w_2 = (v_1, v_3, v_1, v_3, v_1)$, $w_3 = (v_1, v_2, v_3, v_4, v_1)$, $w_4 = (v_1, v_4, v_3, v_2, v_1)$, and four other walks
- $n(v_1, G) = 8$
- $d(w_1) = 2$, $d(w_4) = 4$

Algorithm GreedyWalk

Main idea

Pick the smallest set S of nodes which hit many walks, for $k = \theta(\log n)$ (chosen to be an even number).

Algorithm GreedyWalk

Main idea

Pick the smallest set S of nodes which hit $\underline{\text{at least } W_k(G) - nT^k}$ walks, for $k = \theta(\log n)$ (chosen to be an even number).

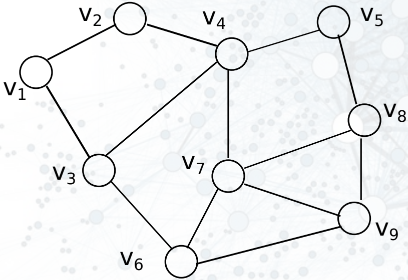
Algorithm GreedyWalk

Main idea

Pick the smallest set S of nodes which hit $\underline{\text{at least } W_k(G) - nT^k}$ walks, for $k = \theta(\log n)$ (chosen to be an even number).

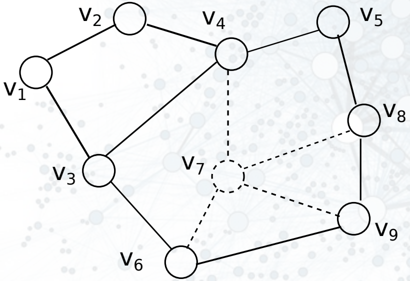
- Initialize $S \leftarrow \varnothing$
- Repeat while $W_k(G[V - S]) \geq nT^k$:
  - Pick the $v \in V - S$ that maximizes $\frac{n(v, G[V-S])}{c(v)}$
  - $S \leftarrow S \cup \{v\}$

GreedyWalk example



Node	$n(v, G)$
v_1	9
v_2	10
v_3	17
v_4	24
v_5	11
v_6	17
v_7	27
v_8	17
v_9	15

GreedyWalk example



Node	$n(v, G)$
v_1	9
v_2	9
v_3	12
v_4	13
v_5	7
v_6	7
v_7	–
v_8	6
v_9	6

Analysis

Lemma

We have $\lambda_1(G[V-S]) \leq (1+\epsilon)T$, *and* $c(S) = O(\frac{1}{\epsilon}\log^2 n \cdot c(V_{opt}(T)))$ *for any* $\epsilon \in (0,1)$.

Main steps in the proof:

(A) Bound spectral radius of residual graph, i.e., $\lambda_1(G[V-S])$

(B) Bound $c(S)$

Proof: bounding spectral radius of residual graph (I)

- Let $G' = G[V - S]$
- By construction: $W_k(G') \leq nT^k$
- Let A denote the adjacency matrix of G'

Proof: bounding spectral radius of residual graph (I)

- Let $G' = G[V - S]$
- By construction: $W_k(G') \leq nT^k$
- Let A denote the adjacency matrix of G'
- Let the eigenvalues of A be $\lambda_1, \lambda_2, \ldots$

Proof: bounding spectral radius of residual graph (I)

- Let $G' = G[V - S]$
- By construction: $W_k(G') \leq nT^k$
- Let A denote the adjacency matrix of G'
- Let the eigenvalues of A be $\lambda_1, \lambda_2, \ldots$
- Eigenvalues of A^k are $\lambda_1^k, \lambda_2^k, \ldots$

Proof: bounding spectral radius of residual graph (I)

- Let $G' = G[V - S]$
- By construction: $W_k(G') \leq nT^k$
- Let A denote the adjacency matrix of G'
- Let the eigenvalues of A be $\lambda_1, \lambda_2, \ldots$
- Eigenvalues of A^k are $\lambda_1^k, \lambda_2^k, \ldots$
- Sum of eigenvalues $=$ trace

Proof: bounding spectral radius of residual graph (I)

- Let $G' = G[V - S]$
- By construction: $W_k(G') \leq nT^k$
- Let A denote the adjacency matrix of G'
- Let the eigenvalues of A be $\lambda_1, \lambda_2, \ldots$
- Eigenvalues of A^k are $\lambda_1^k, \lambda_2^k, \ldots$
- Sum of eigenvalues $=$ trace
- $\Rightarrow \ \sum_i \lambda_i^k = \sum_i A_{ii}^k$

Proof: bounding spectral radius of residual graph (II)

- A^k_{ii}:

Proof: bounding spectral radius of residual graph (II)

- A^k_{ii}:   number of closed walks of length k containing node i

Proof: bounding spectral radius of residual graph (II)

- A_{ii}^k: number of closed walks of length k containing node i
- $\sum_i A_{ii}^k$

Proof: bounding spectral radius of residual graph (II)

- A_{ii}^k: number of closed walks of length k containing node i
- $\sum_i A_{ii}^k$ (over)-counts all walks of length k
  - Walk w gets counted $d(w)$ times
  - Therefore, $\sum_i A_{ii}^k = \sum_{w \in \mathcal{W}_k} d(w)$

Proof: bounding spectral radius of residual graph (III)

Putting everything together

$$\sum_i \lambda_i^k = \sum_i A_{ii}^k = \sum_{w \in \mathcal{W}(G')} d(w) \leq k W_k(G')$$

$$\Rightarrow \sum_i \lambda_i^k \leq k W_k(G') \leq k n T^k$$

$$\lambda_i^k \geq 0 \text{ since } k \text{ is even}$$

$$\Rightarrow \lambda_1^k \leq k n T^k$$

$$\Rightarrow \lambda_1 \leq (kn)^{1/k} T$$

$$\Rightarrow \lambda_1 \leq (1+\epsilon) T \text{ for } k \geqslant \frac{2}{\epsilon} \log n.$$

Proof: bounding $c(S)$ (main idea)

How do we relate $c(S)$ and $c(V_{opt})$?

S: greedily hits as many walks as possible

V_{opt}: reduces λ_1 to below T by removing minimum cost node set

Proof: bounding $c(S)$ (II)

Consider the following instance of hitting set:

- Ground set: V, i.e., all nodes
- Collection of sets $\mathcal{W}_k$, where each closed walk w is a set in $\mathcal{W}_k$
- Goal: choose a subset of V that "hits" at least $L = W_k(G) - nT^k$ walks (subsets in $\mathcal{W}_k$)

Let V_{hitopt} be an optimal solution for this hitting set problem

Proof: bounding $c(S)$ (main idea)

How do we relate $c(S)$ and $c(V_{opt})$?

S: greedily hits as many walks as possible

V_{opt}: reduces λ_1 to below T by removing minimum cost node set

Compare with the cost of set V_{hitopt}.
We will show that
$c(V_{hitopt}) \leq c(V_{opt})$, and $c(S) = O(\frac{1}{\epsilon} \log^2 n \cdot c(V_{hitopt}))$

Proof (B): bounding $c(S)$ (I)

Relating $c(V_{hitopt})$ with $c(V_{opt})$

- Let $\hat{A}$ denote the adjacency matrix corresponding to $G[V - V_{opt}]$, and let the eigenvalues of $\hat{A}$ be $\hat{\lambda}_1 \geq \hat{\lambda}_2 \ldots$
- By definition of V_{opt}: $\hat{\lambda}_1 \leq T$

Proof (B): bounding $c(S)$ (I)

Relating $c(V_{hitopt})$ with $c(V_{opt})$

- Let $\hat{A}$ denote the adjacency matrix corresponding to $G[V - V_{opt}]$, and let the eigenvalues of $\hat{A}$ be $\hat{\lambda}_1 \geq \hat{\lambda}_2 \ldots$
- By definition of V_{opt}: $\hat{\lambda}_1 \leq T$

$$W_k(G[V - V_{opt}]) \leq \sum_i \hat{A}^k_{ii}$$

Proof (B): bounding $c(S)$ (I)

Relating $c(V_{hitopt})$ with $c(V_{opt})$

- Let $\hat{A}$ denote the adjacency matrix corresponding to $G[V - V_{opt}]$, and let the eigenvalues of $\hat{A}$ be $\hat{\lambda}_1 \geq \hat{\lambda}_2 \ldots$
- By definition of V_{opt}: $\hat{\lambda}_1 \leq T$

$$W_k(G[V - V_{opt}]) \leq \sum_i \hat{A}^k_{ii}$$
$$= \sum_{i=1}^{n} \hat{\lambda}^k_i$$

Proof (B): bounding $c(S)$ (I)

Relating $c(V_{hitopt})$ with $c(V_{opt})$

- Let $\hat{A}$ denote the adjacency matrix corresponding to $G[V - V_{opt}]$, and let the eigenvalues of $\hat{A}$ be $\hat{\lambda}_1 \geq \hat{\lambda}_2 \ldots$
- By definition of V_{opt}: $\hat{\lambda}_1 \leq T$

$$
\begin{aligned}
W_k(G[V - V_{opt}]) &\leq \sum_i \hat{A}_{ii}^k \\
&= \sum_{i=1}^n \hat{\lambda}_i^k \\
&\leq n\hat{\lambda}_1^k \text{ (Perron-Frobenius theorem)}
\end{aligned}
$$

Proof (B): bounding $c(S)$ (I)

Relating $c(V_{hitopt})$ with $c(V_{opt})$

- Let $\hat{A}$ denote the adjacency matrix corresponding to $G[V - V_{opt}]$, and let the eigenvalues of $\hat{A}$ be $\hat{\lambda}_1 \geq \hat{\lambda}_2 \ldots$
- By definition of V_{opt}: $\hat{\lambda}_1 \leq T$

$$
\begin{aligned}
W_k(G[V - V_{opt}]) &\leq \sum_i \hat{A}^k_{ii} \\
&= \sum_{i=1}^{n} \hat{\lambda}_i^k \\
&\leq n\hat{\lambda}_1^k \text{ (Perron-Frobenius theorem)} \\
&\leq nT^k
\end{aligned}
$$

Proof (B): bounding $c(S)$ (I)

Relating $c(V_{hitopt})$ with $c(V_{opt})$

- Let $\hat{A}$ denote the adjacency matrix corresponding to $G[V - V_{opt}]$, and let the eigenvalues of $\hat{A}$ be $\hat{\lambda}_1 \geq \hat{\lambda}_2 \dots$
- By definition of V_{opt}: $\hat{\lambda}_1 \leq T$

$$
\begin{aligned}
W_k(G[V - V_{opt}]) &\leq \sum_i \hat{A}^k_{ii} \\
&= \sum_{i=1}^{n} \hat{\lambda}^k_i \\
&\leq n\hat{\lambda}^k_1 \text{ (Perron-Frobenius theorem)}
\end{aligned}
$$

V_{opt} is a feasible solution to this hitting set instance
$\Rightarrow c(V_{\text{hitopt}}) \leq c(V_{opt})$

Proof: bounding $c(S)$ (II)

Relating $c(S)$ and $c(V_{hitopt})$

- V_{hitopt}: an optimal solution for this hitting set problem
- In contrast: S is a greedy solution
- By standard greedy analysis, we can show $c(S) = O(c(V_{\text{hitopt}}) \log(\#\text{sets})) = O(c(V_{\text{hitopt}}) \log |\mathcal{W}_k|)$
- $|\mathcal{W}_k| = W_k(G) \leqslant n\Delta^k$, where Δ is the maximum degree (note: $\Delta \leq n$)
- Therefore, $c(S) = O(c(V_{hitopt}) \cdot (\log n \log \Delta)/\epsilon) = O(\frac{1}{\epsilon} \log^2 n \cdot c(V_{hitopt}))$

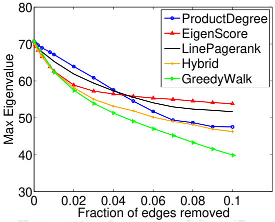
Empirical evaluation (edge version)

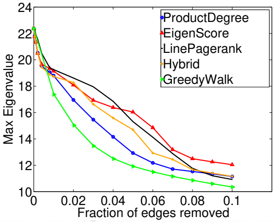
Baselines: prior heuristics

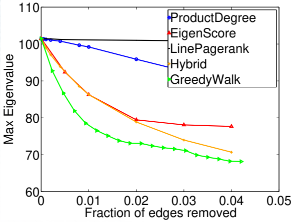
- Pick edges $e = (i, j)$ in decreasing order of *eigenscore*$(i, j) = x^1(i) \cdot x^1(j)$ [Tong et al., 2012], [Van Mieghem et al., 2011]
- Pick edges $e = (i, j)$ in decreasing order of *degscore*$(i, j) = d(i)d(j)$ [Van Mieghem et al., 2011]
- Hybrid rule: pick edge from either order whose removal causes the largest reduction in λ_1

Empirical evaluation

- Significantly better than all prior heuristics for all kinds of networks
- Performance improves with k







Autonomous system network

P2P Gnutella network

Brightkite network

Better approximation factor?

- Partial coverage problem: primal-dual algorithm of [Gandhi et al., 2004] for selecting a minimum cost collection of sets that covers at least k elements, with $O(f)$-approximation, where f is the maximum number of sets containing any element
- Our set system:
  - Sets $\equiv$ nodes, elements $\equiv$ walks in $\mathcal{W}_k$
  - $f = O(\log n)$, since walks have length $k = O(\log n)$
- Set system of size $n^{O(\log n)}$, so cannot apply primal-dual algorithm of [Gandhi et al., 2004] directly
  - Can do updates implicitly and get polynomial time $O(\log n)$-approximation
  - Results in $c(S) = O(c(V_{opt}(T)) \log n)$, $\lambda_1(G[V - S]) \leq (1+\epsilon)T$
- Constant factor approximation by semidefinite programming based rounding.