The 22nd IEEE International Conference on Mobile Ad – Hoc and Smart Systems (MASS 2025)

Effective Delayed Patching for Transient Malware Control on Networks

Minh Phu Vuong[1], Chul-Ho Lee[1], and Do Young Eun[2]

[1]Texas State University

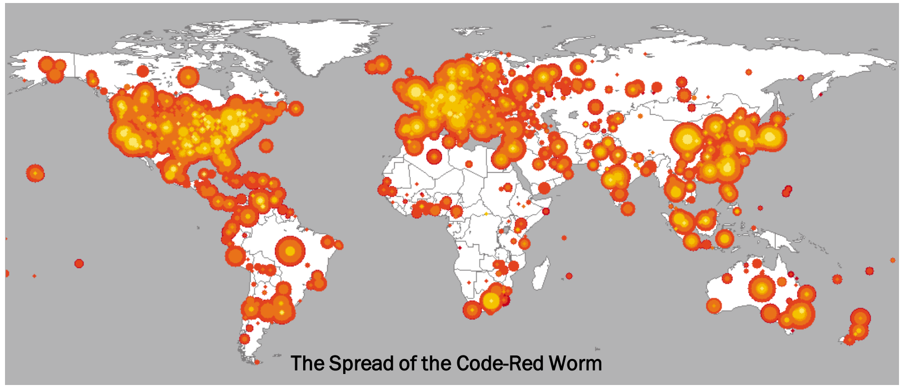
[2]North Carolina State University

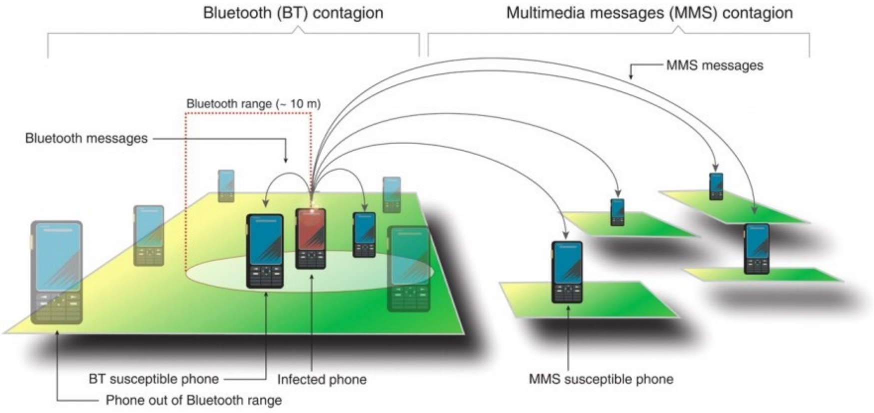
TEXAS STATE UNIVERSITY

NC STATE

Introduction

- Epidemic models are important and useful.
  - For modeling the malware propagation over a network.



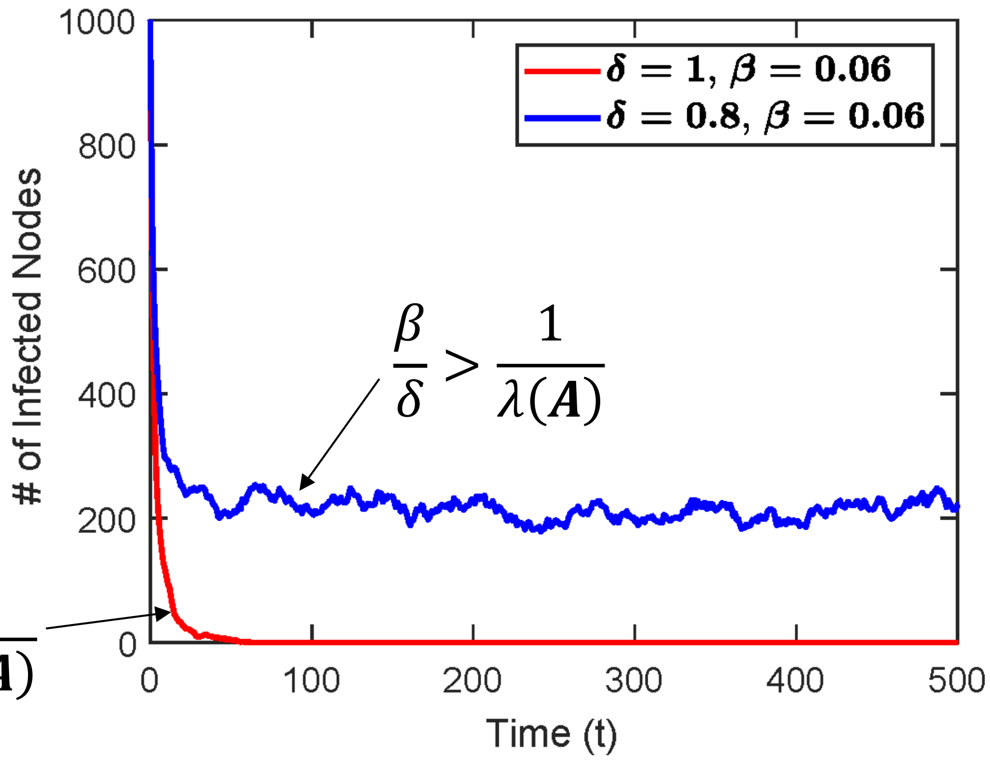


  - For analyzing the spread of an infectious disease and its control.

Motivation

- Most studies have been concerned about the *persistence and extinction* of the epidemics in their *steady state*.
  - ➢ Under what conditions an epidemic dies out quickly.

$$\frac{\beta}{\delta} < \frac{1}{\lambda(\boldsymbol{A})}$$



SIS simulation on a Erdos-Renyi graph with 2000 nodes (1000 nodes initially infected)

- β: infection rate
- δ: recovery rate
- $\lambda(\boldsymbol{A})$: spectral radius of adjacency matrix $\boldsymbol{A}$

Motivation

- Our recent work studied the *transient dynamics of SI epidemic spreading*.
  - Non-negligible amount of time for a patch or vaccine to become available after the outbreak of an epidemic.
  - We developed a tighter upper bound which allows us to predict the likelihood of each node being infected after any time t.

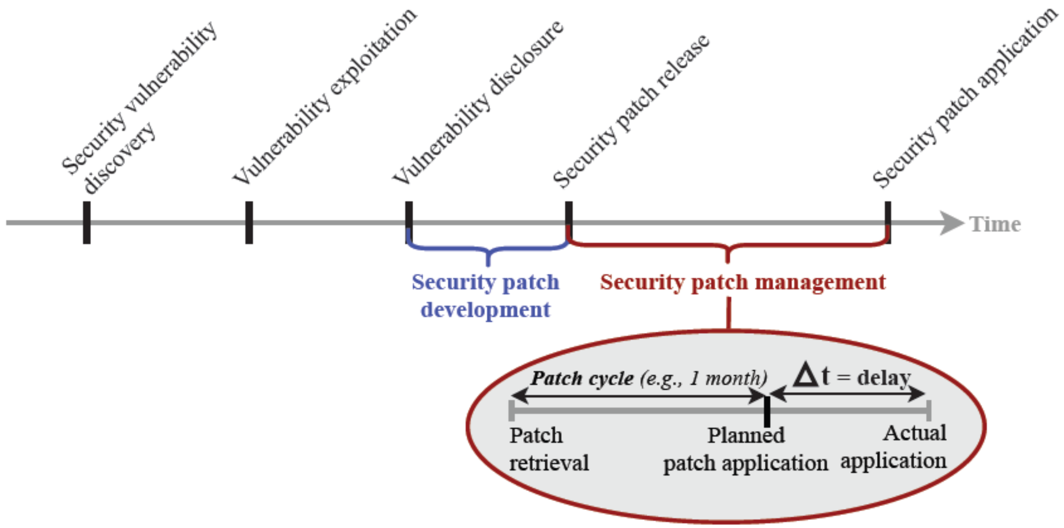
$$\mathbf{x}(t) \preceq \hat{\mathbf{x}}(t) = f\big(\hat{\mathbf{y}}(t)\big), \qquad f(y) = 1 - e^{-y}$$

$$\hat{\mathbf{y}}(t) = -\log\big(1 - \mathbf{x}(0)\big) + \sum_{k=0}^{\infty} \frac{(\beta t)^{k+1}}{(k+1)!} \left[\mathbf{A}\,\mathrm{diag}\big(1 - \mathbf{x}(0)\big)\right]^{k} \mathbf{A}\,\mathbf{x}(0).$$

- C.-H. Lee, S. Tenneti, and D. Y. Eun, “Transient dynamics of epidemic spreading and its mitigation on large networks,” in ACM MobiHoc, 2019.

Motivation

- Software patching process is multi-step and complex.

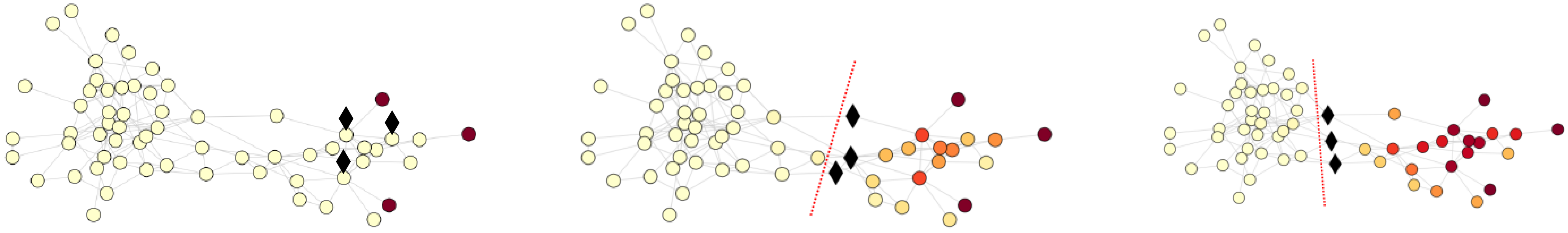


- Possible failure in each round of software patching process leads to non-negligible delay.

- N. Dissanayake, M. Zahedi, A. Jayatilaka, and A. Babar, “Why, how and where of delays in software security patch management: An empirical investigation in the healthcare sector,” in ACM CSCW, 2022.

Problem Formulation

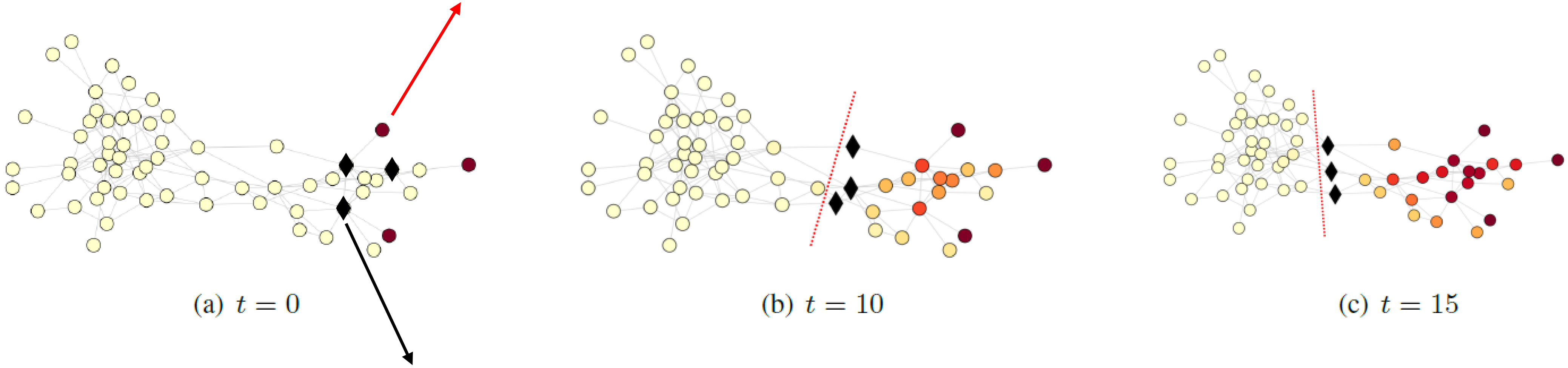
- Objective: Maximize the expected # of nodes that are saved by vaccinating on a graph **G** in the presence of patching delay $\boldsymbol{T}$ and under a limited patching budget $\boldsymbol{b}$.



(a) $t = 0$ (b) $t = 10$ (c) $t = 15$

Problem Formulation

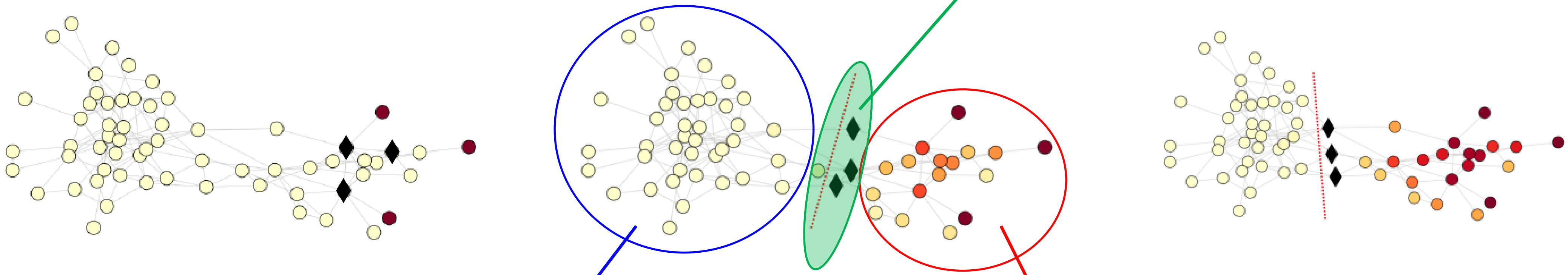
- Objective: Maximize the expected # of nodes that are saved by vaccinating on a graph **G** in the presence of patching delay $\boldsymbol{T}$ and under a limited patching budget $\boldsymbol{b}$.



(a) $t = 0$ (b) $t = 10$ (c) $t = 15$

Problem Formulation

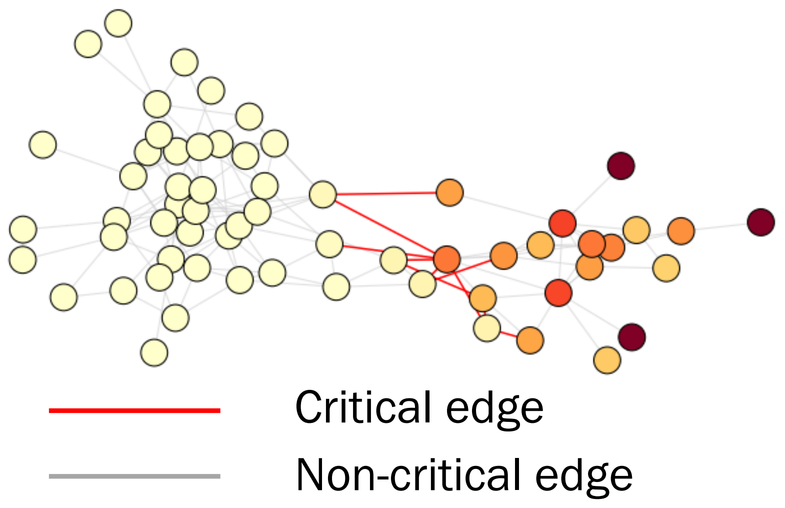
- Objective: Maximize the expected # of nodes that are saved by vaccinating on a graph **G** in the presence of patching delay $\boldsymbol{T}$ and under a limited patching budget $\boldsymbol{b}$.



(a) $t = 0$ (b) $t = 10$ (c) $t = 15$

Problem Formulation

- To identify the boundary, we introduce a notion of ‘critical edges’ - the edges that connect a healthy node to an infected node.
- The edge weight is the probability of an edge being critical at the patching delay time T.

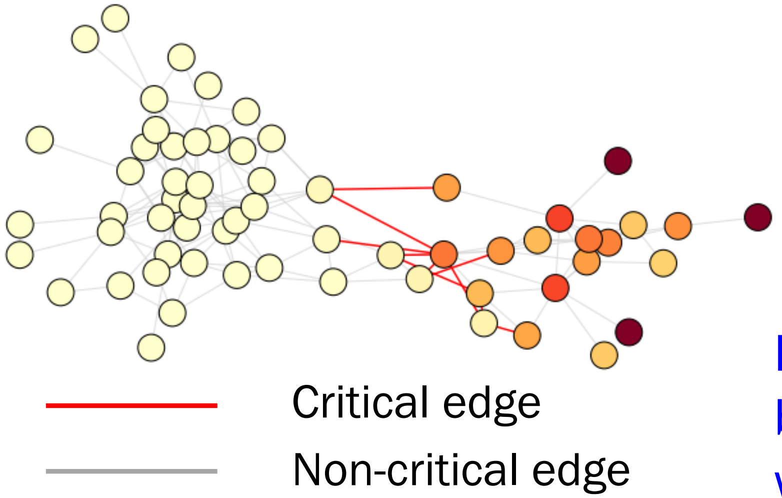


$$w_{i,j}(T) = a_{ij}\left[\hat{x}_i(T)(1-\hat{x}_j(T)) + (1-\hat{x}_i(T))\hat{x}_j(T)\right].$$

* Please refer to our paper for more details.

Problem Formulation

- To identify the boundary, we introduce a notion of 'critical edges' - the edges that connect a healthy node to an infected node.
- The edge weight is the probability of an edge being critical at the patching delay time T.



$$\boxed{w_{i,j}(T)} = a_{ij}\left[\boxed{\hat{x}_i(T)}(1-\boxed{\hat{x}_j(T)}) + (1-\hat{x}_i(T))\hat{x}_j(T)\right].$$

High along infection boundary, low for edges within susceptible or infected regions

Derived from the upper bound $f(\hat{y}(T))$

* Please refer to our paper for more details.

Problem Formulation

- Formulate the problem as Normalized Cut (NCut)

$$\min_{U \subset N} \mathrm{NCut}(U) = \min_{U \subset N} \left(\frac{\mathrm{Cut}(U, U^c)}{\mathrm{vol}(U)} + \frac{\mathrm{Cut}(U, U^c)}{\mathrm{vol}(U^c)} \right)$$

where $\mathrm{Cut}(U, U^c) \triangleq \sum_{i \in U} \sum_{j \in U^c} w_{ij}$

* Please refer to our paper for more details.

Problem Formulation

- Formulate the problem as Normalized Cut (NCut)

$$\min_{U\subset N} \mathrm{NCut}(U) = \min_{U\subset N} \left(\frac{\mathrm{Cut}(U,U^c)}{\mathrm{vol}(U)} + \frac{\mathrm{Cut}(U,U^c)}{\mathrm{vol}(U^c)} \right)$$

where $\mathrm{Cut}(U,U^c) \triangleq \sum_{i\in U}\sum_{j\in U^c} \boxed{w_{ij}} \rightarrow$ We flip the edge weights so NCut partitions along the minimum weights.

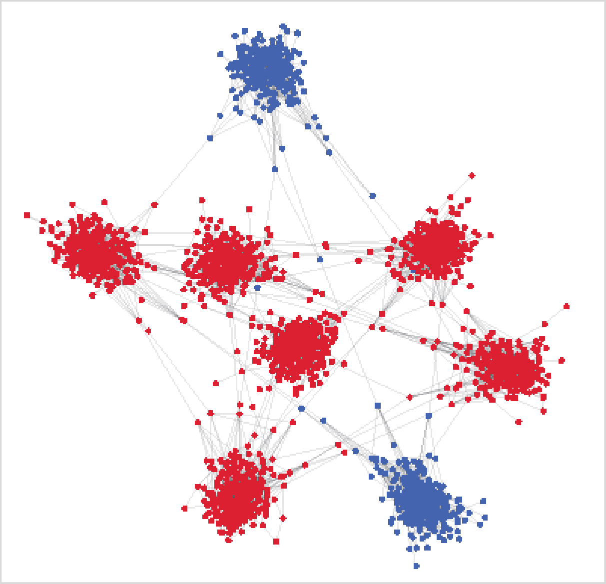
- NCut relaxed form: $\min_{\boldsymbol{v}\in\mathbb{R}^n} \boldsymbol{v}^\top \overline{\mathbf{L}} \boldsymbol{v}$

$$\text{subject to } \|\boldsymbol{v}\|^2 = \mathrm{vol}(N) \text{ and } \boldsymbol{v}^\top \mathbf{D}^{1/2}\mathbf{1} = 0.$$

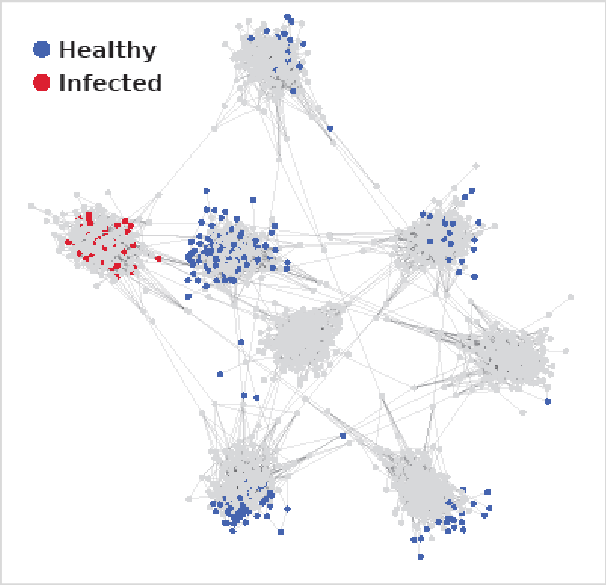
* Please refer to our paper for more details.

Constrained NCut Problem

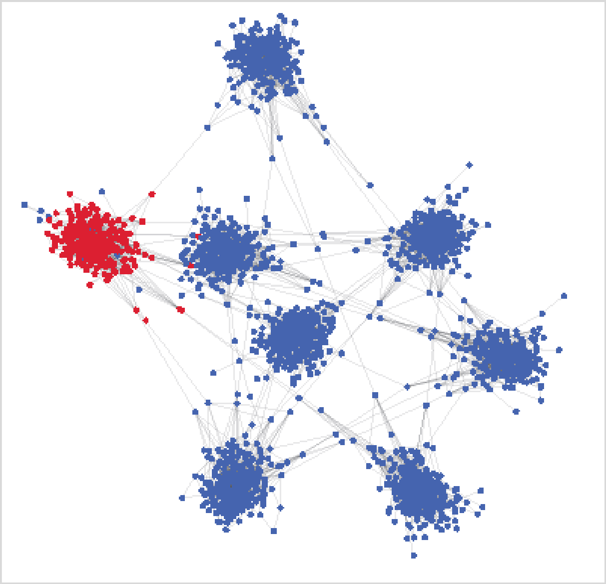
Solution of vanilla NCut



Initial state



Solution of constrained NCut



- ➢ Ignore epidemic dynamics
- ➢ Fail to isolate the infected region

- ➢ Utilize epidemic dynamics as **constraints** for better solution
- ➢ Successfully separate infected region

* Please refer to our paper for more details.

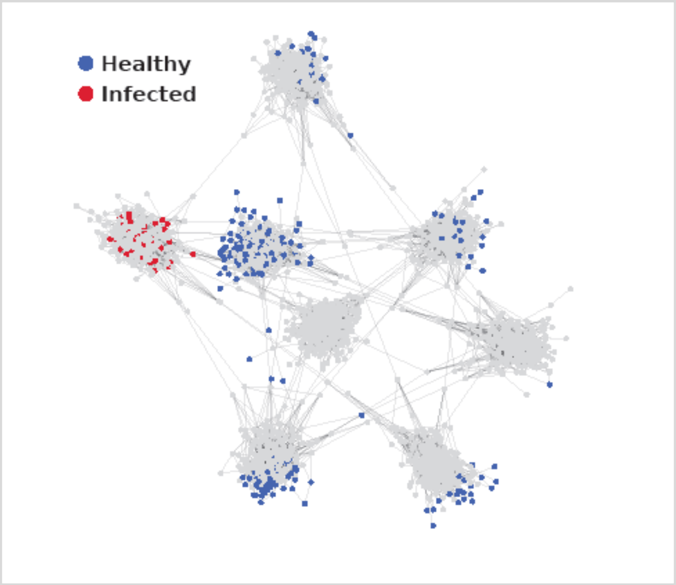
The NCut Problem Pitfall

$$\min_{\boldsymbol{v}\in\mathbb{R}^n} \boldsymbol{v}^\top \overline{\mathbf{L}}\boldsymbol{v}$$

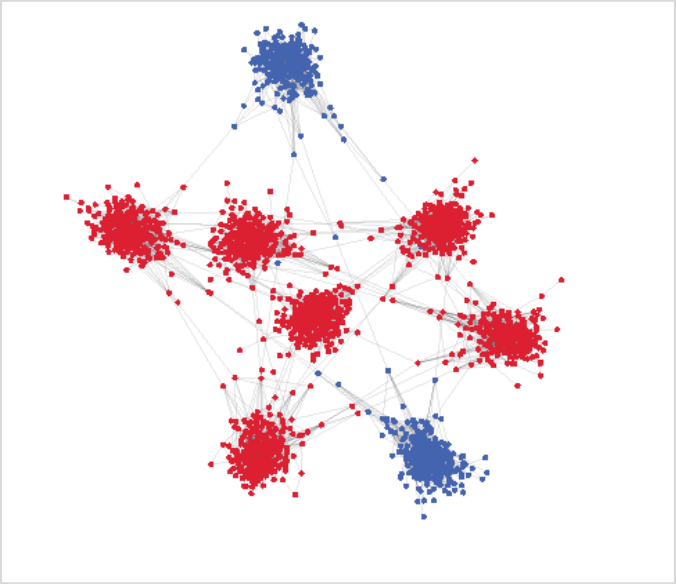
$$\text{subject to } \|\boldsymbol{v}\|^2 = \mathrm{vol}(N) \text{ and } \boldsymbol{v}^\top \mathbf{D}^{1/2}\mathbf{1} = 0.$$

- Ignore epidemic dynamics
- Fail to isolate the infected region

Initial state



Solution of vanilla NCut



* Please refer to our paper for more details.

Constrained NCut Problem

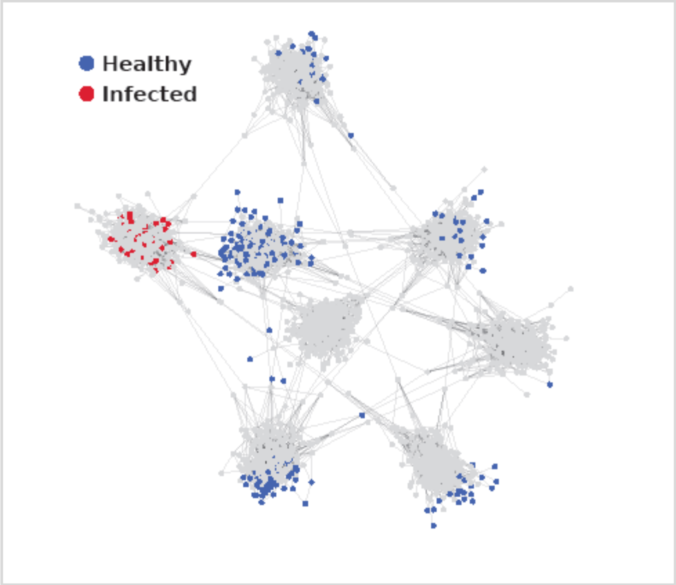
$$\min_{\boldsymbol{v}\in\mathbb{R}^n} \boldsymbol{v}^\top \overline{\mathbf{L}} \boldsymbol{v}$$

$$\text{subject to } \|\boldsymbol{v}\|^2 = \mathrm{vol}(N) \text{ and } \mathbf{B}\boldsymbol{v} = \mathbf{c}.$$

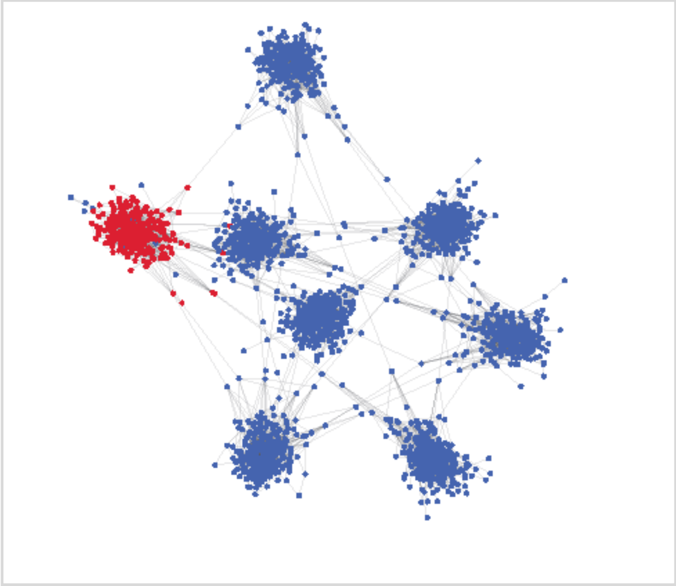
- Utilize epidemic dynamics as linear constraints
- Steer the solution toward a more **meaningful boundary** of critical edges
- Successfully separate infected region

* Please refer to our paper for more details.

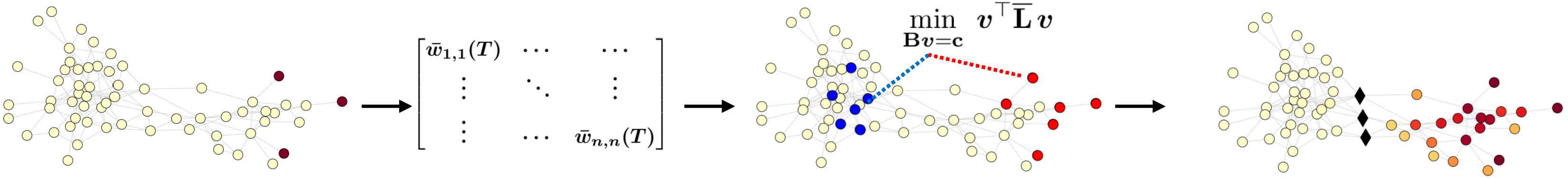
Initial state



Solution of constrained NCut

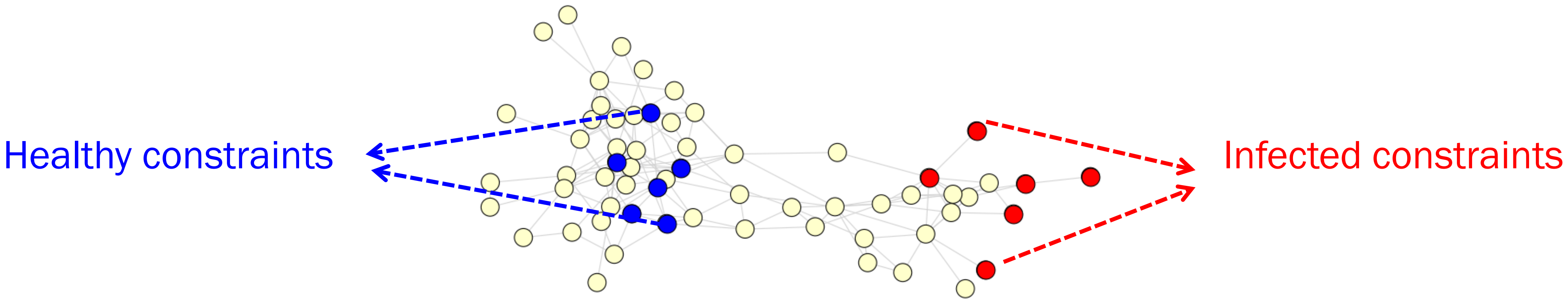


Framework



❶ Input graph ❷ Weight matrix ❸ NCut with linear constraints ❹ Graph partitioning

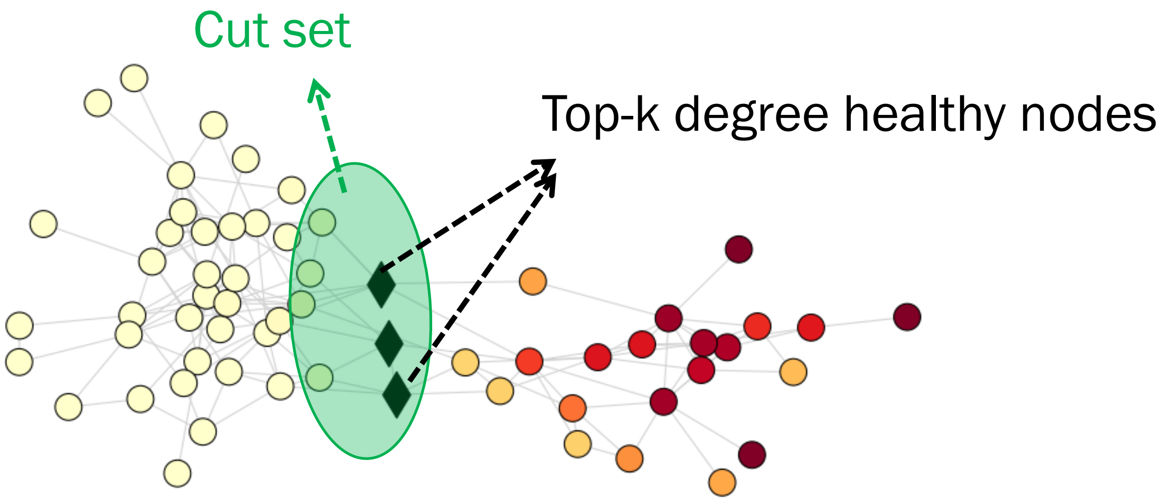
Choosing constrained nodes



- Infected constraints: Initially infected nodes and their one-hop neighbors.
- Healthy constraints: Top-K nodes with the longest shortest-path from the source of infection.

* Please refer to our paper for more details.

Node Selection for Patching under Budget Constraint



- Repeatedly patch the highest degree healthy node on the boundary until the cut set or the budget is empty.
- If the budget is still available, patch unselected one-hop neighbors of the nodes just vaccinated (highest degree first).

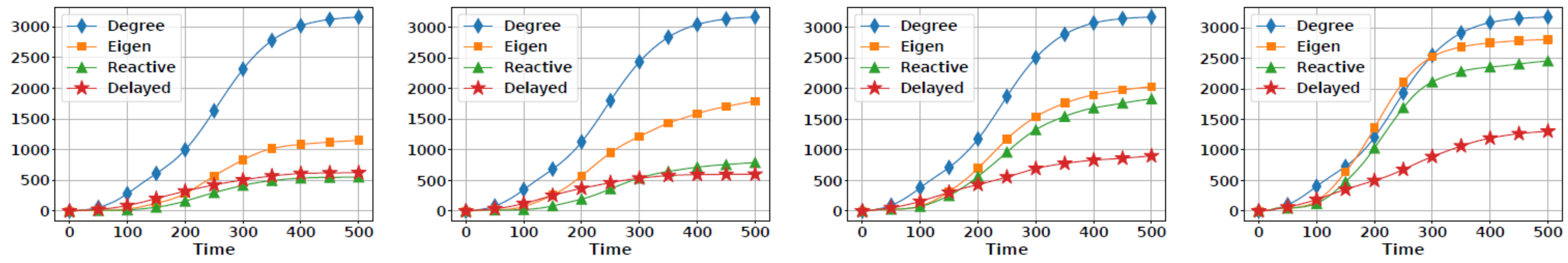
* Please refer to our paper for more details.

Simulation Setup

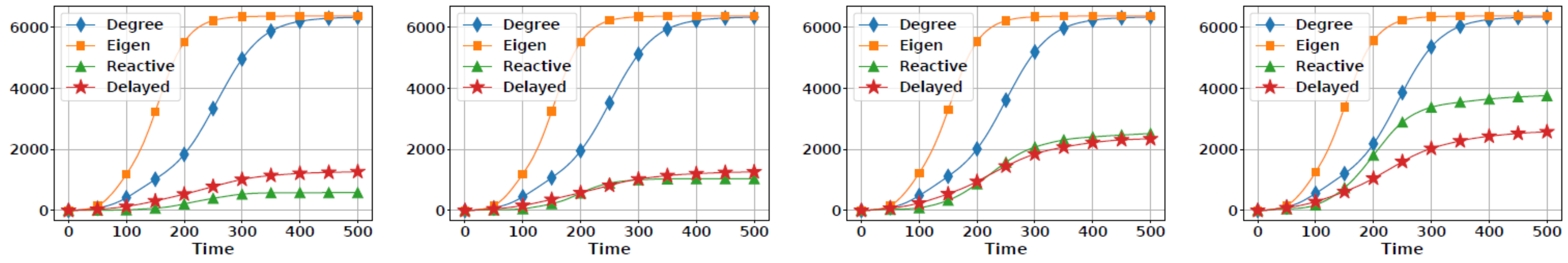
- Datasets
  - Synthetic graph: Stochastic Block Model (SBM) with k communities.
  - Real-world graph: Facebook network.
- Baseline vaccination policies
  - Degree policy: Vaccinate the top-k highest degree nodes.
  - Eigen policy: Vaccinate the top-k highest eigenvector centrality nodes.
  - Reactive policy: Vaccinate the top-k nodes with the highest predicted infection probability at delay T.

Simulation Results: Synthetic Graphs

- The expected number of infected nodes by each vaccination policy



(i) $T = 15, \ n = 4000, \ k = 5$ (j) $T = 20, \ n = 4000, \ k = 5$ (k) $T = 25, \ n = 4000, \ k = 5$ (l) $T = 30, \ n = 4000, \ k = 5$

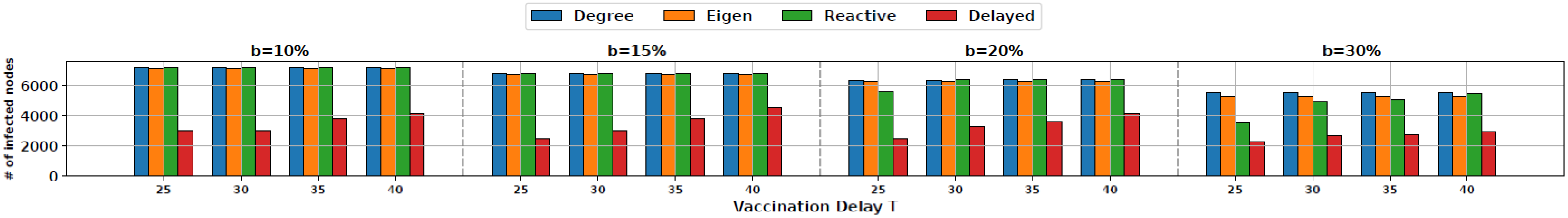


(m) $T = 15, \ n = 8000, \ k = 6$ (n) $T = 20, \ n = 8000, \ k = 6$ (o) $T = 25, \ n = 8000, \ k = 6$ (p) $T = 30, \ n = 8000, \ k = 6$

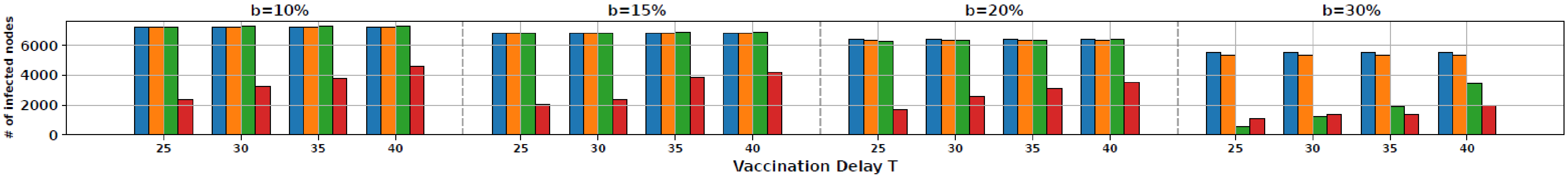
- Observation 1: As the patching delay (T) increases, our Delayed policy becomes significantly more effective.
- Observation 2: Improvements of the Delayed policy over the Reactive, Eigenvector, and Degree policies are up to 50%, 83.3%, and 83.3%.

Simulation Results: Synthetic Graphs

- Impact of the vaccination budget with different delayed time



(a) $k = 3$

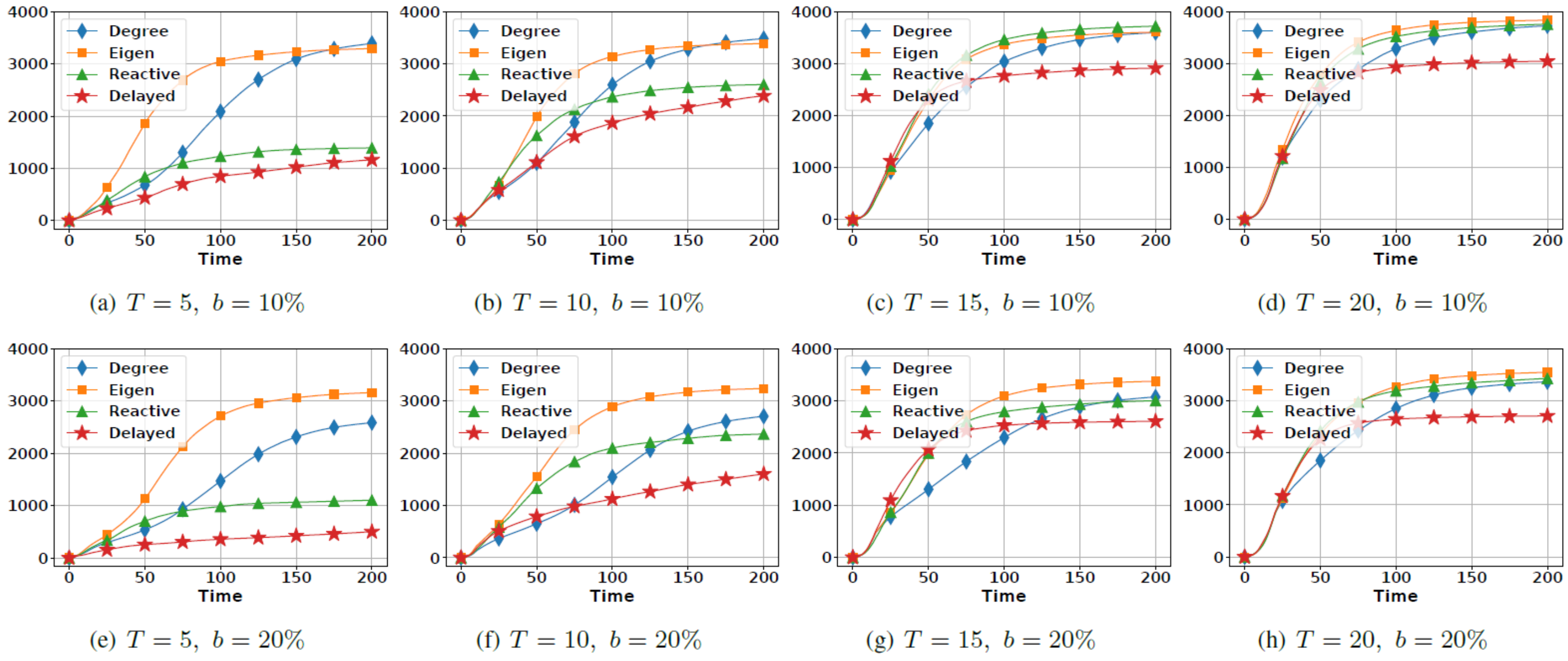


(b) $k = 4$

- Observation 1: The number of infected nodes increases as the delay time increases, while it decreases as the budget increases.
- Observation 2: Our delayed policy achieves the lowest number of infected nodes.

Simulation Results: Real-world graph

- The expected number of infected nodes with varying values of delayed time and vaccination budget.



(a) $T = 5,\ b = 10\%$ (b) $T = 10,\ b = 10\%$ (c) $T = 15,\ b = 10\%$ (d) $T = 20,\ b = 10\%$

(e) $T = 5,\ b = 20\%$ (f) $T = 10,\ b = 20\%$ (g) $T = 15,\ b = 20\%$ (h) $T = 20,\ b = 20\%$

➢ Observation: Our delayed policy remains effective under longer patching delay, while other policies fail as the population becomes almost infected.

Conclusion

- We introduce a novel mathematical framework for effective patching under limited resources and in the presence of patching delays.
- Our policy identifies a minimum-cut boundary to separate infected nodes from the healthy region and optimally select which nodes to patch.
- We demonstrate the superior performance over existing baselines through extensive experiments on synthetic and real-world networks.
- We provide a foundational step toward designing vaccination strategies for general networks under realistic delay and resource constraints.

Thank you!!

Questions & Answers