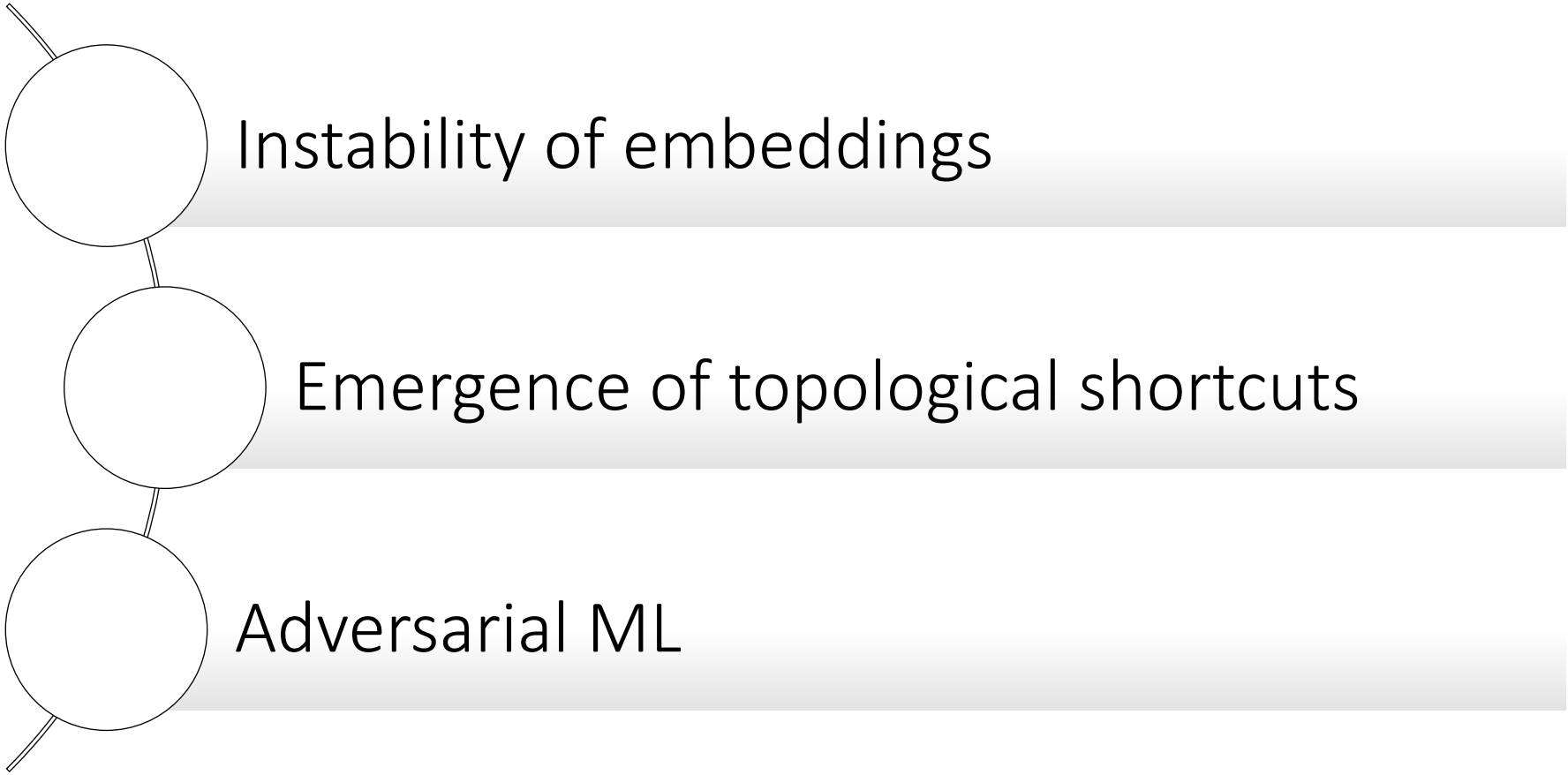
Northeastern University
Network Science Institute
The Institute for Experiential AI
Khoury College of Computer Sciences

SANTA FE INSTITUTE

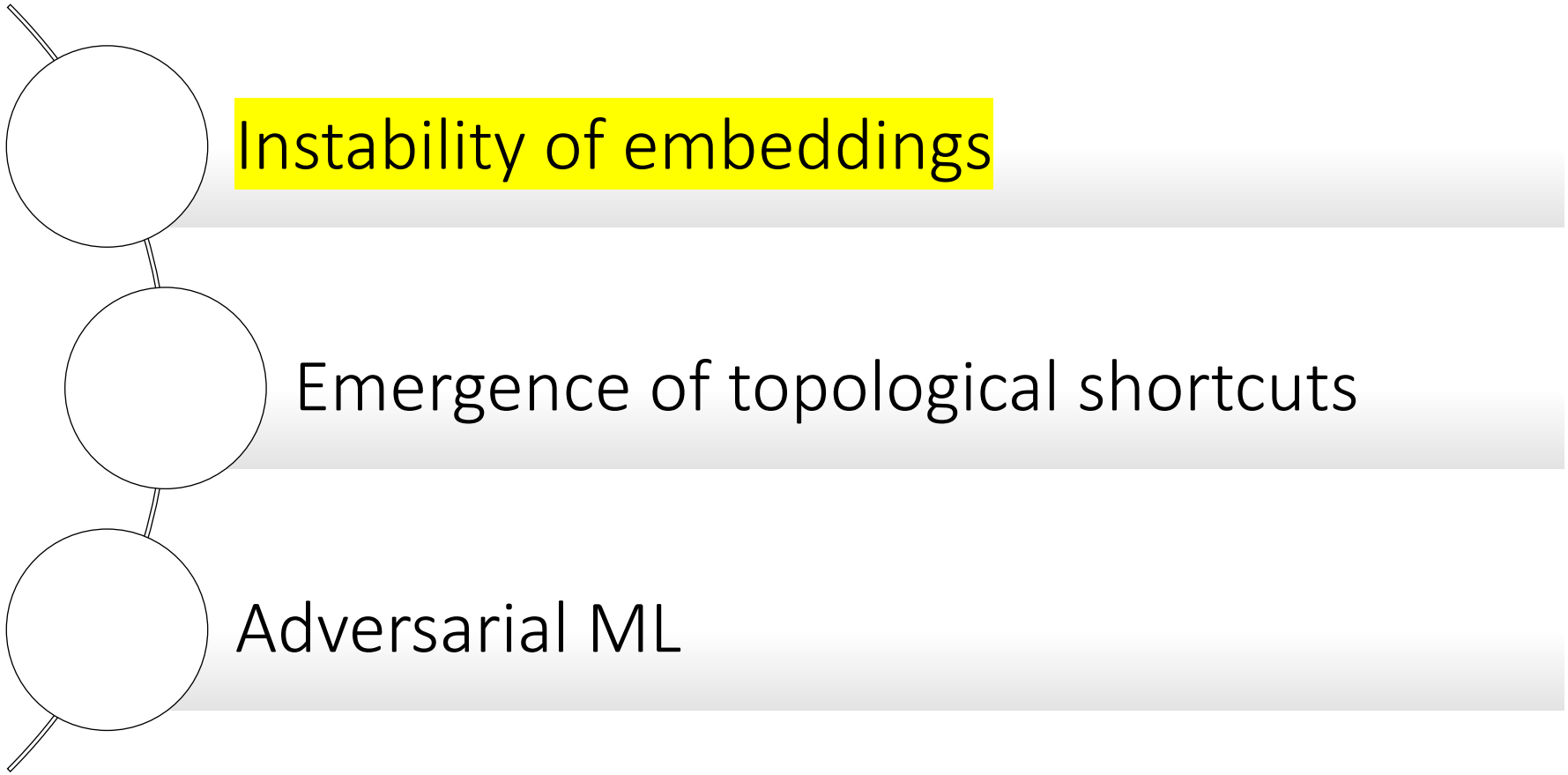
The Pitfalls of Using ML-based Optimization

Tina Eliassi-Rad
tina@eliassi.org

Outline


Instability of embeddings
Emergence of topological shortcuts
Adversarial ML

Outline

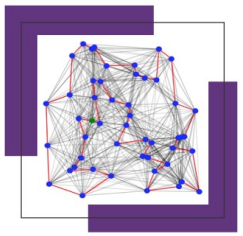

Instability of embeddings
Emergence of topological shortcuts
Adversarial ML

Many ML-based optimization methods use some form of embedding (or encoding) of data into vector spaces.

ipam institute for pure & applied mathematics

Deep Learning and Combinatorial Optimization

February 22 - 25, 2021



Scientific Overview

This workshop will apply deep learning methods to combinatorial optimization problems that typically emerge in finance and revenue management, transportation, manufacturing, supply chain, public policy, hardware design, computing and information technology. Given its high flexibility, approximate nature, and self-learning paradigm, deep learning is particularly attractive to address combinatorial optimization problems. Preliminary but promising advances have already emerged in the Traveling Salesman Problem, MaxCut, Minimum Vertex Cover, Knapsack, Quadratic Assignment Problem and Vehicle Routing Problems. Synergies between the two fields could also lead to the development of new algorithms, especially relevant for applied problems. The workshop will bring together experts in mathematics (optimization, graph theory, sparsity, combinatorics, statistics), combinatorial optimization (assignment problems, routing, planning, Bayesian search, scheduling), machine learning (deep learning, supervised, self-supervised and reinforcement learning) and specific applicative domains (e.g. finance, transportation, hardware design, computing and information technology) to establish the current state of these emerging techniques and discuss the next directions.

Organizers

Peter Battaglia (DeepMind), Xavier Bresson (NTU, Singapore), Stefanie Jegelka (MIT), Yann LeCun (NYU), Andrea Lodi (Polymtl), Stanley Osher (UCLA), Oriol Vinyals (DeepMind), and Max Welling (University of Amsterdam).

Speakers

Sanjeev Arora (Princeton), Xavier Bresson (NTU, Singapore), Joan Bruna (NYU), Laurent Charlin (HEC Montréal), Kyle Cranmer (NYU), Sanjeeb Dash (IBM), Santanu Dey (Georgia Tech), Bistra Dilkina (USC), Tina Eliassi-Rad (Northeastern University), Emma Frejinger (University of Montreal), Maxime Gasse (Polymtl), Stefano Gualandi (University of Pavia), Oktay Gunluk (Cornell), Joseph Huchette (Rice), Stefanie Jegelka (MIT), Ron Kimmel (Technion), Zico Kolter (CMU), Wouter Kool (University of Amsterdam), Azalia Mirhoseini (Google), Vinod Nair (DeepMind), Sebastian Pokutta (ZIB), Louis-Martin Rousseau (Polytechnique Montréal), Thiago Serra (Bucknell University), Le Song (Georgia Tech), Yunhao Tang (Columbia University), Petar Velickovic (DeepMind), and Ellen Vitercik (CMU).

Participation

Additional information about this workshop including links to register and to apply for funding, can be found on the webpage listed below. Encouraging the careers of women and minority mathematicians and scientists is an important component of IPAM's mission, and we welcome their applications.

NSF UCLA

For more information, visit the program webpage:
www.ipam.ucla.edu/dlc2021

https://bit.ly/3Zw3v3q

Morales et al. Appl Netw Sci (2021) 6:24
https://doi.org/10.1007/s41109-021-00365-8

RESEARCH Open Access

Selective network discovery via deep reinforcement learning on embedded spaces

Peter Morales[1*], Rajmonda Sulo Caceres[1] and Tina Eliassi-Rad[2]

*Correspondence:
pmorales@mit.edu
[1] MIT Lincoln Laboratory, Lexington, USA
Full list of author information is available at the end of the article

Abstract

Complex networks are often either too large for full exploration, partially accessible, or partially observed. Downstream learning tasks on these incomplete networks can produce low quality results. In addition, reducing the incompleteness of the network can be costly and nontrivial. As a result, network discovery algorithms optimized for specific downstream learning tasks given resource collection constraints are of great interest. In this paper, we formulate the task-specific network discovery problem as a sequential decision-making problem. Our downstream task is selective harvesting, the optimal collection of vertices with a particular attribute. We propose a framework, called network actor critic (NAC), which learns a policy and notion of future reward in an offline setting via a deep reinforcement learning algorithm. The NAC paradigm utilizes a task-specific network embedding to reduce the state space complexity. A detailed comparative analysis of popular network embeddings is presented with respect to their role in supporting offline planning. Furthermore, a quantitative study is presented on various synthetic and real benchmarks using NAC and several baselines. We show that offline models of reward and network discovery policies lead to significantly improved performance when compared to competitive online discovery algorithms. Finally, we outline learning regimes where planning is critical in addressing sparse and changing reward signals.

Keywords: Incomplete networks, Reinforcement learning, Network embedding

Introduction

Complex networks are critical to many applications such as those in the social, cyber, and bio domains. We commonly have access to partially observed data. The challenge is to discover enough of the complex network so that we can perform a learning task well. The network discovery step is especially critical in the case where the learning task has the characteristics of the "needle in a haystack" problem. If the discovery process is not carefully tuned, the noise introduced, almost always, overwhelms the signal. This presents an optimization problem: how should we grow an incomplete network to achieve a learning objective on the network, while at the same time minimize the cost of observing new data?

In this work, we view the network discovery problem from a decision-theoretic lens, where notions of utility and resource cost are naturally defined and jointly leveraged in

Springer Open

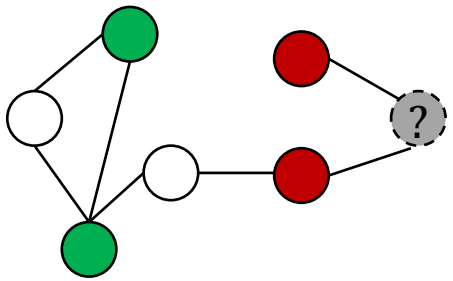
© The Author(s) 2021. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Background: Node embedding

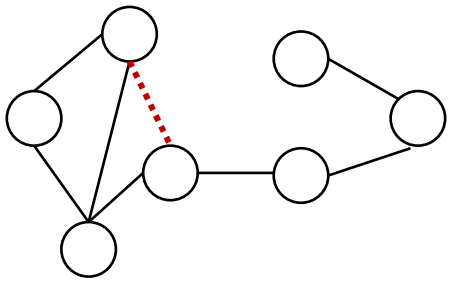
- For a graph $G = (V, E)$, represent every node in $\mathbb{R}^d$ where $d \ll |V|$

- Common uses of node embeddings

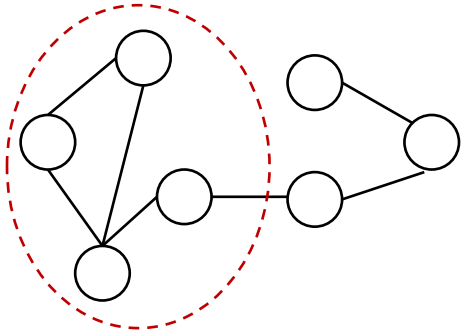
Node Classification



Link Prediction



Clustering



Question

How sensitive are node embeddings to graph perturbations?

David Liu

D. Liu, T. Eliassi-Rad. **STABLE: Identifying and Mitigating Instability in Embeddings of the Degenerate Core**. In *Proceedings of the 2023 SIAM International Conference on Data Mining* (SDM), April 2023.

Motivation

- Real-world graphs are incomplete and noisy.

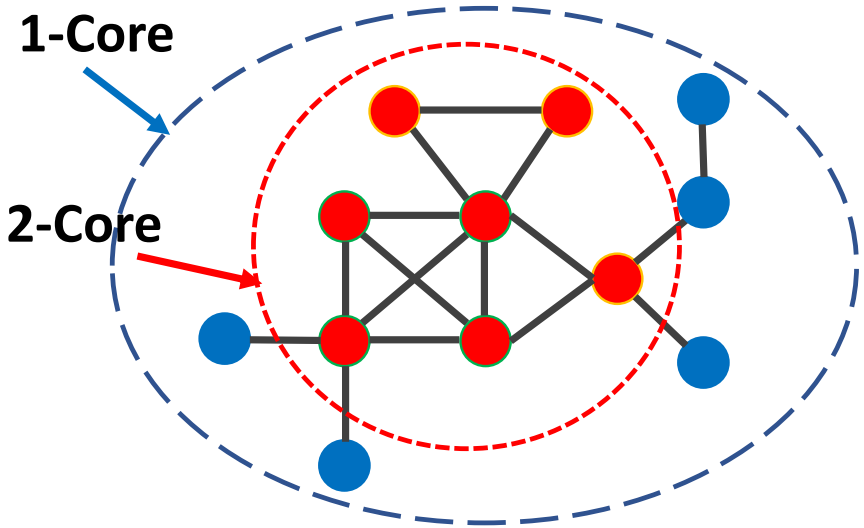
- A common approach to understanding stability is to measure changes to algorithmic output due to input perturbations.

Original question: How sensitive are node embeddings to graph perturbations?

Modified question: How does the embedding of the degenerate core change as nodes in the periphery are removed and the graph is re-embedded?

Background: *k*-core and the degenerate core

- *k*-core: maximal subgraph of nodes with degree ≥ *k*
- Degenerate core: the *k*-core with max *k*



Why should you care about the degenerate core and its embedding stability?

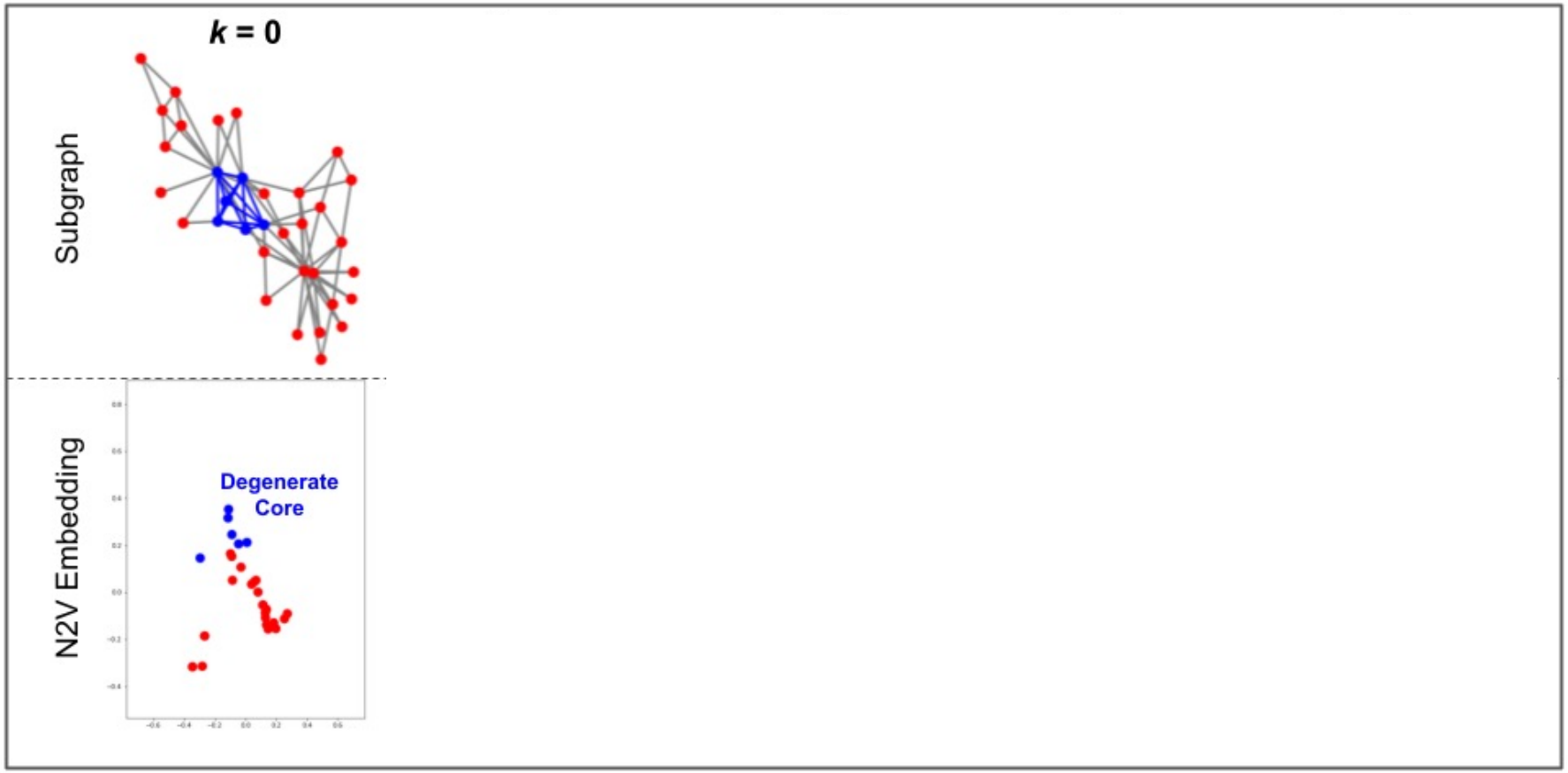
- Shin et al. [IEEE ICDM 2016] show that while degenerate cores are dense, they are generally not cliques. Instead, they contain community structure.

- Liu et al. [Scientific Reports 2015] and Laishram et al. [SDM 2020] show that in marketing applications, the removal of a node in a dense subgraph can trigger a cascade of node removals.

- Barbera et al [PLOS ONE 2015] show that in online activism, core nodes rely on periphery nodes to amplify messages that originated from core nodes.

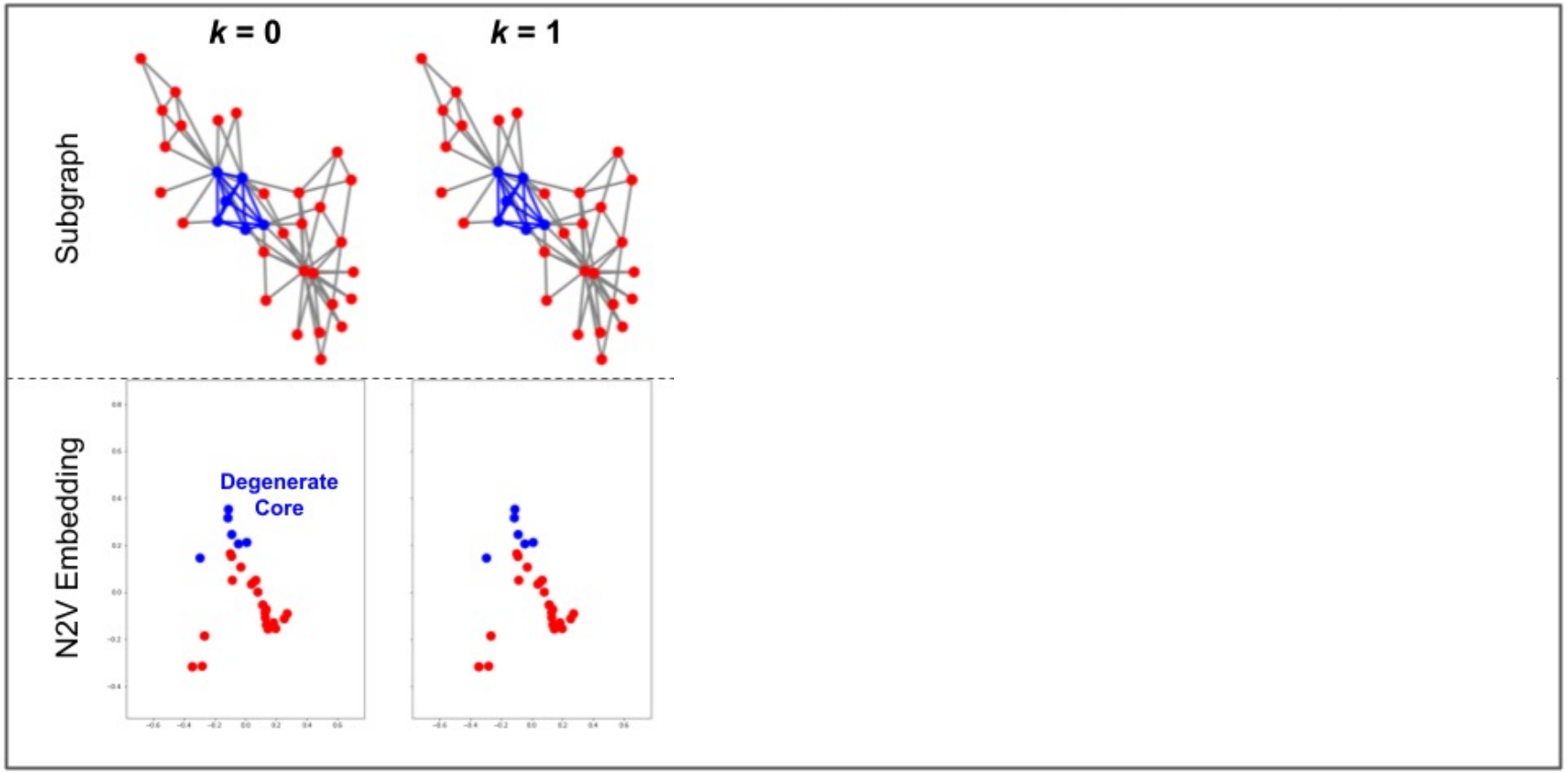
Method for evaluating stability

Shave and Re-embed



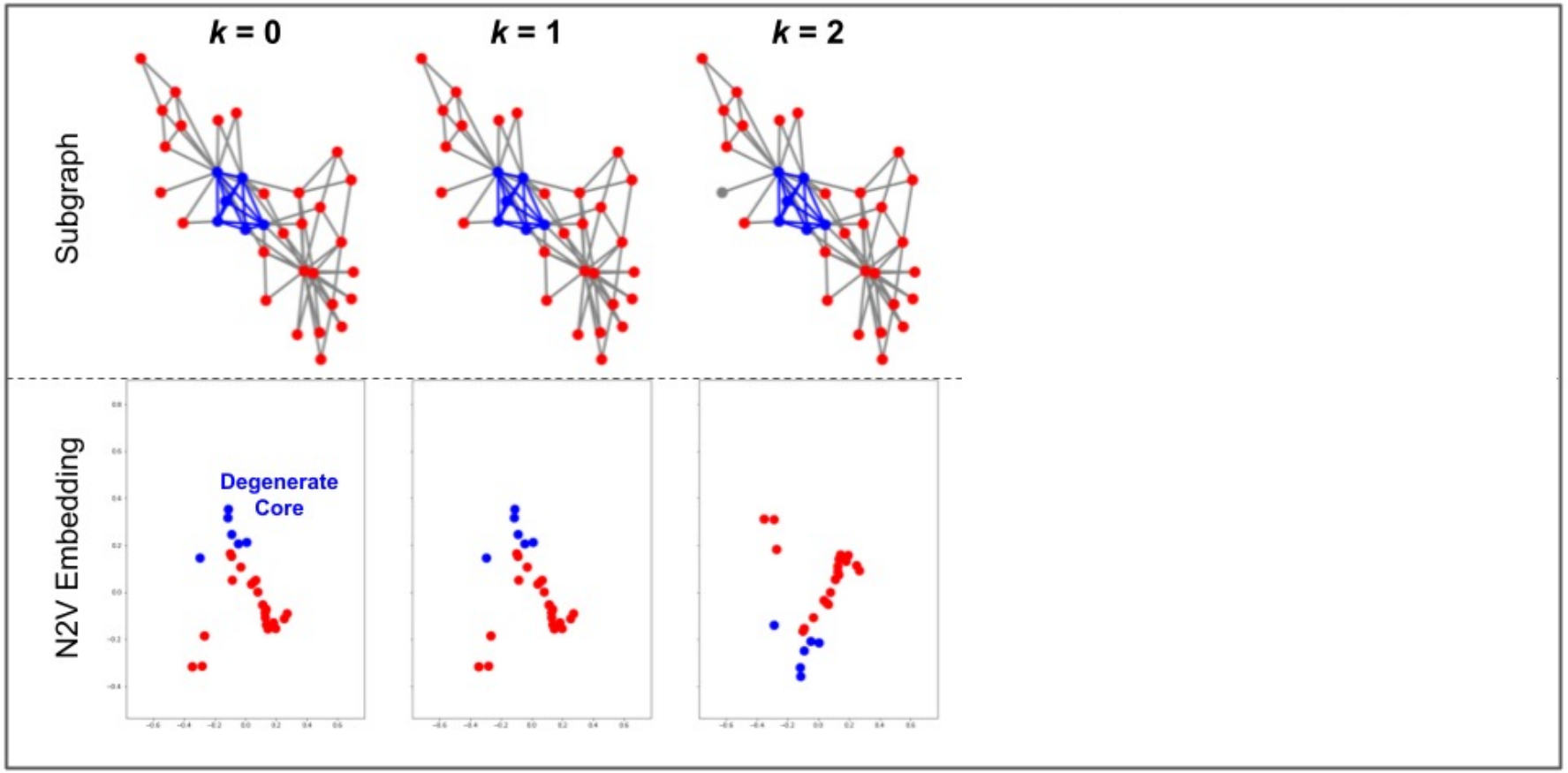
Method for evaluating stability

Shave and Re-embed



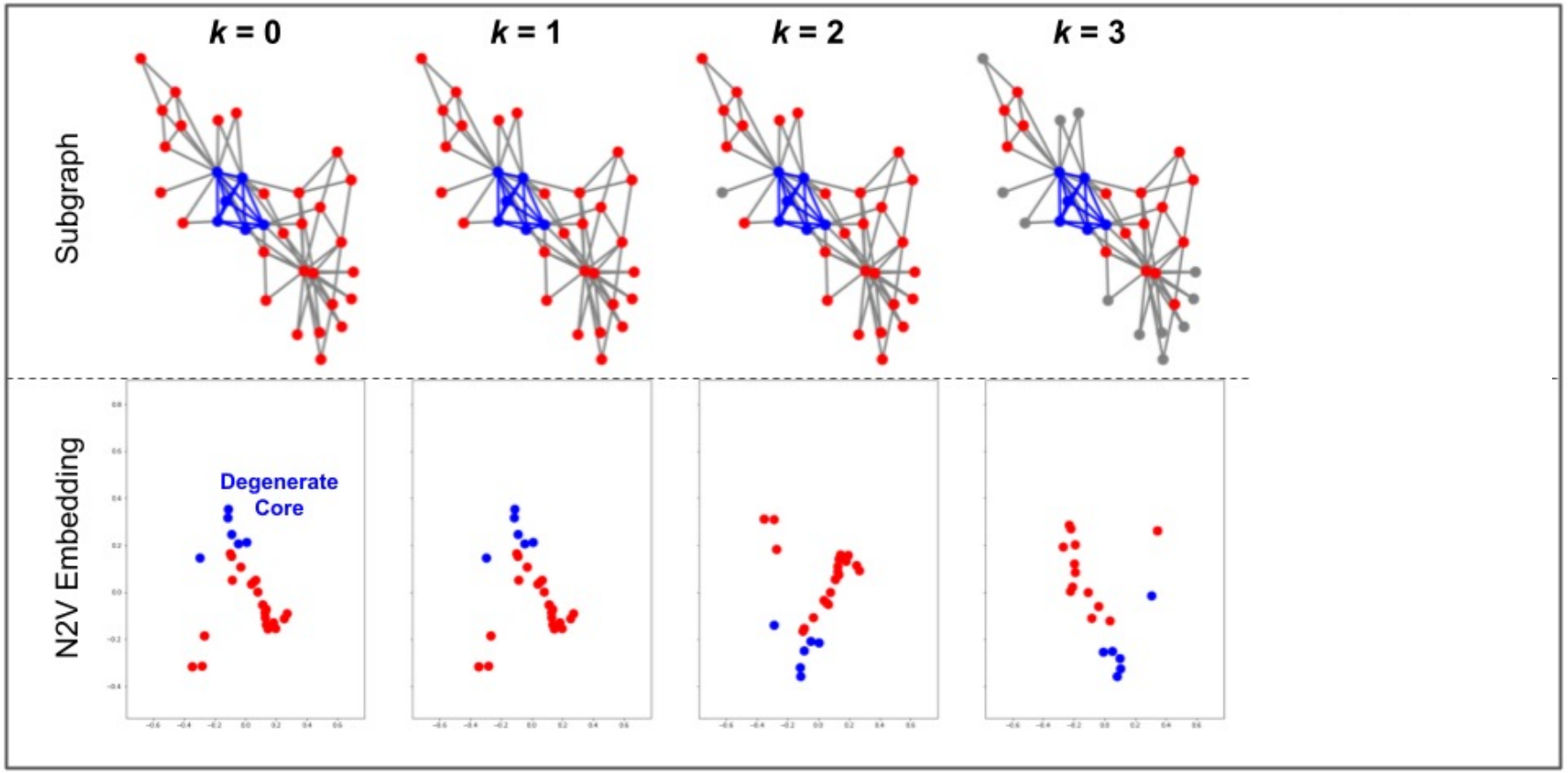
Method for evaluating stability

Shave and Re-embed



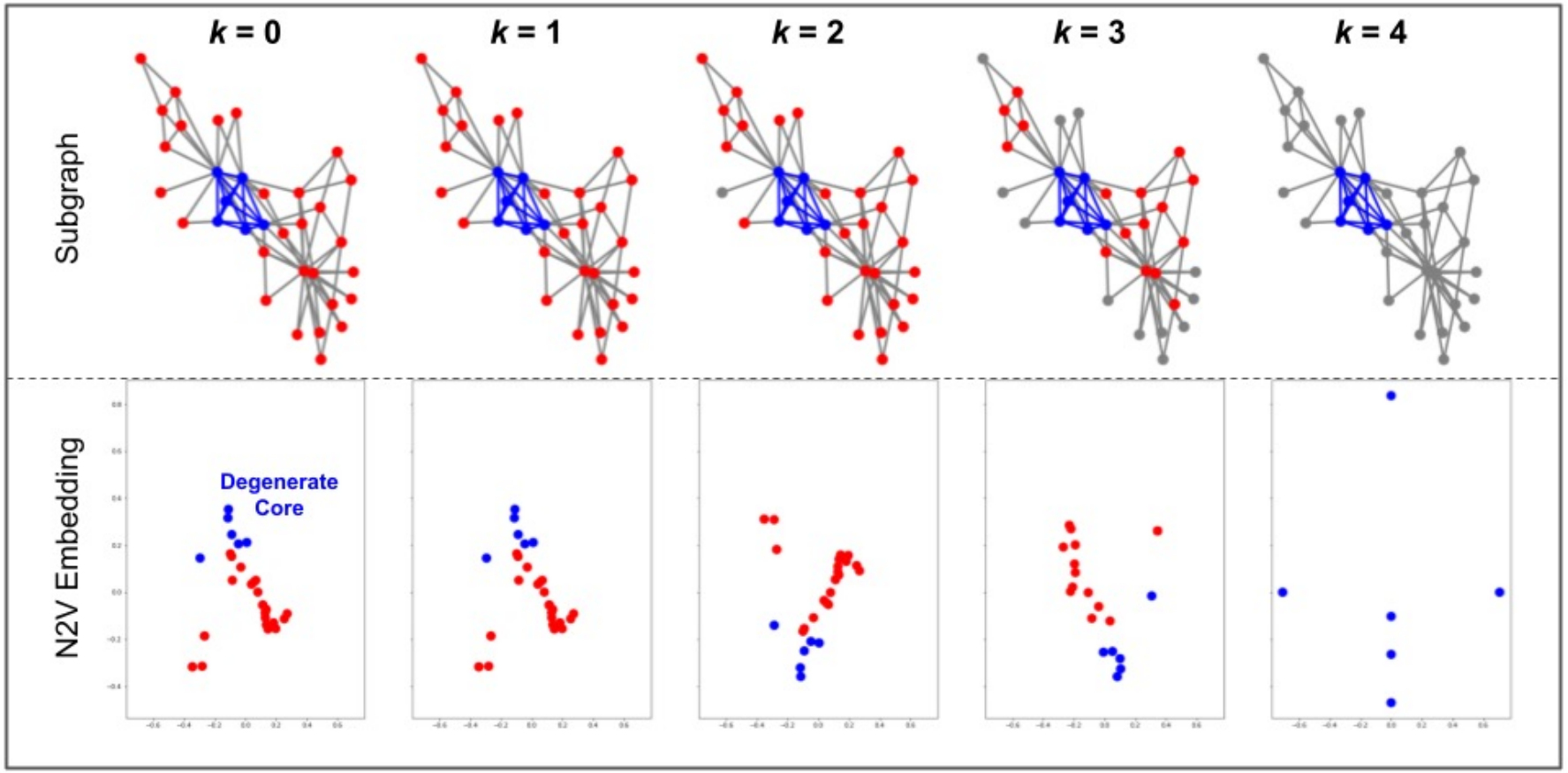
Method for evaluating stability

Shave and Re-embed



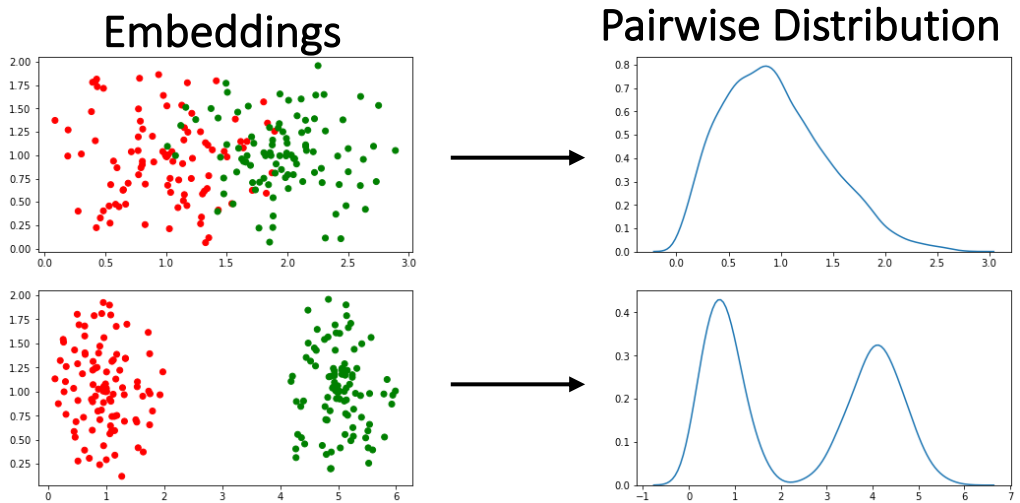
Method for evaluating stability

Shave and Re-embed



Measure of (peeling) instability

- We compute the **Degenerate-Core Pairwise Distribution,** D_k, for each k-shell: $\{\|f(v_i) - f(v_j)\| \; \forall i, j \in G_D\}$
- This captures the relative geometric relationships among the embeddings of the nodes in the degenerate core.



Measure of (peeling) instability

- We compute the **Degenerate-Core Pairwise Distribution, D_k**, for each k-shell: $\{\|f(v_i) - f(v_j)\| \; \forall i, j \in G_D\}$

- We define instability at the k^{th} core as $EMD(D_k, D_{k-1})$
  - How much does the degenerate core's pairwise distribution change between the k^{th} and $(k-1)^{\text{th}}$ shells?

EMD = Wasserstein metric

Peeling instability of real and synthetic networks

Data

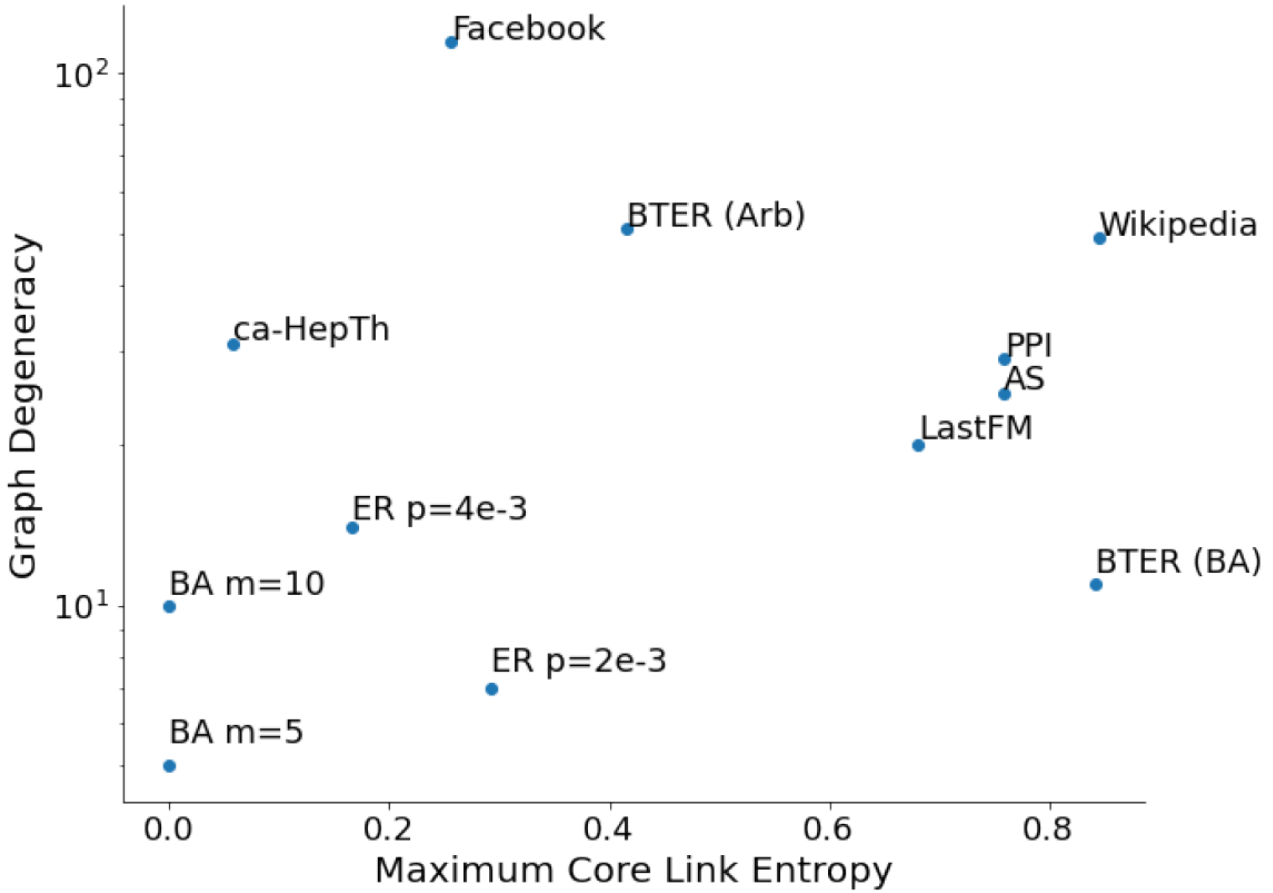
Graph	n	m	$k_{\max}$	$\%\mathcal{D}$	$G_{\mathcal{D}}$ D
Wikipedia	4.8K	185K	49	3.1	0.526
Facebook	4.0K	88K	115	3.9	0.898
PPI	3.9K	77K	29	2.8	0.404
ca-HepTh	9.9K	26K	31	0.3	1.0
LastFM	7.6K	28K	20	0.6	0.614
AS	23K	48K	25	0.3	0.545
ER ($p = .002$)	5K	25K	7	67	0.002
ER ($p = .004$)	5K	50K	14	87	0.004
BA ($m = 5$)	5K	25K	5	100	0.002
BA ($m = 10$)	5K	50K	10	100	0.004
BTER (PA)	5K	25K	1	1.5	0.234
BTER (Arb.)	4.8K	35K	51	3.3	0.445

Algorithms

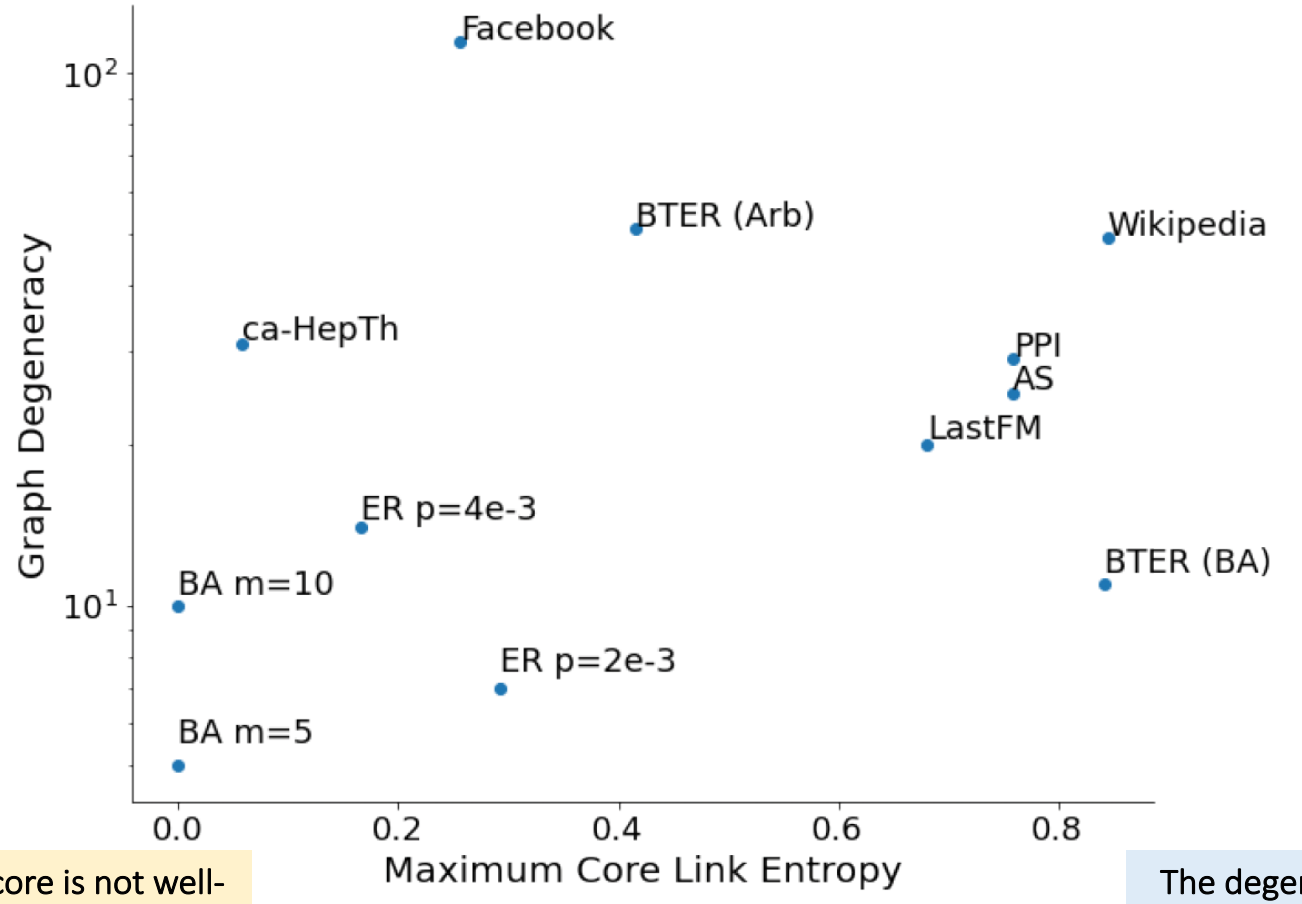
1. Hope
2. Laplacian Eigenmap
3. Node2Vec (N2V)
4. SDNE
5. Hyperbolic GCN
6. PCA

For references, see paper at https://dliu18.github.io/publication/stable

We chose graph datasets that have diverse *k*-core structures



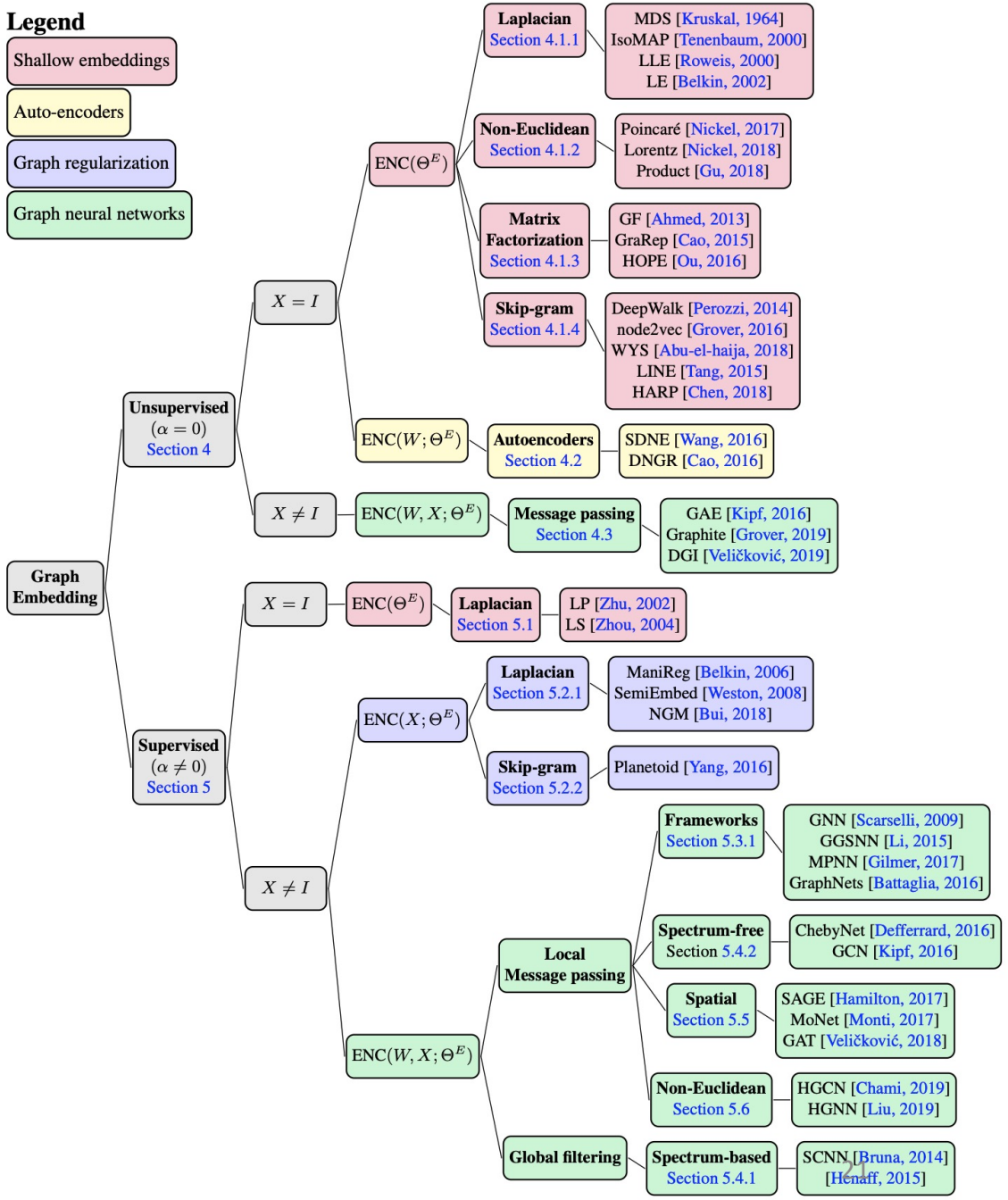
We chose graph datasets that have diverse *k*-core structures

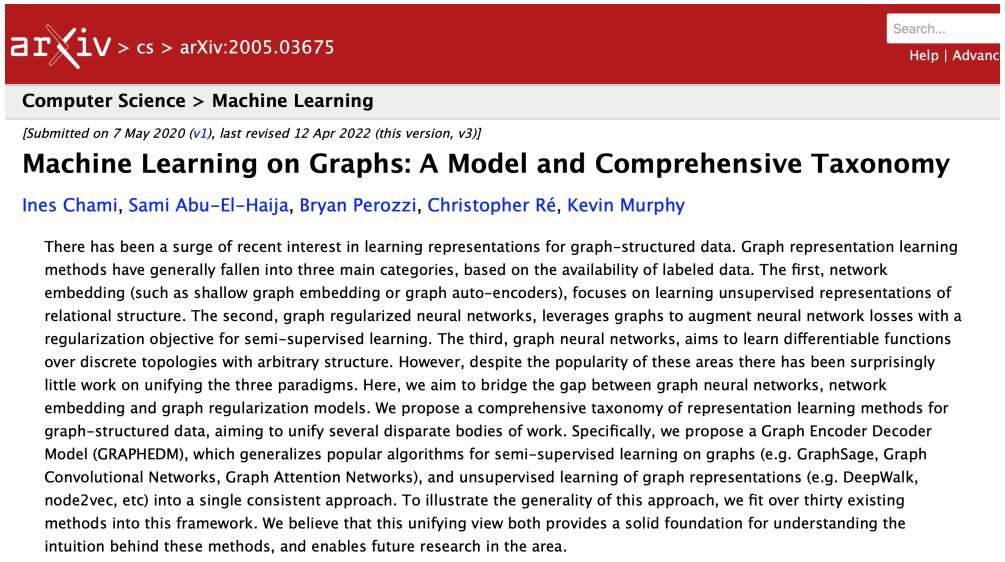


The degenerate core is not well-connected to the outer shells.

The degenerate core is well-connected to the outer shells.

We chose our algorithms based on the taxonomy of graph representation learning methods





arXiv > cs > arXiv:2005.03675

Search... Help | Advanc

Computer Science > Machine Learning

[Submitted on 7 May 2020 (v1), last revised 12 Apr 2022 (this version, v3)]

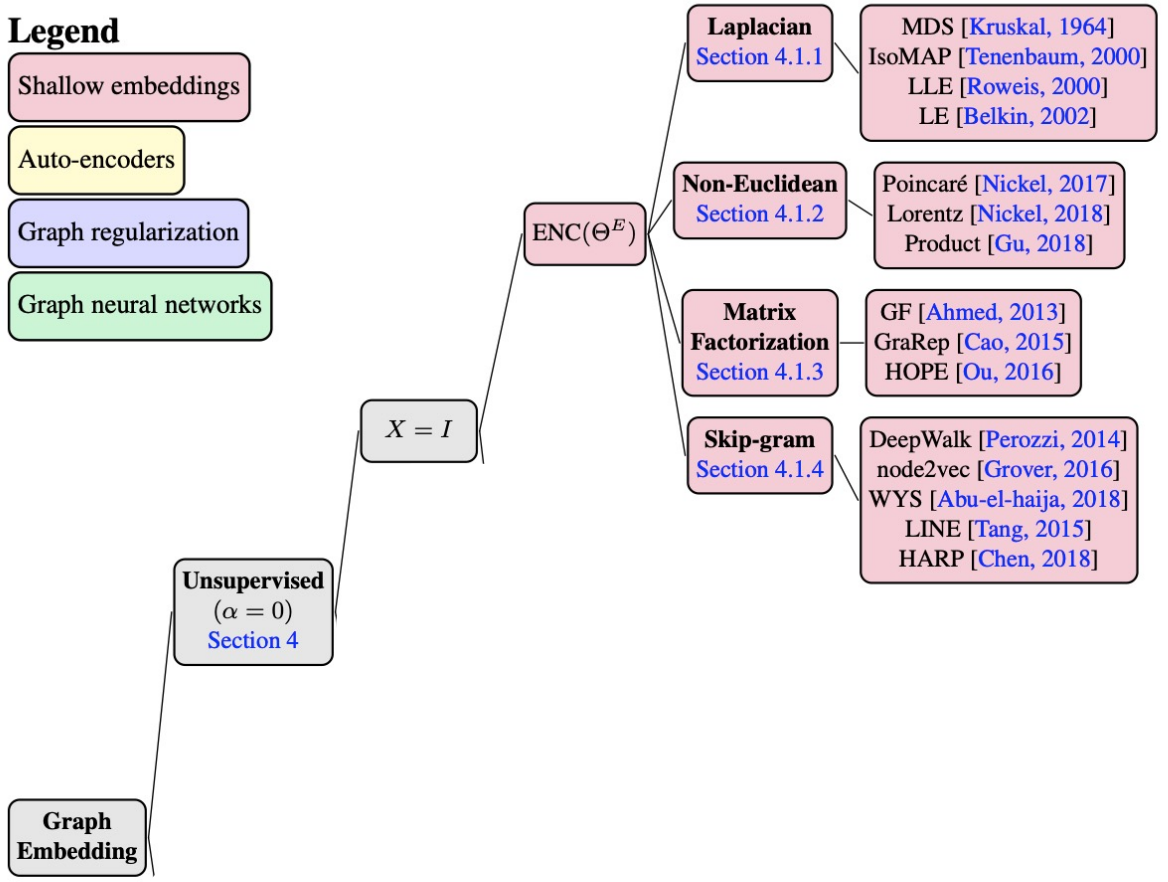
Machine Learning on Graphs: A Model and Comprehensive Taxonomy

Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, Kevin Murphy

There has been a surge of recent interest in learning representations for graph-structured data. Graph representation learning methods have generally fallen into three main categories, based on the availability of labeled data. The first, network embedding (such as shallow graph embedding or graph auto-encoders), focuses on learning unsupervised representations of relational structure. The second, graph regularized neural networks, leverages graphs to augment neural network losses with a regularization objective for semi-supervised learning. The third, graph neural networks, aims to learn differentiable functions over discrete topologies with arbitrary structure. However, despite the popularity of these areas there has been surprisingly little work on unifying the three paradigms. Here, we aim to bridge the gap between graph neural networks, network embedding and graph regularization models. We propose a comprehensive taxonomy of representation learning methods for graph-structured data, aiming to unify several disparate bodies of work. Specifically, we propose a Graph Encoder Decoder Model (GRAPHEDM), which generalizes popular algorithms for semi-supervised learning on graphs (e.g. GraphSage, Graph Convolutional Networks, Graph Attention Networks), and unsupervised learning of graph representations (e.g. DeepWalk, node2vec, etc) into a single consistent approach. To illustrate the generality of this approach, we fit over thirty existing methods into this framework. We believe that this unifying view both provides a solid foundation for understanding the intuition behind these methods, and enables future research in the area.

Figure 3 from https://arxiv.org/abs/2005.03675

We chose our algorithms based on the taxonomy of graph representation learning methods



arXiv > cs > arXiv:2005.03675

Search... Help | Advanc

Computer Science > Machine Learning

[Submitted on 7 May 2020 (v1), last revised 12 Apr 2022 (this version, v3)]

Machine Learning on Graphs: A Model and Comprehensive Taxonomy

Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, Kevin Murphy

There has been a surge of recent interest in learning representations for graph-structured data. Graph representation learning methods have generally fallen into three main categories, based on the availability of labeled data. The first, network embedding (such as shallow graph embedding or graph auto-encoders), focuses on learning unsupervised representations of relational structure. The second, graph regularized neural networks, leverages graphs to augment neural network losses with a regularization objective for semi-supervised learning. The third, graph neural networks, aims to learn differentiable functions over discrete topologies with arbitrary structure. However, despite the popularity of these areas there has been surprisingly little work on unifying the three paradigms. Here, we aim to bridge the gap between graph neural networks, network embedding and graph regularization models. We propose a comprehensive taxonomy of representation learning methods for graph-structured data, aiming to unify several disparate bodies of work. Specifically, we propose a Graph Encoder Decoder Model (GRAPHEDM), which generalizes popular algorithms for semi-supervised learning on graphs (e.g. GraphSage, Graph Convolutional Networks, Graph Attention Networks), and unsupervised learning of graph representations (e.g. DeepWalk, node2vec, etc) into a single consistent approach. To illustrate the generality of this approach, we fit over thirty existing methods into this framework. We believe that this unifying view both provides a solid foundation for understanding the intuition behind these methods, and enables future research in the area.

Figure 3 from https://arxiv.org/abs/2005.03675

We found three patterns

1. Degenerate-core embeddings are sensitive to the removal of specific *k*-shells.

We found three patterns

1. Degenerate-core embeddings are sensitive to the removal of specific *k*-shells.

2. Degenerate-core embeddings for Erdős-Rényi (ER) and Barabási-Albert (BA) random graphs are stable.

We found three patterns

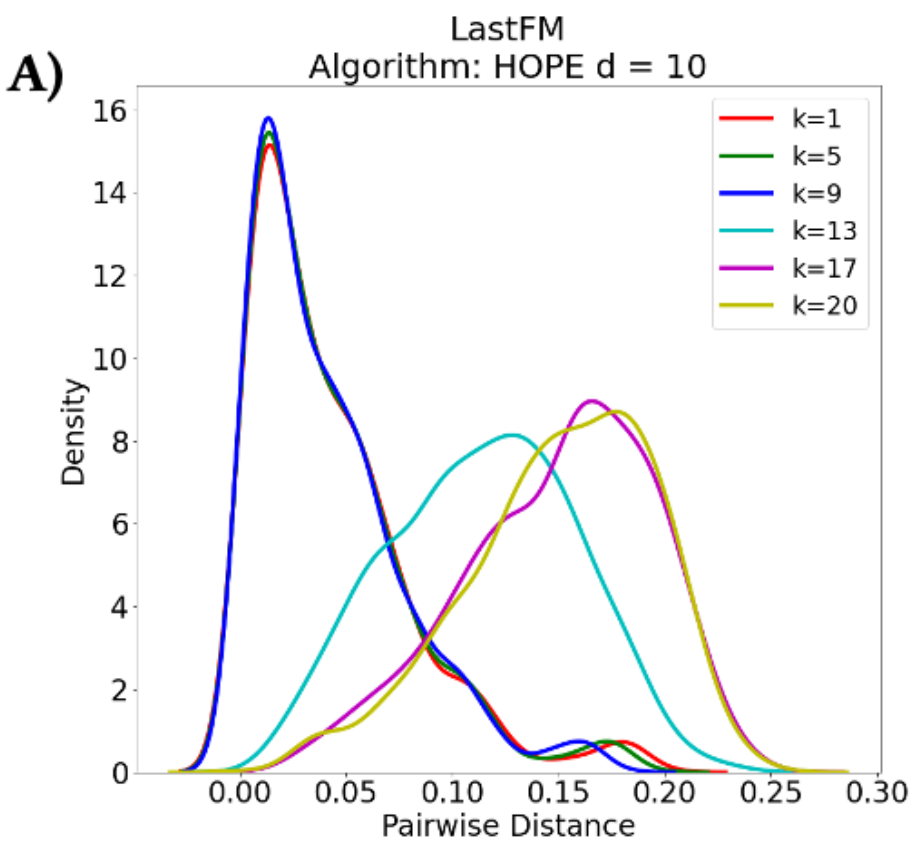
1. Degenerate-core embeddings are sensitive to the removal of specific *k*-shells.

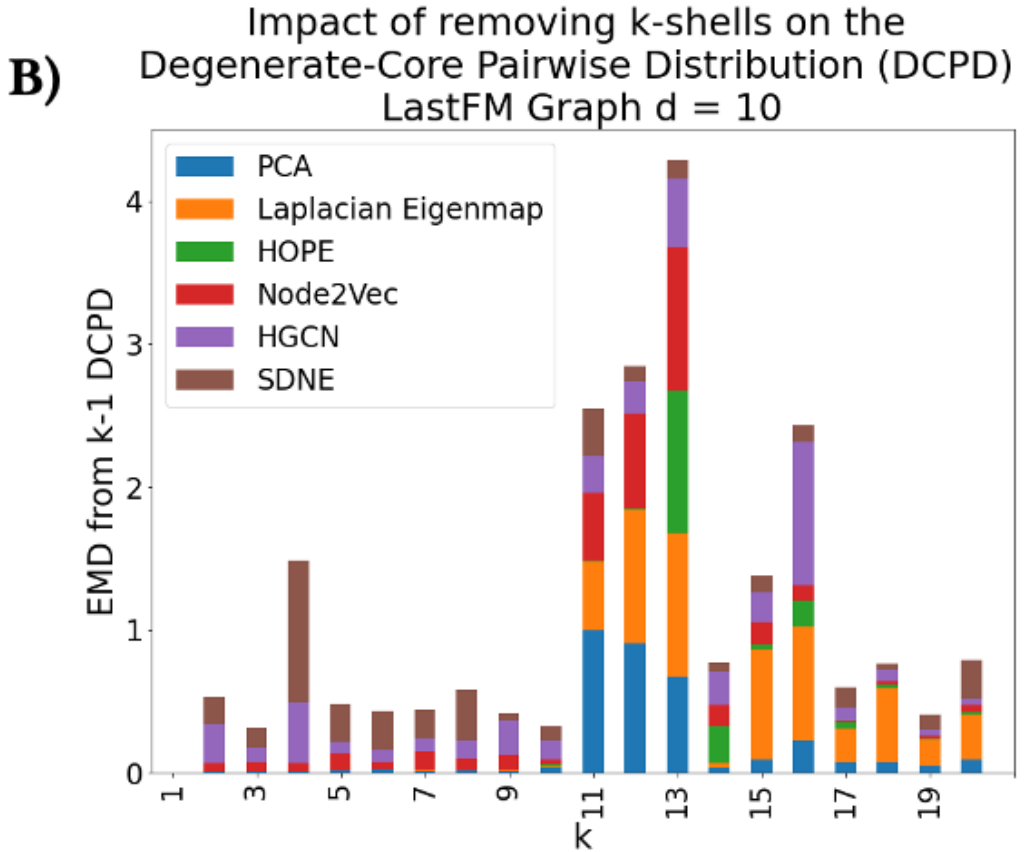
2. Degenerate-core embeddings for Erdős-Rényi (ER) and Barabási-Albert (BA) random graphs are stable.

3. As the periphery is removed, the degenerate-core pairwise distribution becomes smoother.

Pattern 1: Point of instability

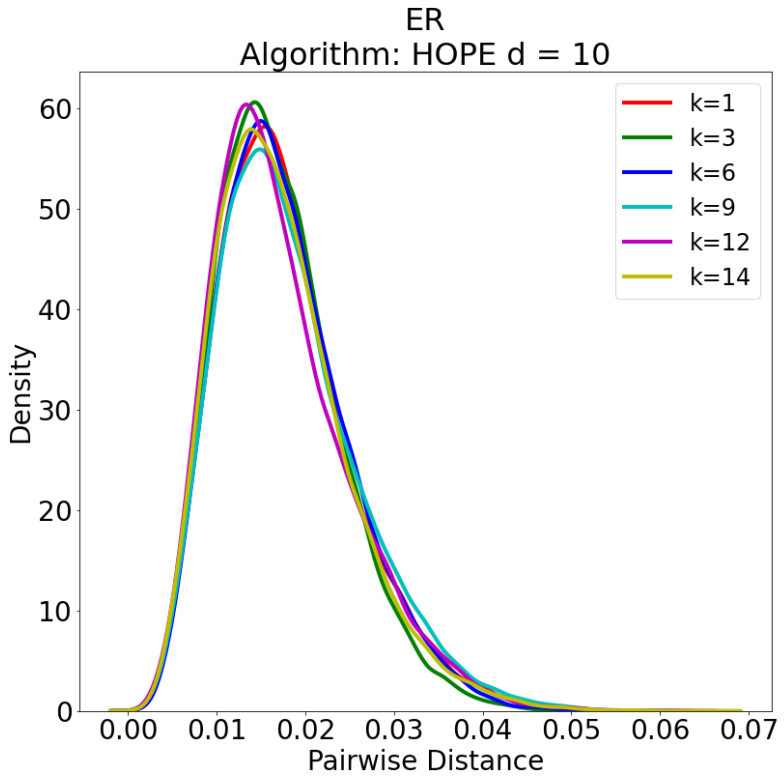
Degenerate-core embeddings are sensitive
to the removal of specific *k*-shells.

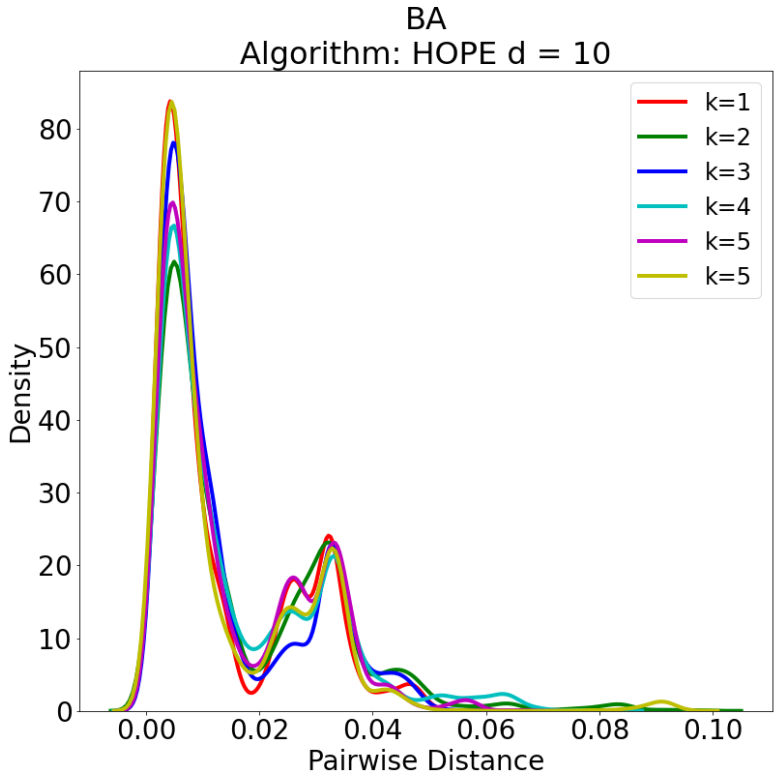




Pattern 2: Stable ER and BA graphs

Degenerate-core embeddings for Erdős-Rényi (ER) and Barabási-Albert (BA) random graphs are stable.

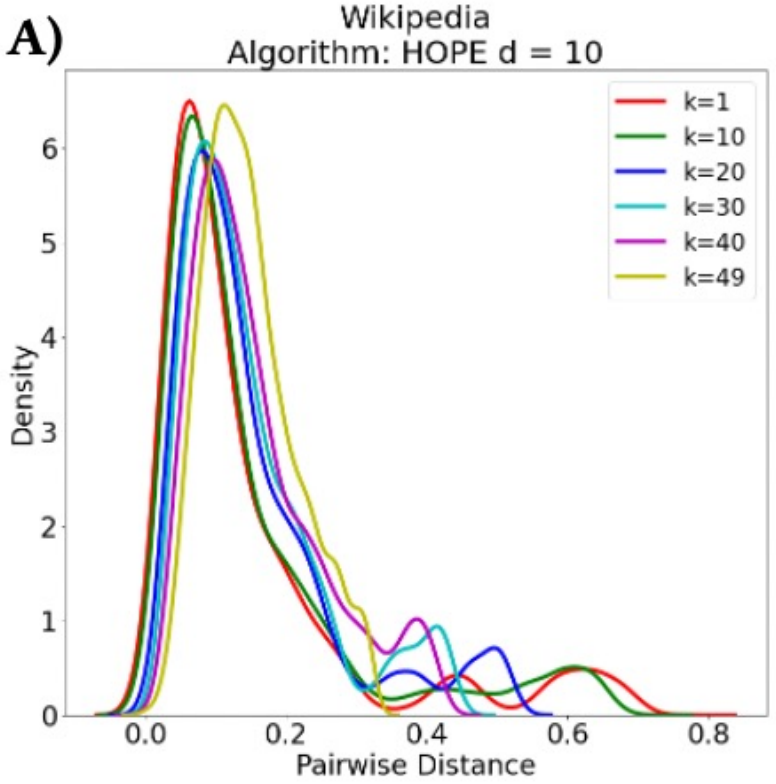


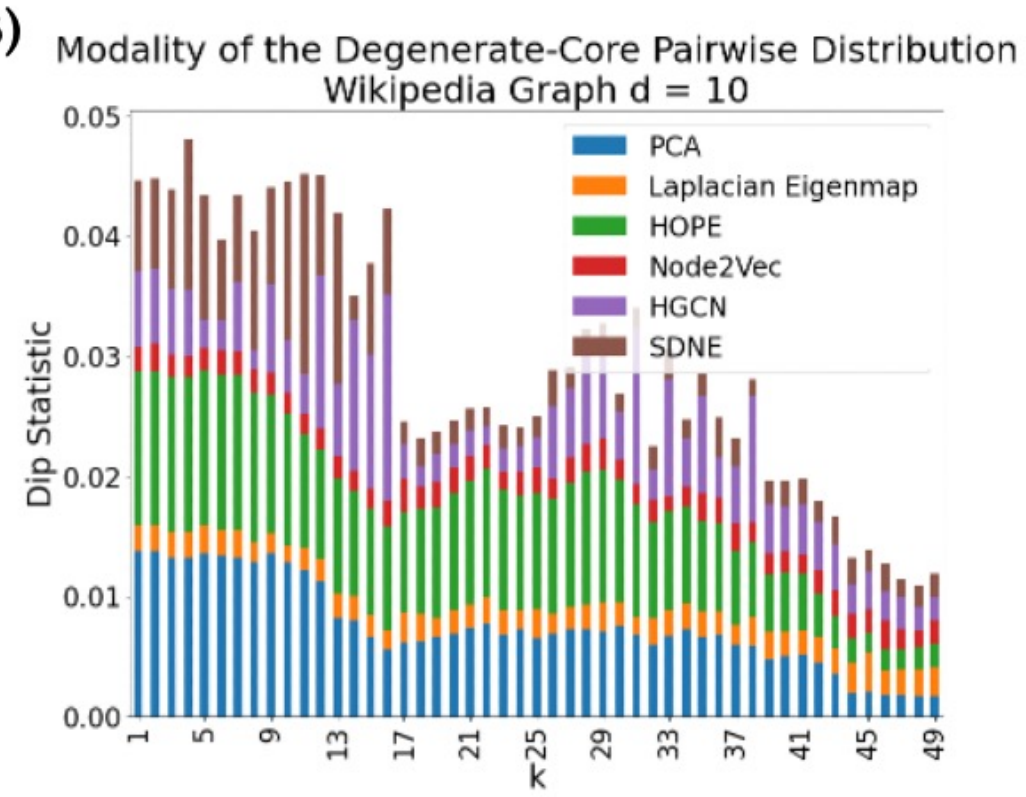


The degenerate core constitutes a large proportion of the entire graph.

Pattern 3: Smoother pairwise distribution

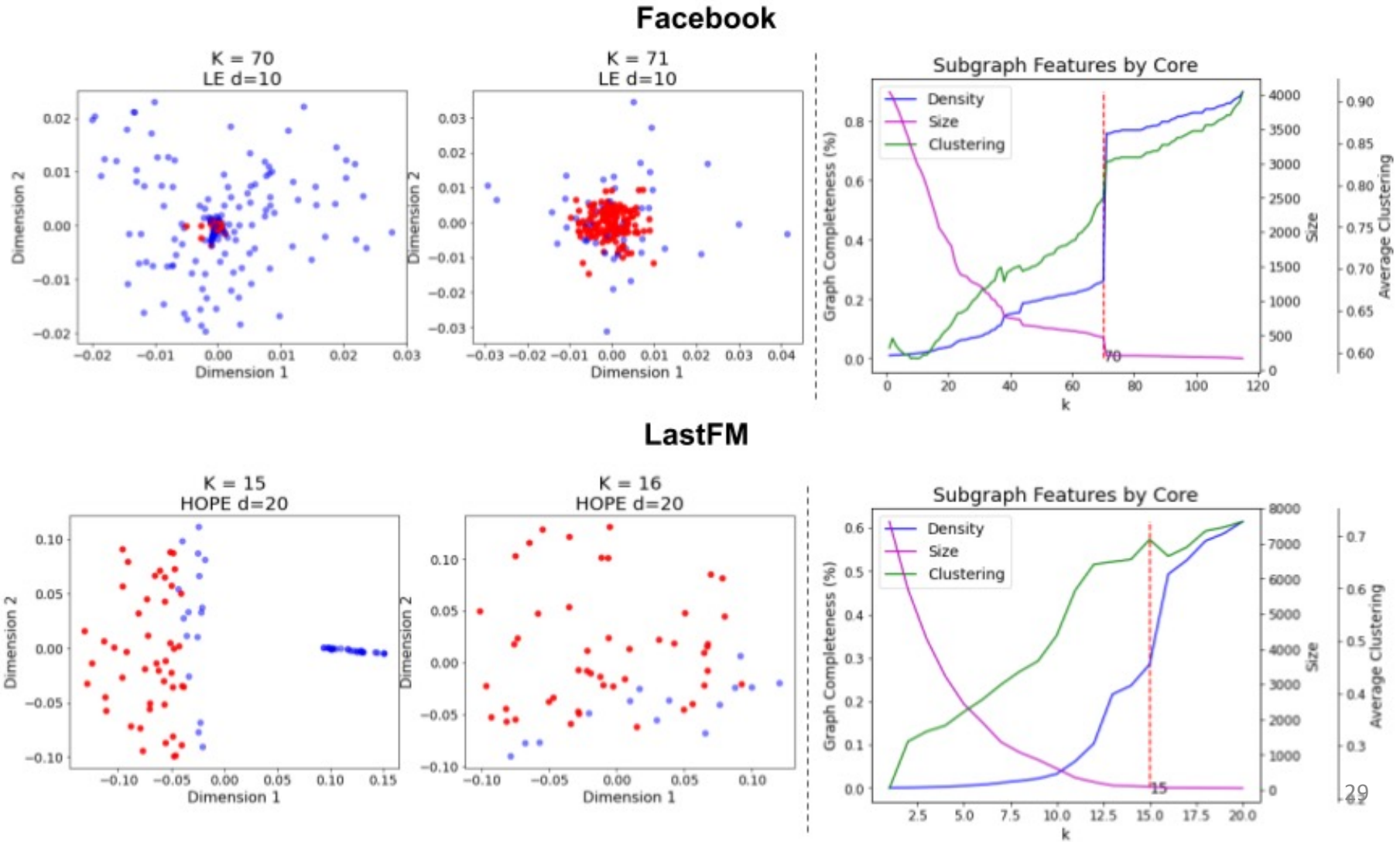
As the periphery is removed, the degenerate-core pairwise distribution becomes smoother.





Hartigan's Dip static: a value close to zero suggests unimodality

Maximum instability of embeddings is correlated with increases in edge density

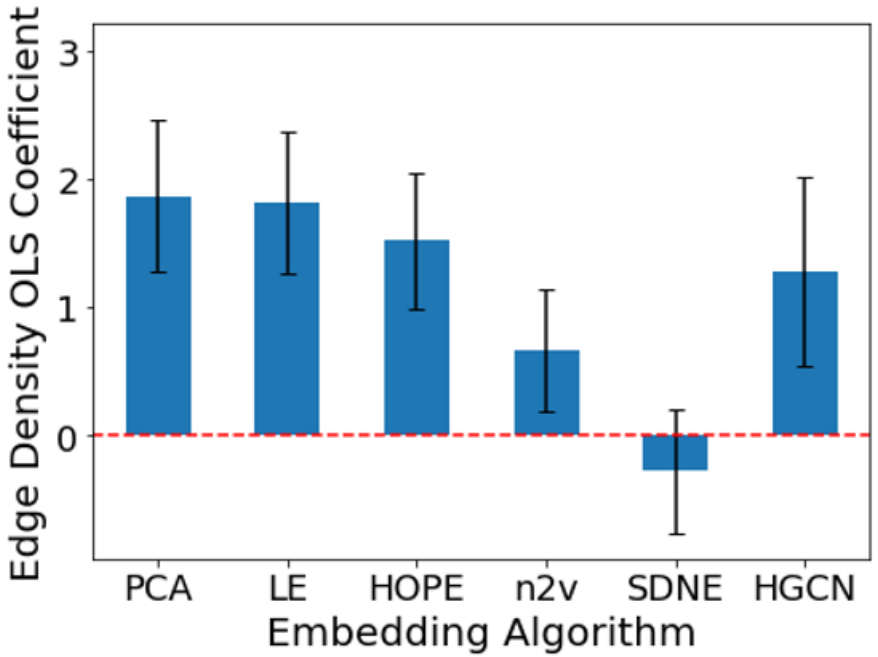


The degenerate core is colored red.

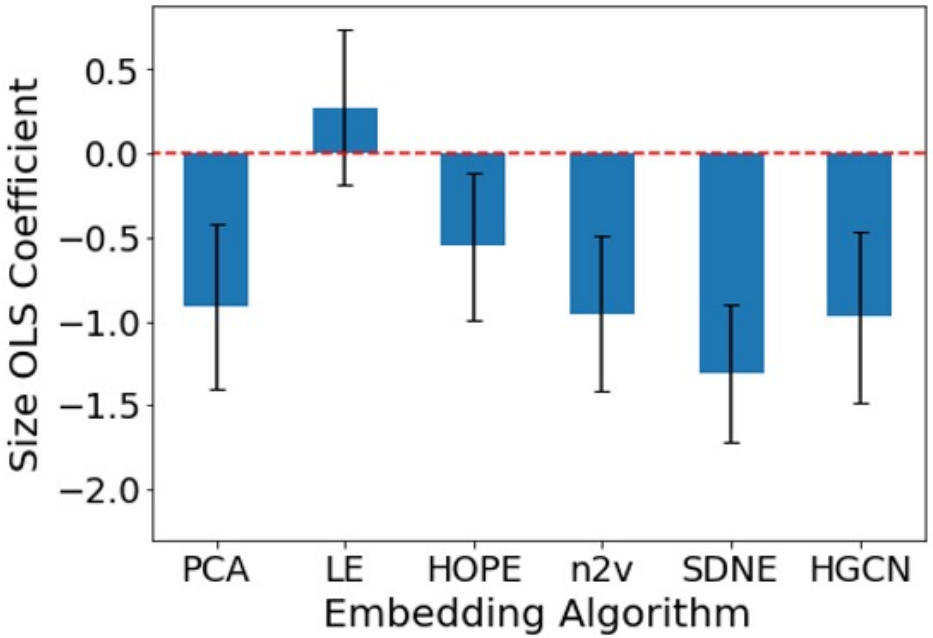
Increases in subgraph edge density are correlated with peeling instability

$$\Delta EMD = \beta_0 + \beta_1 \Delta_{size} + \beta_2 \Delta_{edge\ density} + \beta_3 \Delta_{clustering} + \beta_4 \Delta_{transitivity}$$

Edge Density



Size



Graph size = # of nodes in the subgraph / # of nodes in G. The error bars report 95% confidence intervals.

For ER graphs embedded with Laplacian Eigenmaps, the best and worst possible embeddings become less distinguishable as edge density increases.

THEOREM 3.1. *Let G be an Erdös-Rényi graph with n nodes and edge probability p, and let $l_G(X)$ be the Laplacian Eigenmap loss for a set of embeddings $X \in \mathbb{R}^{n \times d}$ with embedding dimension d. Then, almost surely, the gap between the best set of embeddings and the worst set decreases as a function of p, where the ratio is lower-bounded as:*

$$\frac{min_X l_G(X)}{max_X l_G(X)} \geq \frac{\sqrt{(n-1)p} - o(1)}{\sqrt{(n-1)p} + o(1)}$$

See supplementary material for proof: https://dliu18.github.io/files/papers/stable/stable-Supplemental-SDM-23.pdf

STABLE: Our algorithm for peeling stability

Augment an existing loss function

$$\mathbf{Y}^* = \underset{\mathbf{Y} \in \mathbb{R}^{nxd}}{\arg\min}\, \mathcal{L}_b\,(\mathbf{Y}, W) + \alpha \mathcal{L}_s\,(\mathbf{Y}, W, \mathcal{D})$$

Original loss function. Embed similar nodes close to each other.

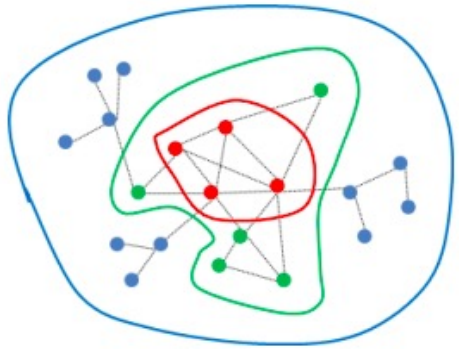
Penalize unstable embeddings of the degenerate core

Define Instability Loss

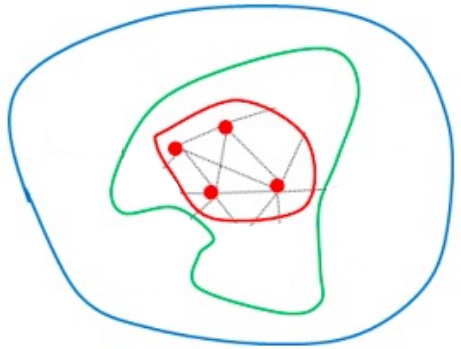
First-order Proximity

$$p\left(\boldsymbol{u}_i, \boldsymbol{u}_j\right) = \frac{1}{1 + e^{-\boldsymbol{u}_i^T \boldsymbol{u}_j}}$$

Larger *p* implies greater similarity



$\boldsymbol{u}$



$\hat{\boldsymbol{u}}$

$$\mathcal{L}_s = \sum_{i,j \in \mathcal{D}} \left| p\left(\boldsymbol{u}_i, \boldsymbol{u}_j\right) - p\left(\hat{\boldsymbol{u}}_i, \hat{\boldsymbol{u}}_j\right) \right|^2$$

If *i* and *j* are similar in $\hat{\boldsymbol{u}}$, then embed them similarly in $\boldsymbol{u}$.

STABLE's embeddings preserve AUC on link prediction

Graph	Laplacian Eigenmaps AUC		LINE AUC	
	Base	STABLE	Base	STABLE
Facebook	**0.982 ± 0.000**	**0.924 ± 0.047**	**0.971 ± 0.001**	**0.933 ± 0.001**
LastFM	**0.910 ± 0.001**	**0.785 ± 0.001**	**0.914 ± 0.001**	**0.895 ± 0.002**
ca-HepTh	0.811 ± 0.008	0.811 ± 0.008	**0.893 ± 0.003**	**0.890 ± 0.001**
Protein-Protein	0.770 ± 0.016	0.761 ± 0.008	0.638 ± 0.036	0.660 ± 0.022
Autonomous Systems	0.693 ± 0.002	0.699 ± 0.020	**0.693 ± 0.007**	**0.672 ± 0.004**
Wikipedia	0.614 ± 0.001	0.615 ± 0.001	**0.458 ± 0.008**	**0.499 ± 0.003**

We found …

1. Degenerate core embeddings of real-world networks are unstable to periphery perturbations.
   - Significant because real networks are noisy / incomplete and embeddings are used in downstream tasks.
2. Instability spikes when edge density increases.
3. We present a generic algorithm, STABLE, that augments existing graph embedding algorithms to produce more stable embeddings.

Project page: https://dliu18.github.io/publication/stable
Code at: https://github.com/dliu18/stable

There are other forms of instability

- How does negative sampling affect the stability of node embeddings?
  - A common negative sampling method is to sample from a Zipf distribution with ¾ as the value of the exponent characterizing the distribution
  - This value (¾) produces good results, but why?

C. “Sesh” Seshadhri (UCSC)

The impossibility of low-rank representations for triangle-rich complex networks

C. Seshadhri[a,1], Aneesh Sharma[b], Andrew Stolman[a], and Ashish Goel[c]

[a]Department of Computer Science, University of California, Santa Cruz, CA 95064; [b]Google, Mountain View, CA 94043; and [c]Department of Management Science and Engineering, Stanford University, Stanford, CA 94305

Edited by Mark E. J. Newman, University of Michigan, Ann Arbor, MI, and accepted by Editorial Board Member Peter J. Bickel February 1, 2020 (received for review June 26, 2019)

The study of complex networks is a significant development in modern science, and has enriched the social sciences, biology, physics, and computer science. Models and algorithms for such networks are pervasive in our society, and impact human behavior via social networks, search engines, and recommender systems, to name a few. A widely used algorithmic technique for modeling such complex networks is to construct a low-dimensional Euclidean embedding of the vertices of the network, where proximity of vertices is interpreted as the likelihood of an edge. Contrary to the common view, we argue that such graph embeddings do not capture salient properties of complex networks. The two properties we focus on are low degree and large clustering coefficients, which have been widely established to be empirically true for real-world networks. We mathematically prove that any embedding (that uses dot products to measure similarity) that can successfully create these two properties must have a rank that is nearly linear in the number of vertices. Among other implications, this establishes that popular embedding techniques such as singular value decomposition and node2vec fail to capture significant structural aspects of real-world complex networks. Furthermore, we empirically study a number of different embedding techniques based on dot product, and show that they all fail to capture the triangle structure.

graph embeddings | graph representations | low-dimensional embeddings | low-rank representations | singular value decomposition

Complex networks (or graphs) are a fundamental object of study in modern science, across domains as diverse as the social sciences, biology, physics, computer science, and engineering (1–3). Designing good models for these networks is a crucial area of research, and also affects society at large, given the role of online social networks in modern human interaction (4–6). Complex networks are massive, high-dimensional, discrete objects, and are challenging to work with in a modeling context. A common method of dealing with this challenge is to construct a low-dimensional Euclidean embedding that tries to capture the structure of the network (see ref. 7 for a recent survey). Formally, we think of the n vertices as vectors $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n \in \mathbb{R}^d$, where d is typically constant (or very slowly growing in n). The likelihood of an edge (i, j) is proportional to (usually a nonnegative monotone function in) $\vec{v}_i \cdot \vec{v}_j$ (8, 9). This gives a graph distribution that the observed network is assumed to be generated from.

The most important method to get such embeddings is the singular value decomposition (SVD) or other matrix factorizations of the adjacency matrix (8). Recently, there has also been an explosion of interest in using methods from deep neural networks to learn such graph embeddings (9–12) (refer to ref. 7 for more references). Regardless of the specific method, a key goal in building an embedding is to keep the dimension d small—while trying to preserve the network structure—as the embeddings are used in a variety of downstream modeling tasks such as graph clustering, nearest-neighbor search, and link prediction (13). Yet a fundamental question remains unanswered: To what extent do such low-dimensional embeddings actually capture the structure of a complex network?

These models are often justified by treating the (few) dimensions as “interests” of individuals, and using similarity of interests (dot product) to form edges. Contrary to the dominant view, we argue that low-dimensional embeddings are not good representations of complex networks. We demonstrate mathematically and empirically that they lose local structure, one of the hallmarks of complex networks. This runs counter to the ubiquitous use of SVD in data analysis. The weaknesses of SVD have been empirically observed in recommendation tasks (14–16), and our result provides a mathematical validation of these findings.

Let us define the setting formally. Consider a set of vectors $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n \in \mathbb{R}^d$ (denoted by the $d \times n$ matrix V) used to represent the n vertices in a network. Let $\mathcal{G}_V$ denote the following distribution of graphs over the vertex set $[n]$. For each index pair i, j, independently insert (undirected) edge (i, j) with probability $\max(0, \min(\vec{v}_i \cdot \vec{v}_j, 1))$. (If $\vec{v}_i \cdot \vec{v}_j$ is negative, (i, j) is never inserted. If $\vec{v}_i \cdot \vec{v}_j \geq 1$, (i, j) is always inserted.) We will refer to this model as the “embedding” of a graph G, and focus on this formulation in our theoretical results. This is a standard model in the literature, and subsumes the classic Stochastic Block Model (17) and Random Dot Product Model (18, 19). There are alternate models that use different functions of the dot product for

Significance

Our main message is that the popular method of low-dimensional embeddings provably cannot capture important properties of real-world complex networks. A widely used algorithmic technique for modeling these networks is to construct a low-dimensional Euclidean embedding of the vertices of the network, where proximity of vertices is interpreted as the likelihood of an edge. Contrary to common wisdom, we argue that such graph embeddings do not capture salient properties of complex networks. We mathematically prove that low-dimensional embeddings cannot generate graphs with both low average degree and large clustering coefficients, which have been widely established to be empirically true for real-world networks. This establishes that popular low-dimensional embedding methods fail to capture significant structural aspects of real-world complex networks.

Author contributions: C.S., A. Sharma, A. Stolman, and A.G. designed research, performed research, analyzed data, and wrote the paper.

The authors declare no competing interest.

This article is a PNAS Direct Submission. M.E.J.N. is a guest editor invited by the Editorial Board.

This open access article is distributed under Creative Commons Attribution License 4.0 (CC BY).

[1]To whom correspondence may be addressed. Email: sesh@ucsc.edu.

This article contains supporting information online at https://www.pnas.org/lookup/suppl/doi:10.1073/pnas.1911030117/-/DCSupplemental.

First published March 2, 2020.

https://www.pnas.org/content/early/2020/02/26/1911030117

Classic Graph Structural Features Outperform Factorization-Based Graph Embedding Methods on Community Labeling

Andrew Stolman* Caleb Levy† C. Seshadhri‡ Aneesh Sharma§

Abstract

Graph representation learning (also called *graph embeddings*) is a popular technique for incorporating network structure into machine learning models. Unsupervised graph embedding methods aim to capture graph structure by learning a low-dimensional vector representation (the *embedding*) for each node. Despite the widespread use of these embeddings for a variety of downstream transductive machine learning tasks, there is little principled analysis of the effectiveness of this approach for common tasks. In this work, we provide an empirical and theoretical analysis for the performance of a class of embeddings on the common task of pairwise community labeling. This is a binary variant of the classic community detection problem, which seeks to build a classifier to determine whether a pair of vertices participate in a community. In line with our goal of foundational understanding, we focus on a popular class of unsupervised embedding techniques that learn low rank factorizations of a vertex proximity matrix (this class includes methods like GraRep, DeepWalk, node2vec, NetMF). We perform detailed empirical analysis for community labeling over a variety of real and synthetic graphs with ground truth. In all cases we studied, the models trained from embedding features perform poorly on community labeling. In constrast, a simple logistic model with classic graph structural features handily outperforms the embedding models. For a more principled understanding, we provide a theoretical analysis for the (in)effectiveness of these embeddings in capturing the community structure. We formally prove that popular low-dimensional factorization methods either cannot produce community structure, or can only produce “unstable” communities. These communities are inherently unstable under small perturbations. This theoretical result suggests that even though “good” factorizations exist, they are unlikely to be found by computational methods.

1 Introduction

Graph structured data is ubiquitous. Capturing the graph structure is important for a wide variety of machine learning tasks, such as ranking in social networks, content recommendations, and clustering [EK10]. A long-studied challenge for building such machine learned models has been to capture the graph structure for use in a variety of modeling tasks. *Graph representation learning*, or *low-dimensional graph embeddings*, provide a convenient solution to this problem. Given a graph G on n vertices, these methods map each vertex to a vector in $\mathbb{R}^d$, where $d \ll n$, in an unsupervised or a self-supervised manner (it is also sometimes referred to as a pre-training procedure). Typically, the goal of the embedding is to represent graph proximity by (a function of) the dot product of vectors, thereby implicitly giving a geometric representation of the graph.[1] The dot product formulation provides a convenient form for building a models (e.g. using deep learning). Moreover, the geometry of the embedding allows efficient reverse-index lookups, using nearest neighbor search [CAS16, Twi18].

The study of low-dimensional graph embeddings is an incredibly popular research area, and has generated many exciting results over the past few years (see surveys [HYL18, CAEHP+20] and a Chapter 23 in [Mur21]). Nonetheless, there is limited principled understanding of the power of low-dimensional embeddings (a few recent papers address this topic [SSSG20, CMST20, Lou20, GJJ20]). Our work aims to understand the effectiveness of a class of graph embeddings in preserving graph structure as it manifests in performance on different downstream

*University of California, Santa Cruz. astolman@ucsc.edu

†University of California, Santa Cruz. cclevy@ucsc.edu

‡University of California, Santa Cruz. sesh@ucsc.edu. Supported by NSF DMS-2023495, CCF-1740850, CCF-1813165, CCF-1839317, CCF-1908384, CCF-1909790, and ARO Award W911NF1910294.

§Google. aneesh@google.com

[1]Since there is a wide range of methods for Graph representation learning, we refer the reader to the “Shallow embeddings” class in a recent survey [CAEHP+20] for a more comprehensive overview.

https://arxiv.org/pdf/2201.08481.pdf

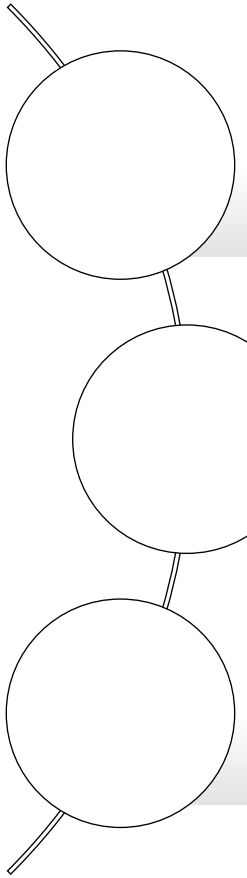
Find an embedding with geometric properties instead of spectral ones

- There is a bijection between undirected graphs on n nodes and $n-1$ dimensional simplices.
- We can encode graph structure in geometric terms using the simplex geometry of the Laplacian.
- Some of the most popular graph embedding methods (that rely on distance minimization) perform well **only** when clustering coefficient is high.

Leo Torres

L. Torres, K.S. Chan, T. Eliassi-Rad. **GLEE: Geometric Laplacian Eigenmap Embedding**. *Journal of Complex Networks*, 8(2), cnaa007, 2020.

Outline



Instability of embeddings

Emergence of topological shortcuts

Adversarial ML

what the model learns

$\neq$

what you think the model learns

An old example of what the model learns ≠ what you think the model learns

AI & Soc (1992) 6: 18–26
© 1992 Springer-Verlag London Limited

AI & SOCIETY

What Artificial Experts Can and Cannot Do

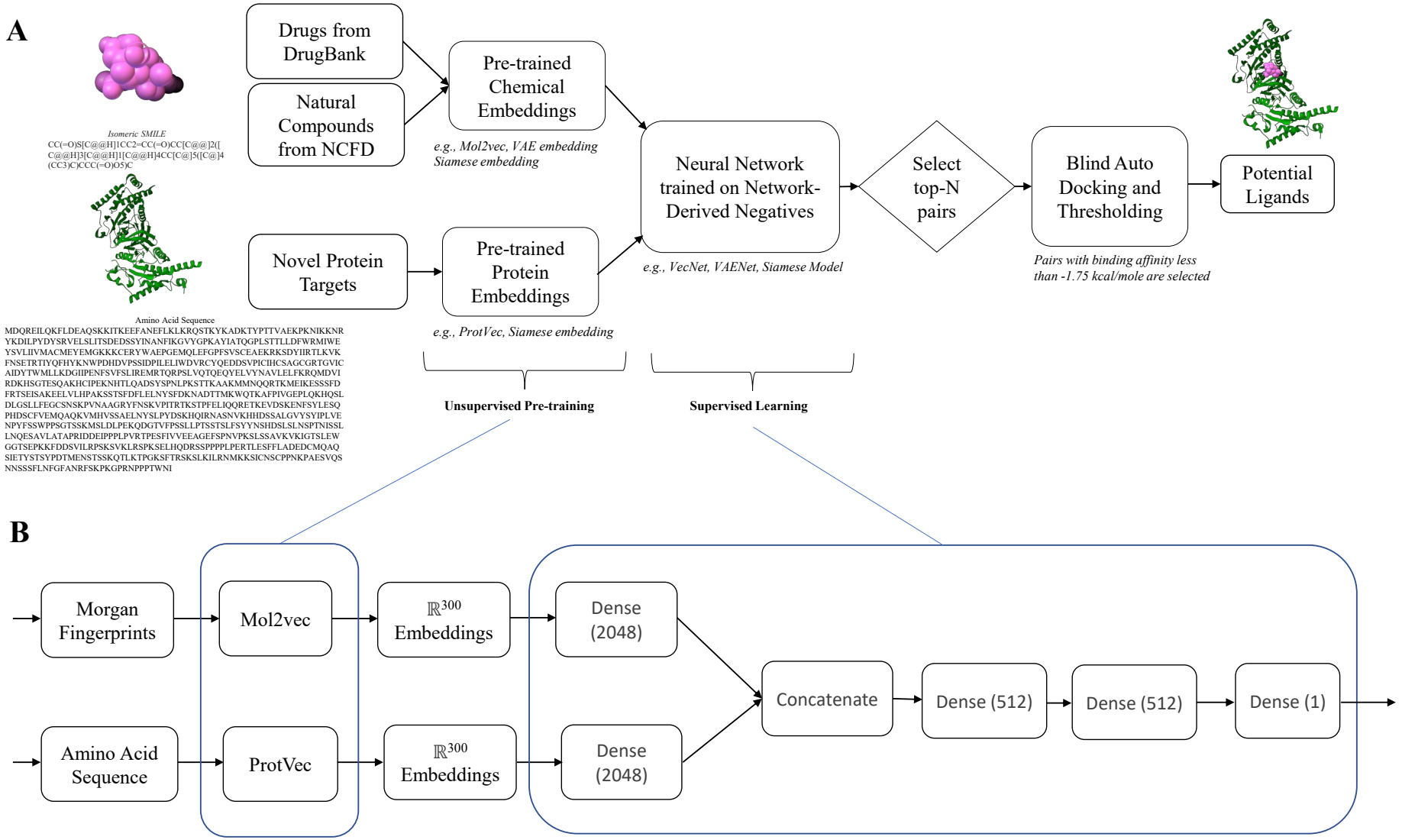
Hubert L. Dreyfus and Stuart E. Dreyfus
Department of Philosophy, University of California, Berkeley, USA

Abstract. One's model of skill determines what one expects from neural network modelling and how one proposes to go about enhancing expertise. We view skill acquisition as a progression from acting on the basis of a rough theory of a domain in terms of facts and rules to being able to respond appropriately to the current situation on the basis of neuron connections changed by the results of responses to the relevant aspects of many past situations. Viewing skill acquisition in this ways suggests how one can avoid the problem currently facing AI of how to train a network to make human-like generalizations. In training a network one must progress, as the human learner does, from rules and facts to wholistic responses. As to future work, from our perspective one should not try to enhance expertise as in traditional AI by attempting to construct improved theories of a domain, but rather by improving the learner's access to the relevant aspects of a domain so as to facilitate learning from experience.

How did we discover the emergence of topological shortcuts?

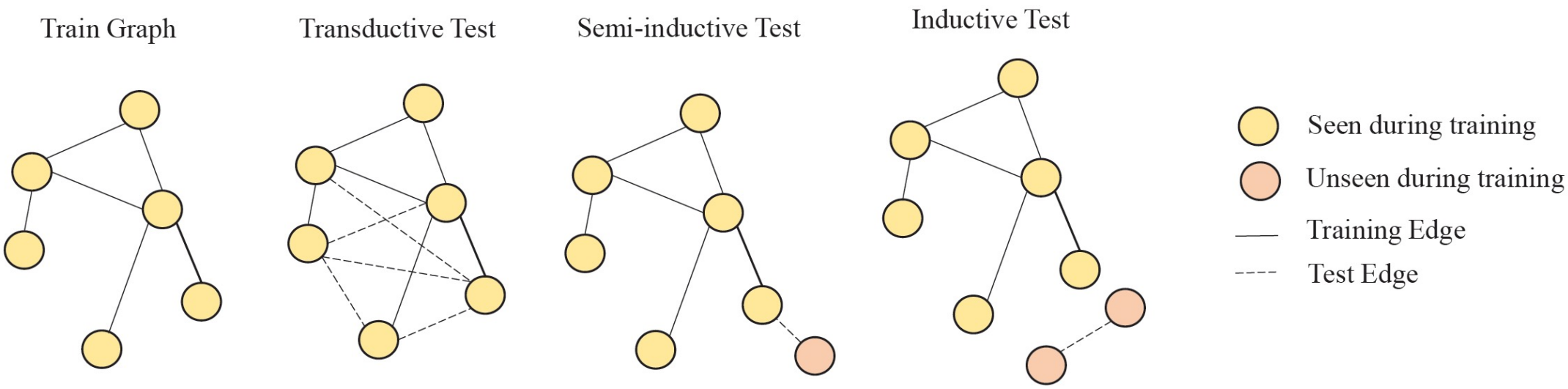
AI-Bind: Predicting bindings

Ayan Chatterjee

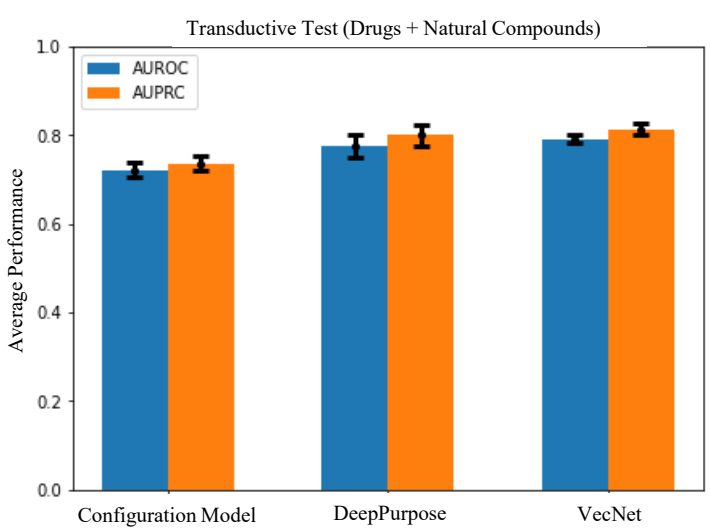


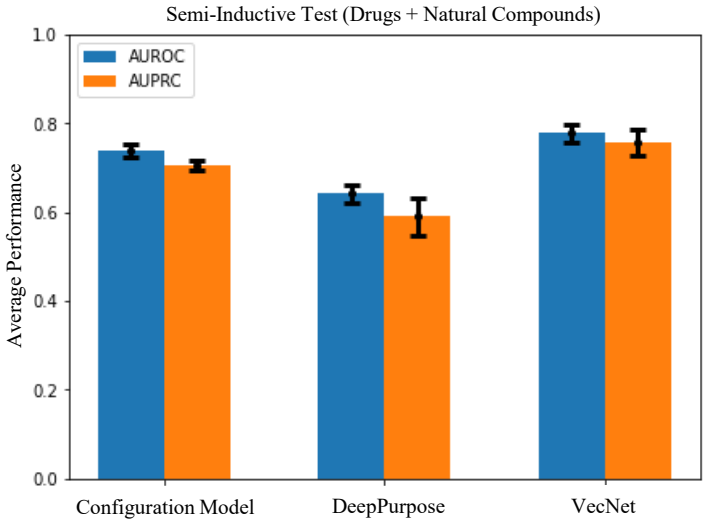
A. Chatterjee, O. Shafi Ahmed, R. Walters, Z. Shafi, D. Gysi, R. Yu, T. Eliassi-Rad, A.-L. Barabási, G. Menichetti. **AI-Bind: Improving Binding Predictions for Novel Protein Targets and Ligands**. *Nature Communications*, 2023 (forthcoming).

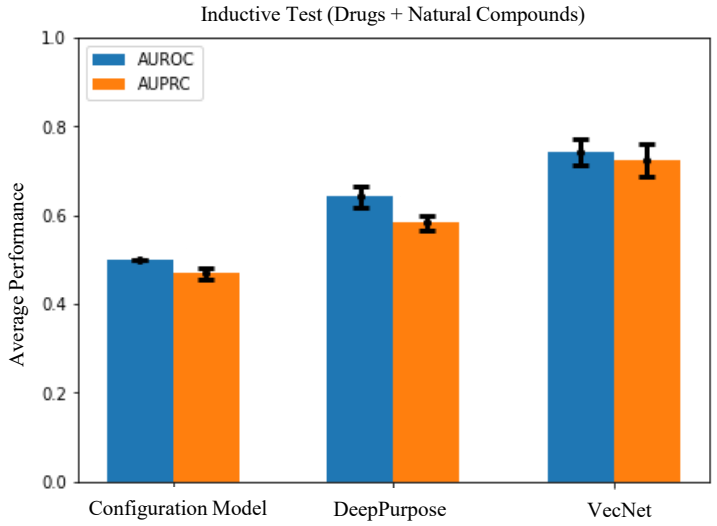
Three experimental settings



Experimental results







Configuration model performs well under transductive and semi-inductive settings.

* A. Chatterjee, O. Shafi Ahmed, R. Walters, Z. Shafi, D. Gysi, R. Yu, T. Eliassi-Rad, A.-L. Barabási, G. Menichetti. **AI-Bind: Improving Binding Predictions for Novel Protein Targets and Ligands**. *Nature Communications*, 2023 (forthcoming).

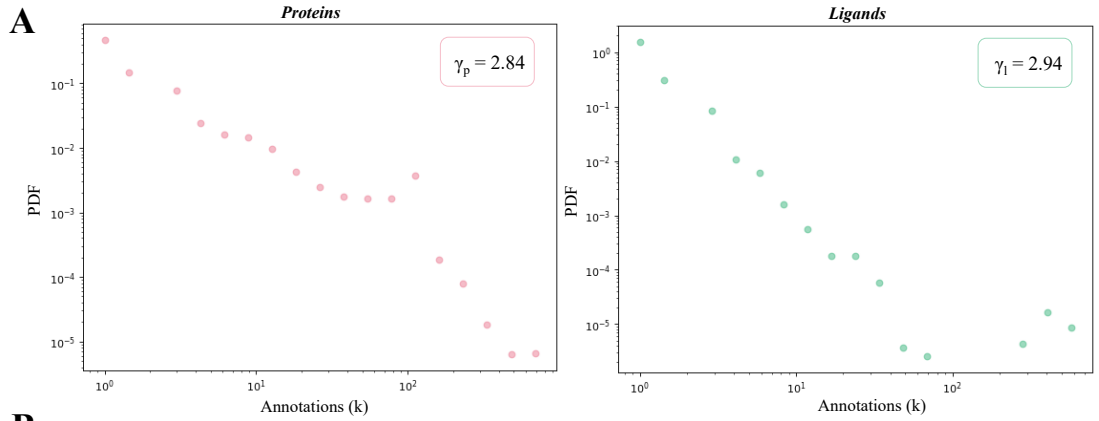
* K. Huang, et al. **DeepPurpose: A Deep Learning Library for Drug–target Interaction Prediction**. Bioinformatics (2020).

Question

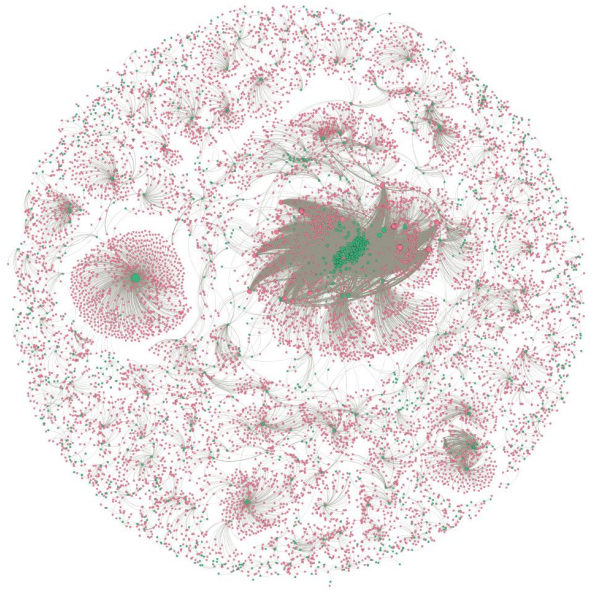
When do topological shortcuts occur?

We found …

Topological shortcuts occur when **annotation imbalance** prompts the ML models to leverage degree information (+ and − annotations) in making binding prediction instead of learning binding patterns from the molecular structures.



B

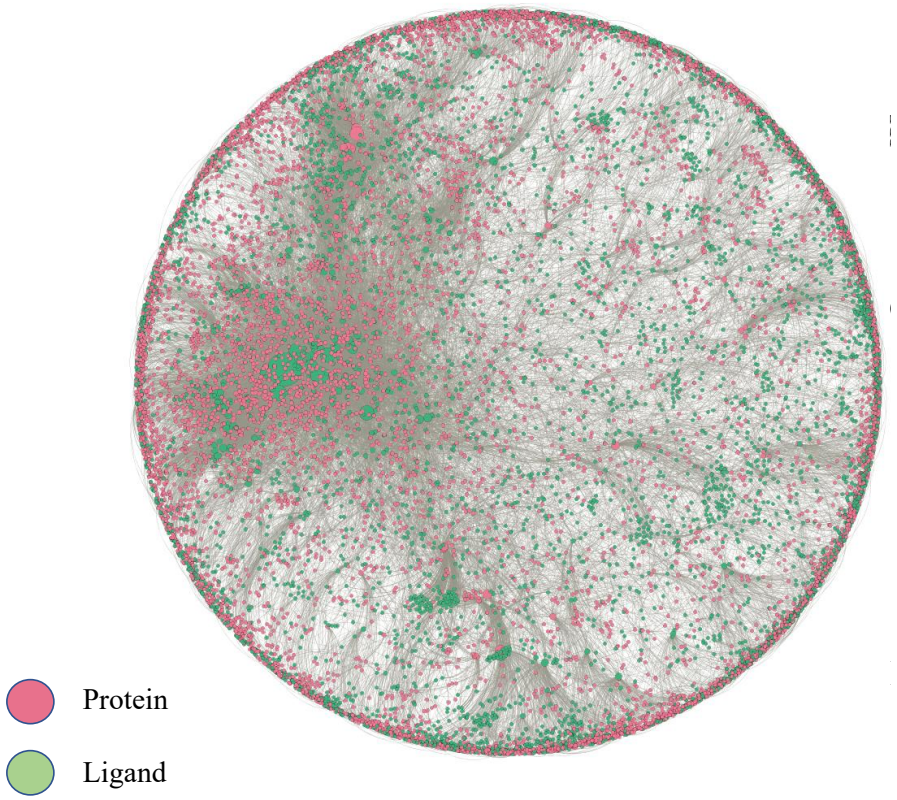


DTI Network Statistics	
Nodes	11,807
Proteins	1,391
Ligands	10,416
Positive Edges	5,591
Negative Edges	31,976

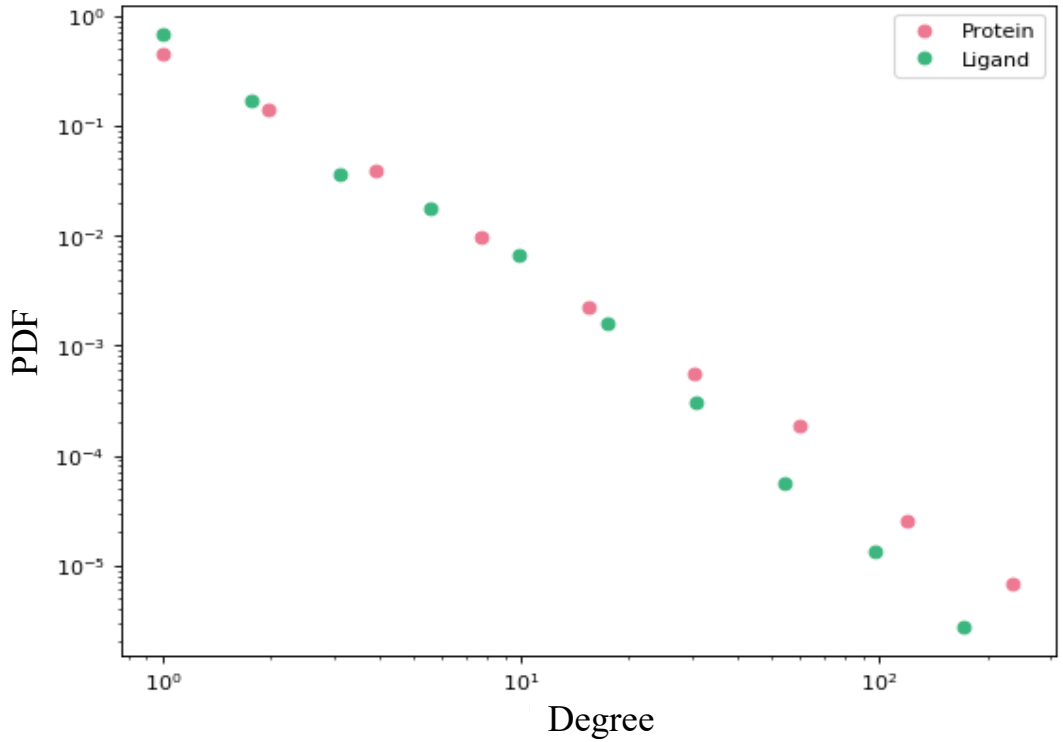
Protein
Ligand

Protein-ligands bipartite network and its degree distributions

Network consists of only binding (positive) annotations for drugs and natural compounds.

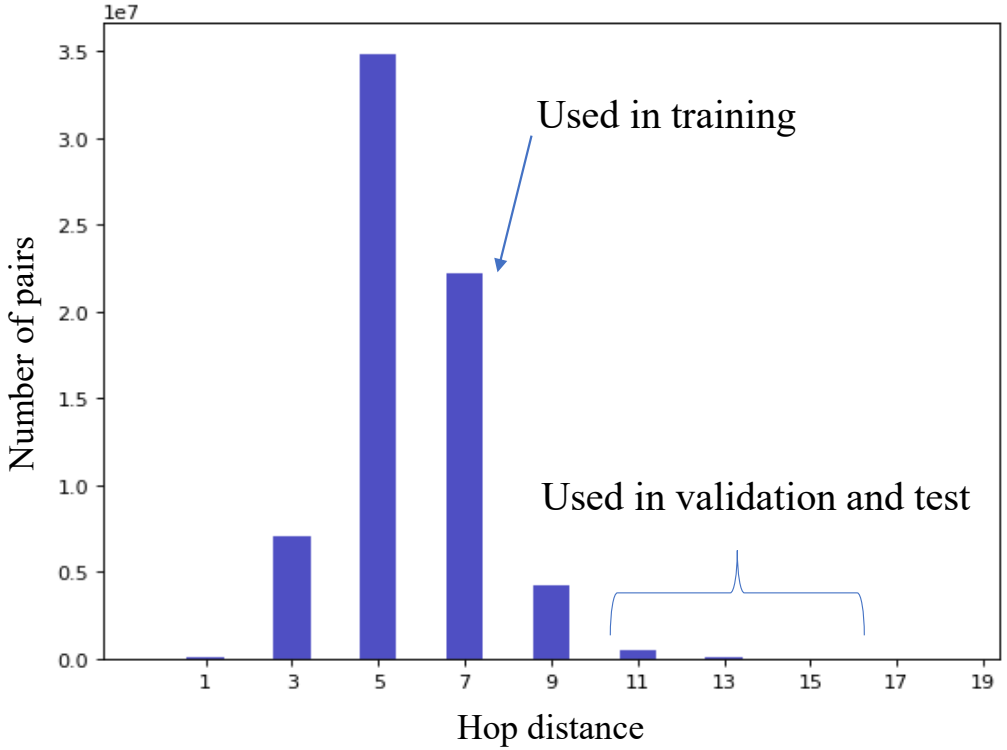


Degree distributions of ligands and proteins are fat-tailed in nature.

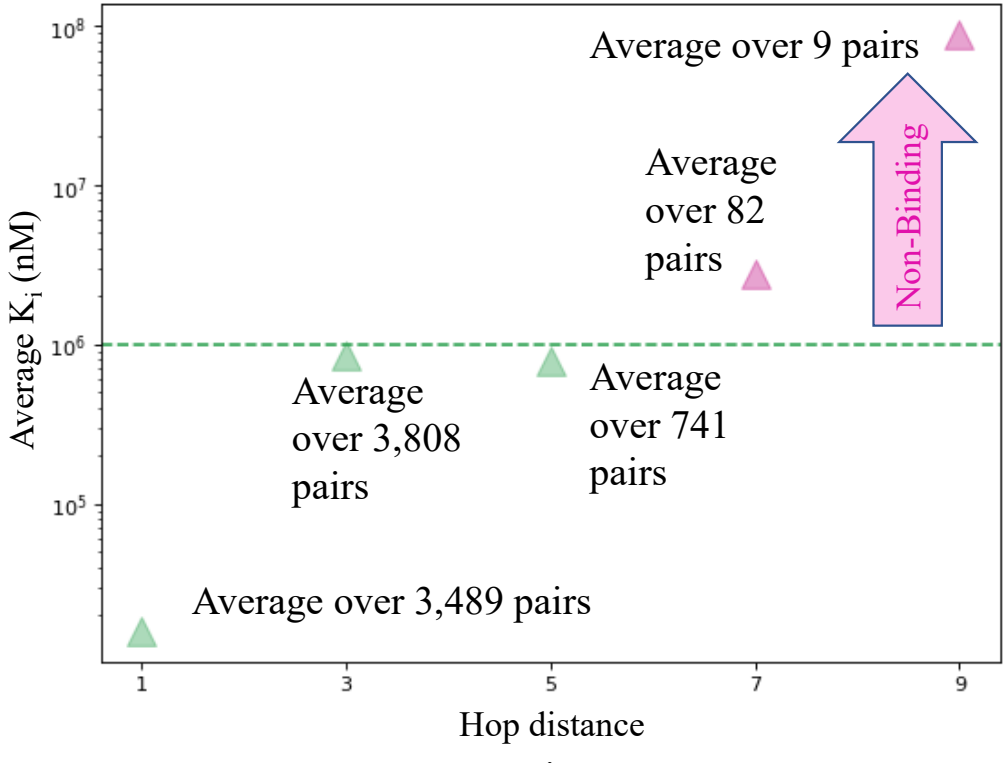


Network-derived negatives

Shortest-path length distribution capturing all possible protein-ligand pairs.

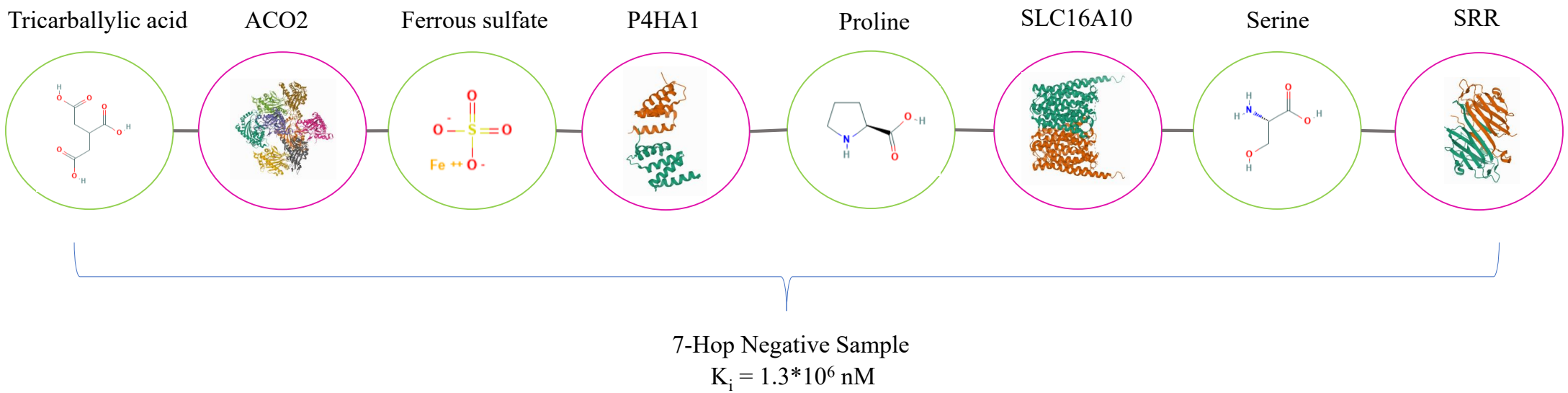


Average experimental kinetic constant as a function of the shortest path distance.

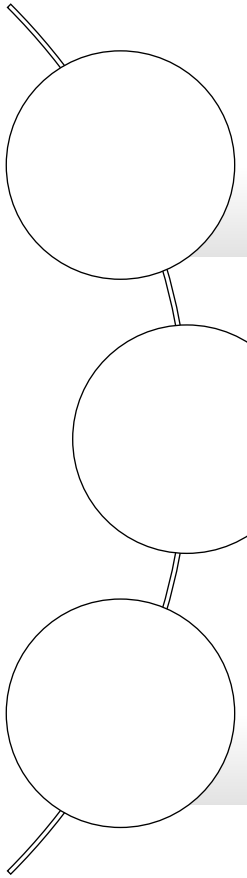


Network-derived negatives

An example of a protein-ligand pair which is 7 hops apart
and is used as a negative sample in the AI-Bind training set.



Outline



Instability of embeddings

Emergence of topological shortcuts

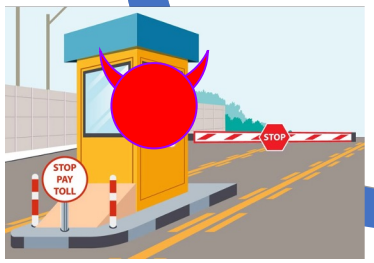
Adversarial ML

Man-in-the-Middle Attacks

Attacking shortest paths

populated area

popular destination



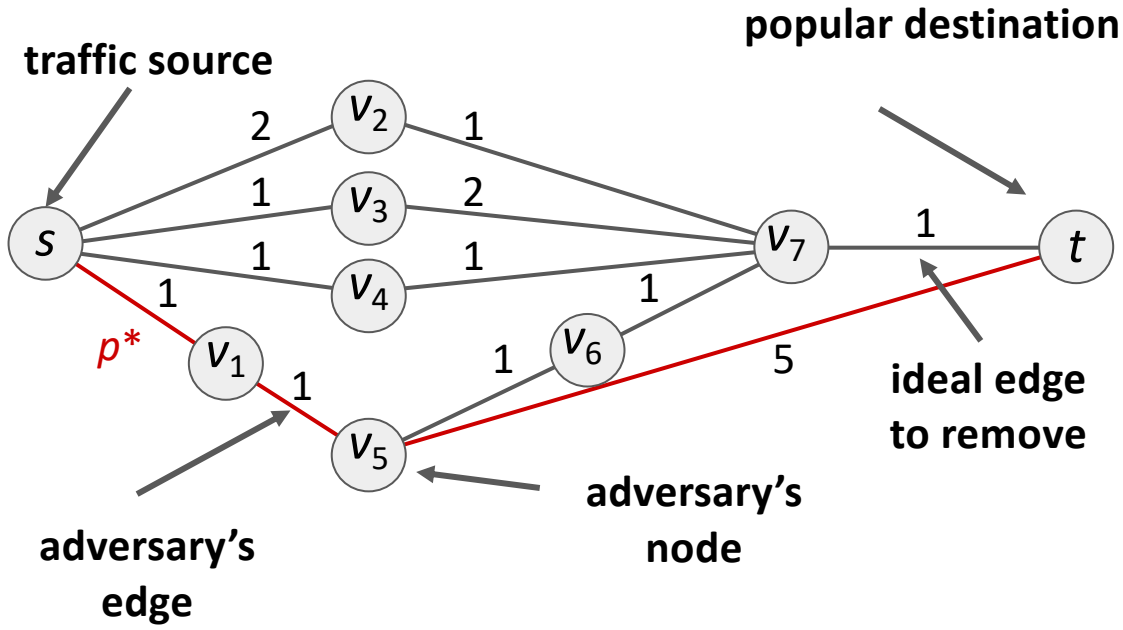
Can an adversary force a specific route to be shortest?

Problem: Force Path Cut

Given:

- a graph $G = (V, E)$
- edge weights (distances)
- edge removal costs
- a path p^* from node s to node t
- a budget b

can edges be removed with total cost less than b such that p^* is the shortest path from s to t?



Ben Miller

B.A. Miller, Z. Shafi, W. Ruml, Y. Vorobeychik, T. Eliassi-Rad, S. Alfeld. **PATHATTACK: Attacking Shortest Paths in Complex Networks**, In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (ECML PKDD), September 2021. Post-conference version is at https://arxiv.org/abs/2211.11141)

Metropolitan traffic systems

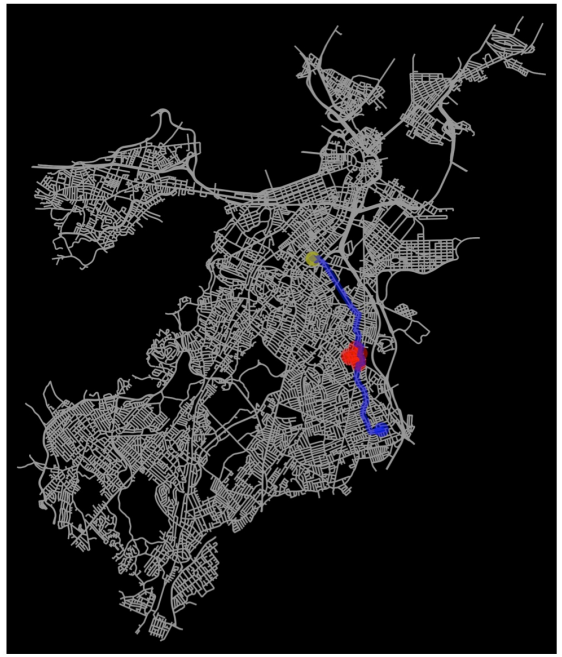


Fig. 1. A Boston experiment using Brigham and Women's hospital as the target destination, source was randomly selected. LENGTH is the weight type and WIDTH is the cost type.

Fig. 2. A San Francisco experiment using UCSF Medical Center at Mission Bay as the target destination, source was randomly selected. LENGTH is the weight type and WIDTH is the cost type.

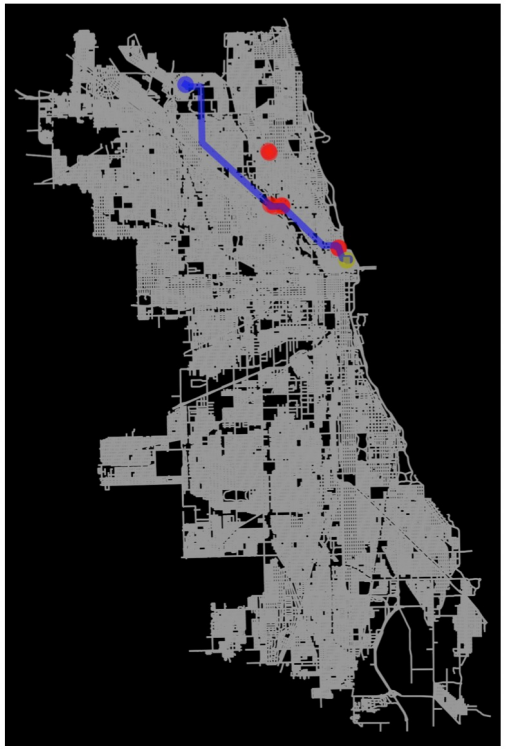


Fig. 3. A Chicago experiment using Northwestern hospital as the target destination, sources was randomly selected. LENGTH is the weight type and UNIFORM is the cost type.

Fig. 4. A Los Angeles experiment using LA Downtown Medical Center as the target destination, source was randomly selected. TIME is the weight type and LANES is the cost type.

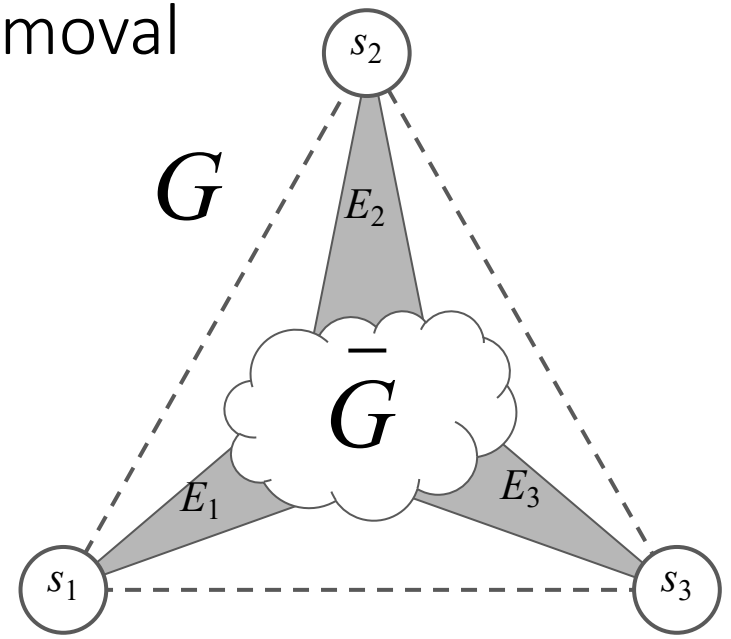
S. La Fontaine, N. Muralidhar, M. Clifford, T. Eliassi-Rad, C. Nita-Rotaru. Alternative Route- Based Attacks in Metropolitan Traffic Systems, In *The 8th International Workshop on Safety and Security of Intelligent Vehicles* (co-located with DSN2022: The 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks), Baltimore, MD, June 2022. https://cnitarot.github.io/papers/arb_ssiv2022.pdf

Questions

- Can an adversary efficiently find an optimal attack for a target path, edge, or node?
- Can approximate solutions be found more efficiently than optimal ones?
- Is there an observable tradeoff between time and the quality of the solution?

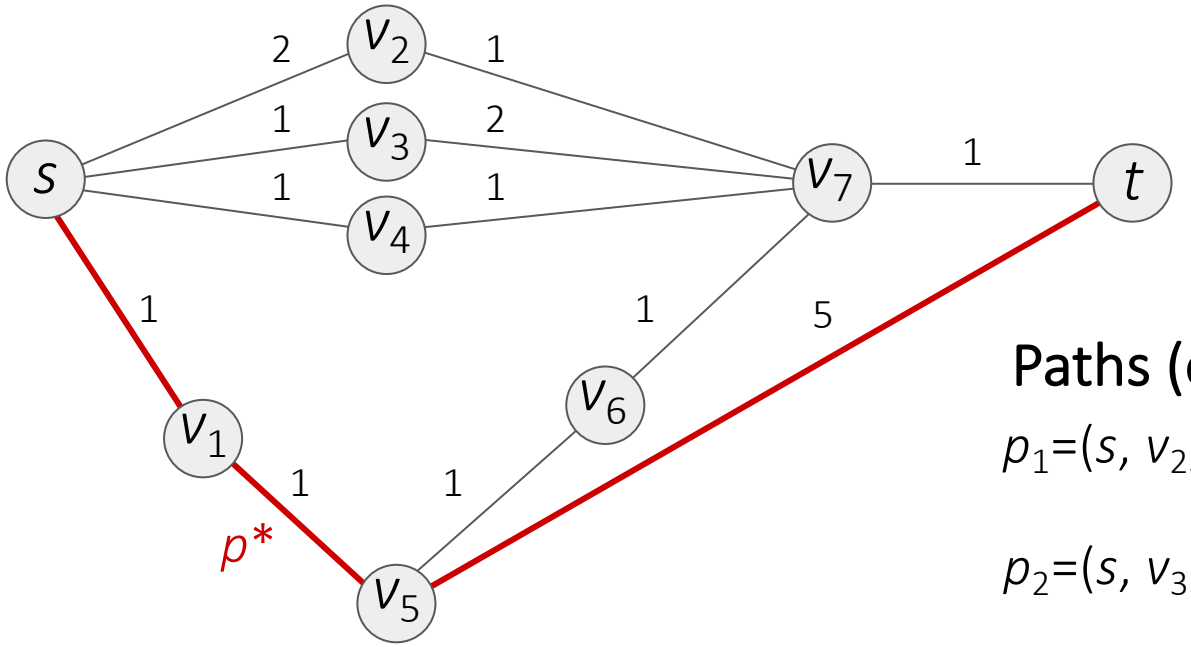
Theorem: Force path cut is APX-hard

- The 3-Terminal Cut problem is reducible to Force Path Cut
- 3-Terminal Cut: Given a graph and three terminal nodes, what is the minimum-cost set of edges whose removal disconnects the terminals?
- 3-Terminal Cut cannot be approximated in polynomial time within a factor of 1.2 (unless P=NP)



E. Dahlhaus et al., "The complexity of multiterminal cuts," *SIAM J. Computing*, vol. 23, no. 4, 1994

Force path cut as set cover



Paths (elements)

$p_1=(s, v_2, v_7, t)$

$p_2=(s, v_3, v_7, t)$

$p_3=(s, v_4, v_7, t)$

$p_4=(s, v_1, v_5, v_6, v_7, t)$

Edges (sets)

(s, v_2): $\{p_1\}$

(s, v_3): $\{p_2\}$

(s, v_4): $\{p_3\}$

(v_2, v_7): $\{p_1\}$

(v_3, v_7): $\{p_2\}$

(v_4, v_7): $\{p_2\}$

(v_7, t): $\{p_1, p_2, p_3, p_4\}$

(v_5, v_6): $\{p_4\}$

(v_6, v_7): $\{p_4\}$

cover an element ⇔ cut a path's edge

Approximation algorithms for set cover

Greedy Method

- Start with no sets selected
- While not all elements are covered
  - Select the set that contains the most uncovered elements

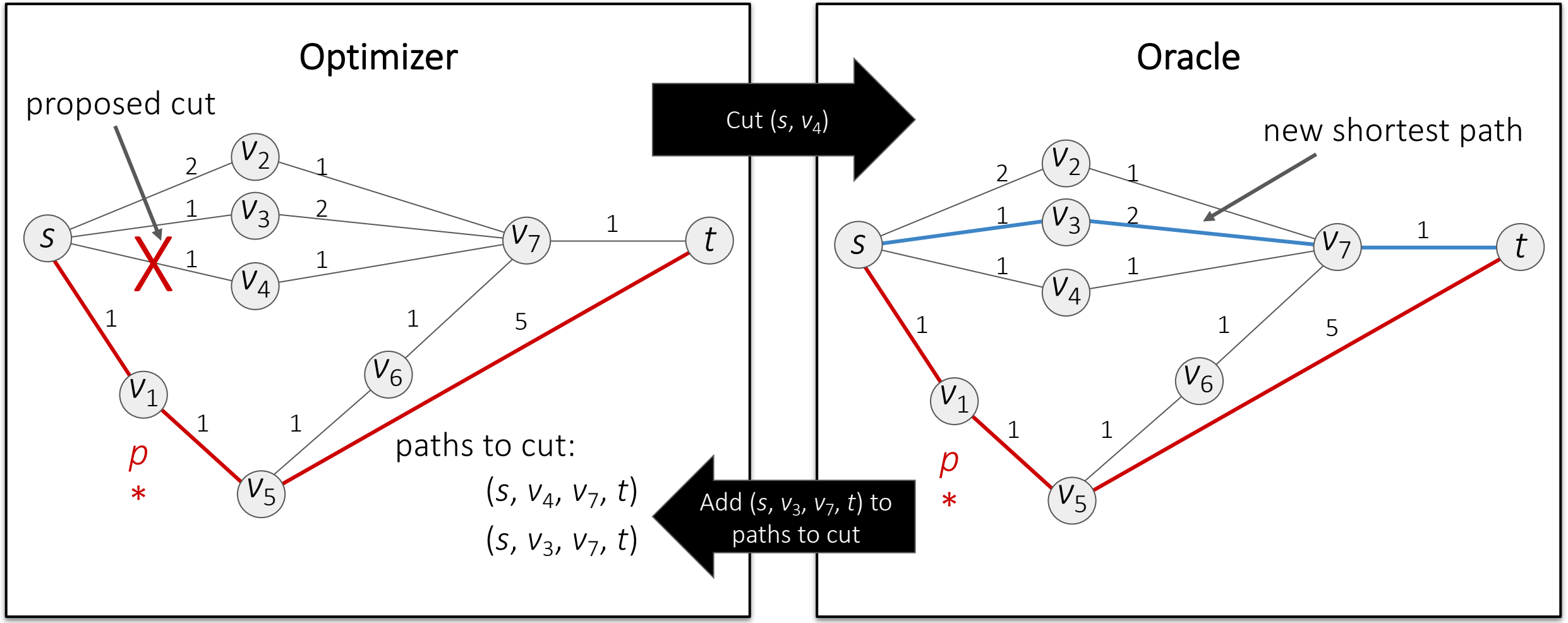
Linear Programming Method

- Formulate set cover as an integer program
- Relax integer constraints into reals
- Solve the linear program
- Obtain integer solution by randomized rounding

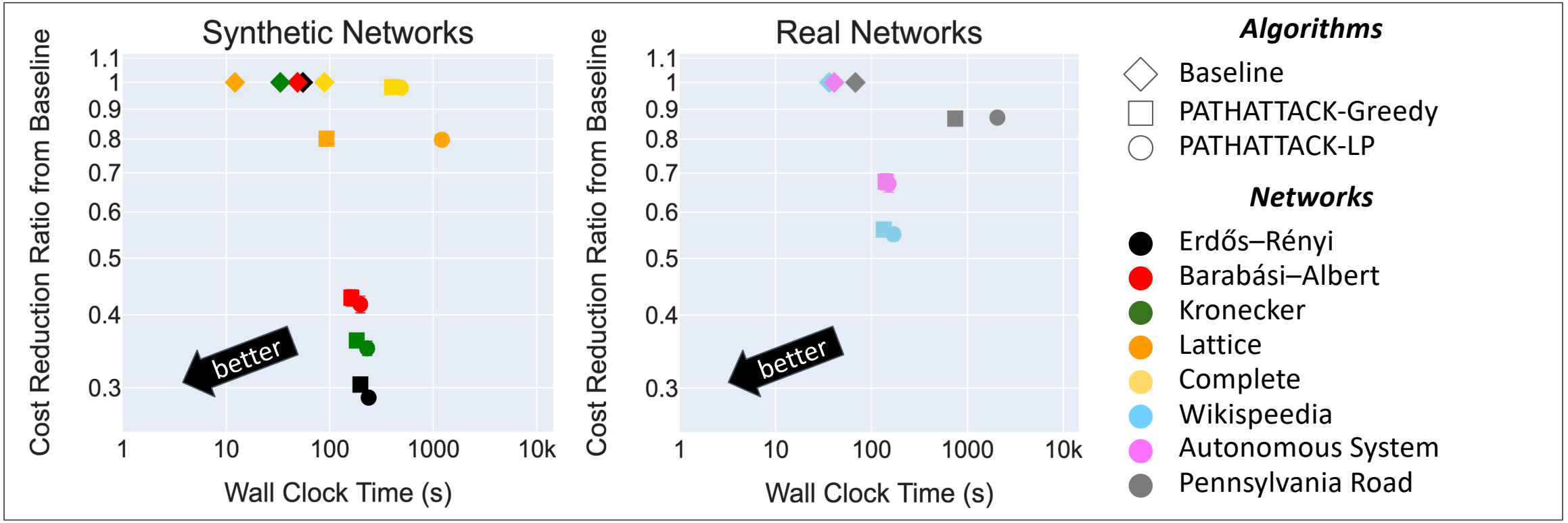
Both algorithms guarantee a solution at most a logarithmic factor larger than optimal.*

* V. V. Vazirani, *Approximation Algorithms*, Springer, 2003

PATHATTACK algorithm

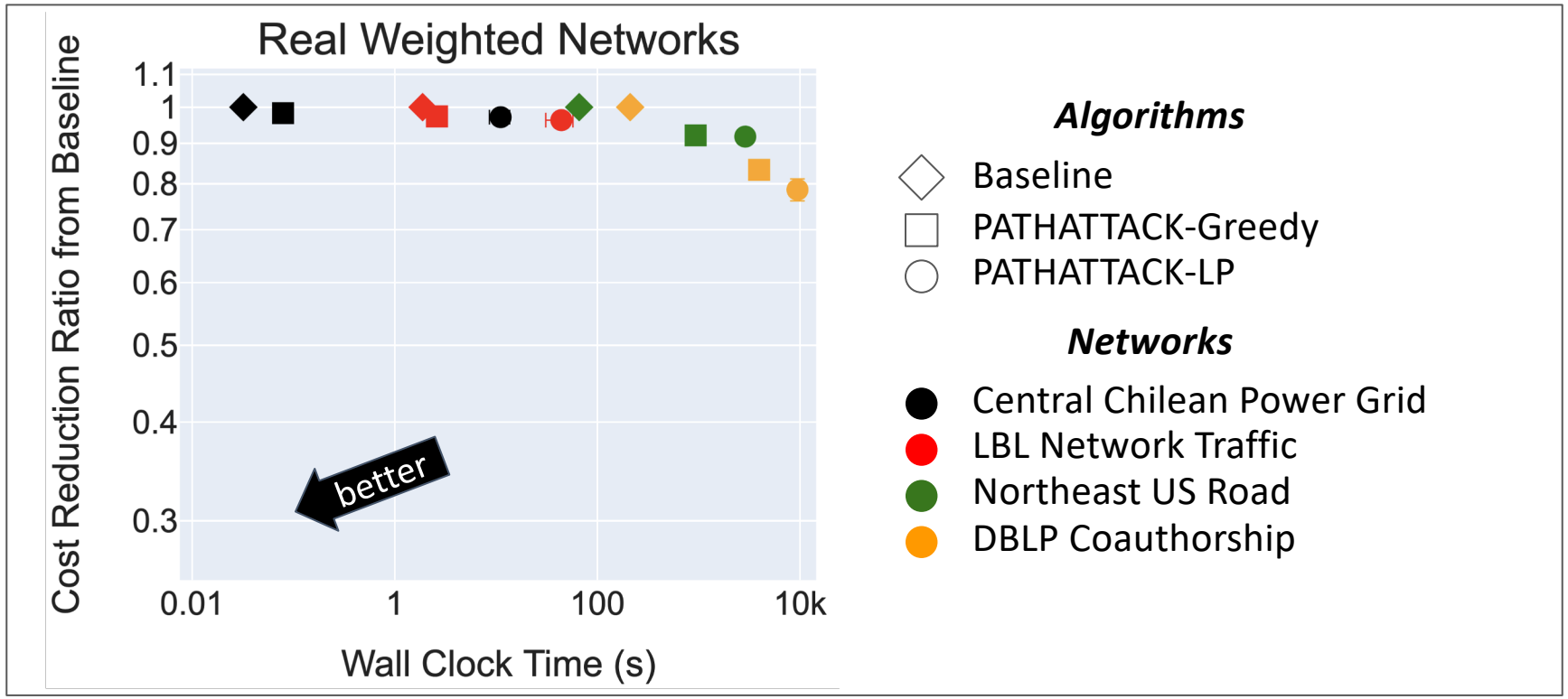


Results: Unweighted Graphs



- Both approximation algorithms yield similar cost
- Larger running time difference in grid-like networks

Results: Weighted Graphs



- Use natural weights as distances (invert if similarities)
- Again, smaller improvements for lattice-like graphs

ML-based optimization (work in progress)

- Learn a network representation
  - Node embedding is not as effective as edge or path embedding in this case
  - The instability issues persist
- Need constraint generation
  - Dijkstra's algorithm
  - A* (heuristic defined by embeddings)
  - Neural algorithmic framework (needs tailoring)
- Replace optimization module with either a SL model to predict which edges to remove or a RL model to learn an optimal policy of removing edges
  - As always, the RL model is harder to learn (has a big state space, needs lots of training, *etc*)

We found …

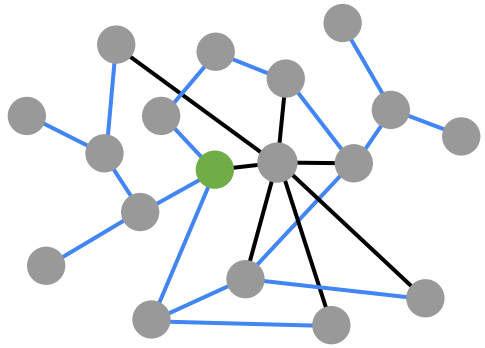
1. Can we solve Force Path Cut in polynomial time? ✗ **No, it's APX-hard**
2. Can we find an approximate solution more efficiently than an exact one? ✓ **Yes, and the linear programming method found the optimal solution in over 98% of our experiments**
3. Is there a tradeoff between running time and performance among approaches? ✓ **Yes, empirically (improvement for additional time is reduced in lattice-like graphs)**

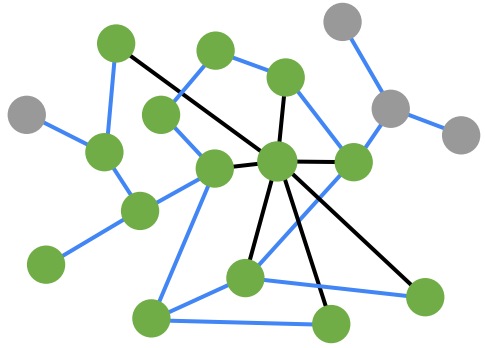
Targeted Diffusion

Diffusion processes

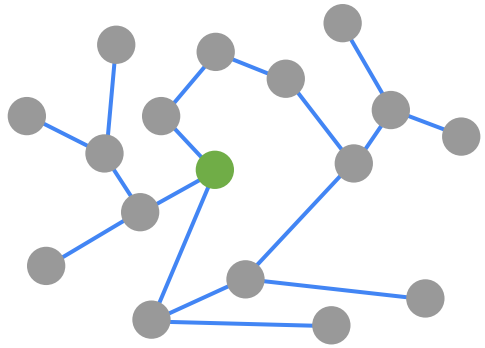
$t = 0$

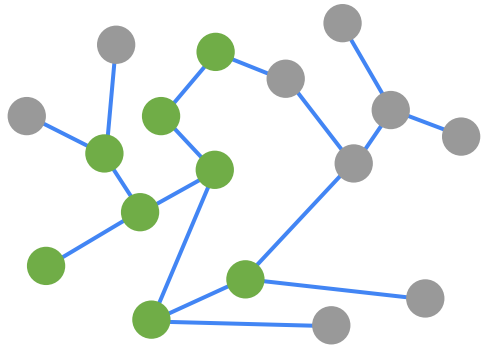
$t \to \infty$





More infected





Fewer infected

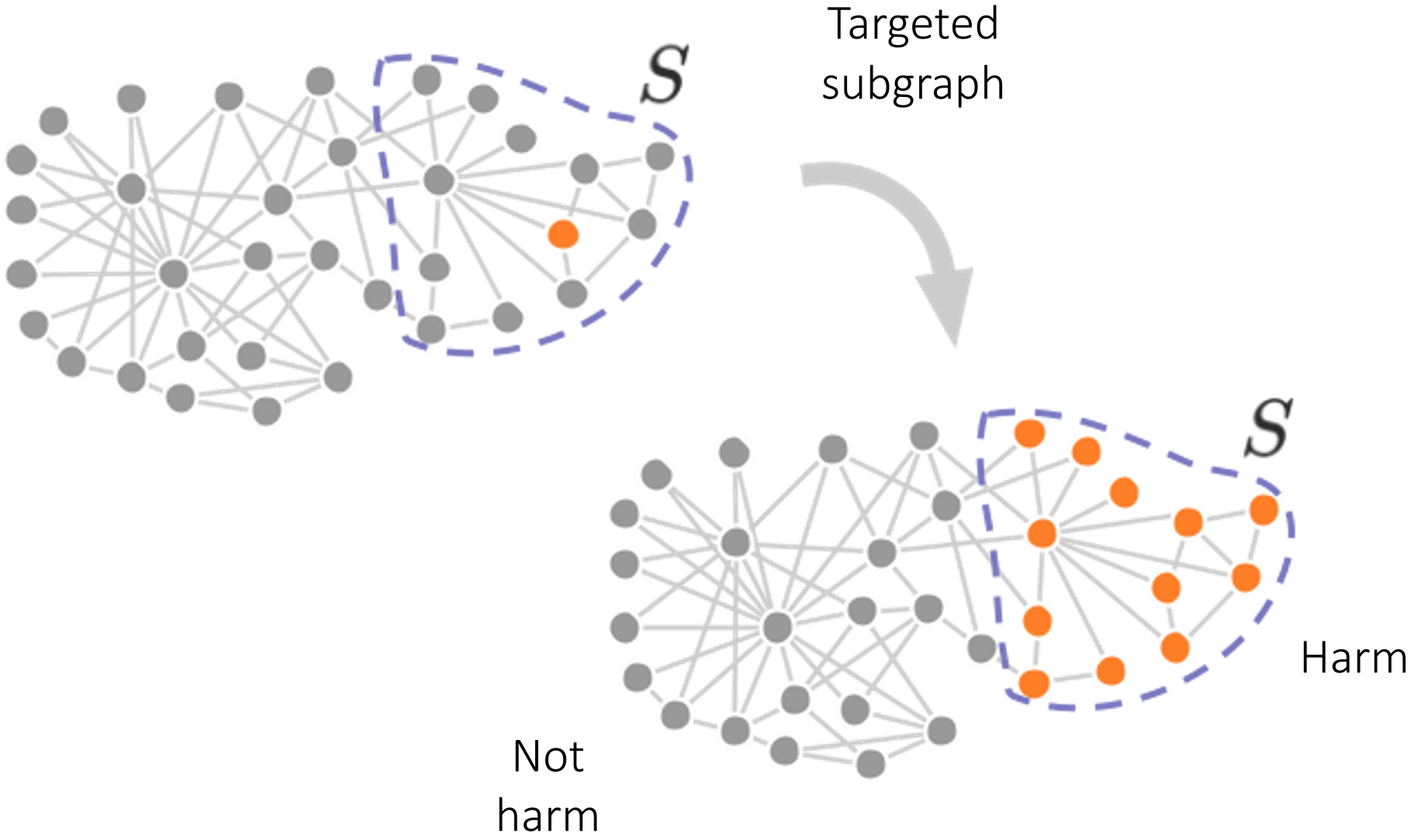
Question

How can an attacker manipulate the structure of a graph to increase or decrease the impact of diffusion?

Targeted diffusion

Targeted subgraph

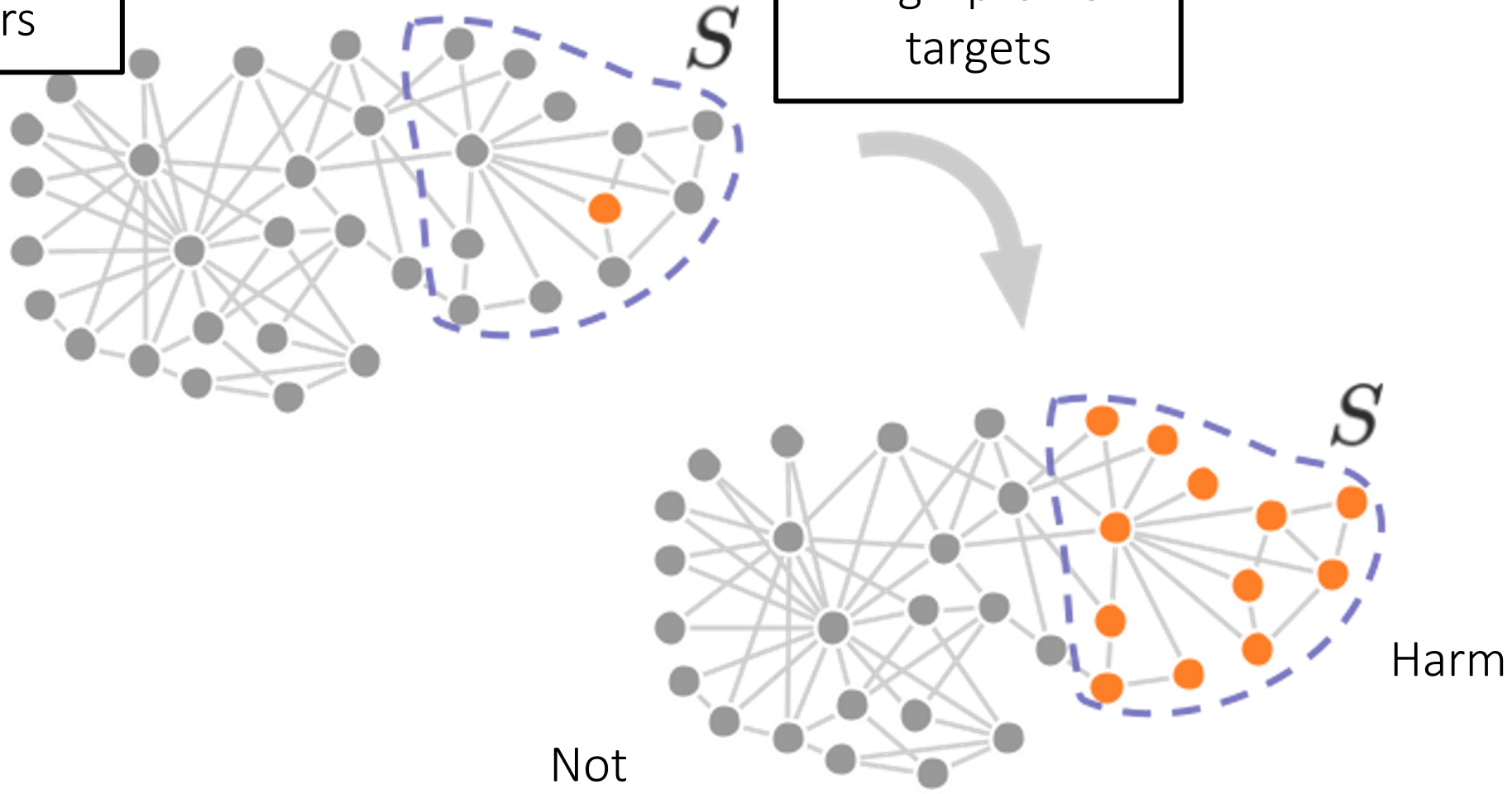
Targeted diffusion



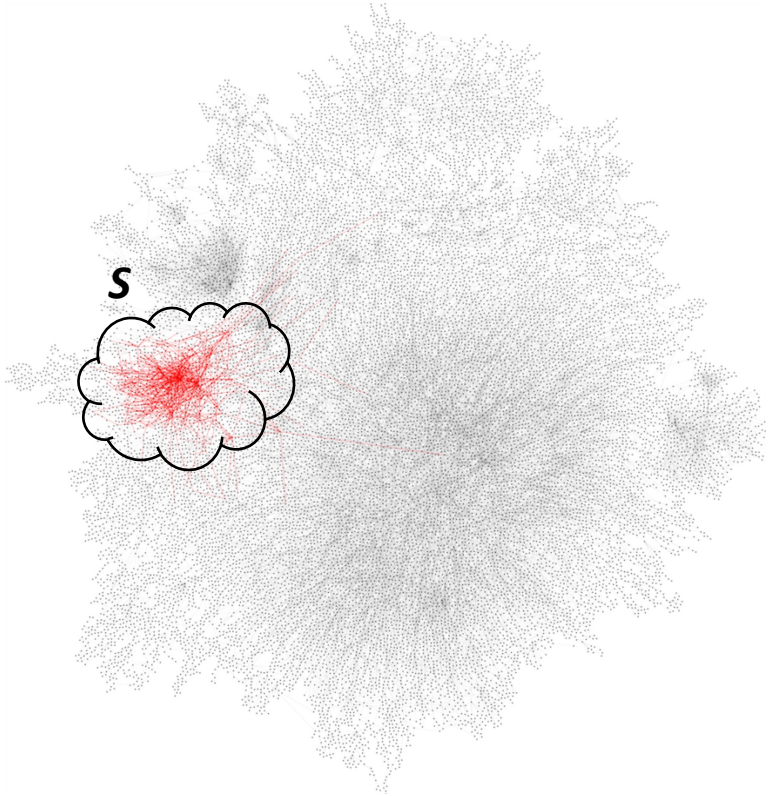
Targeted diffusion in cybersecurity

Computers

High-profile targets



Targeted diffusion



A malicious actor, 😈, can modify the graph structure $G = (V, E)$ to achieve two goals

1. Maximum the diffusion spread to a target subgraph S
2. Minimize the impact on the remaining graph $G \backslash S$

Sixie Yu

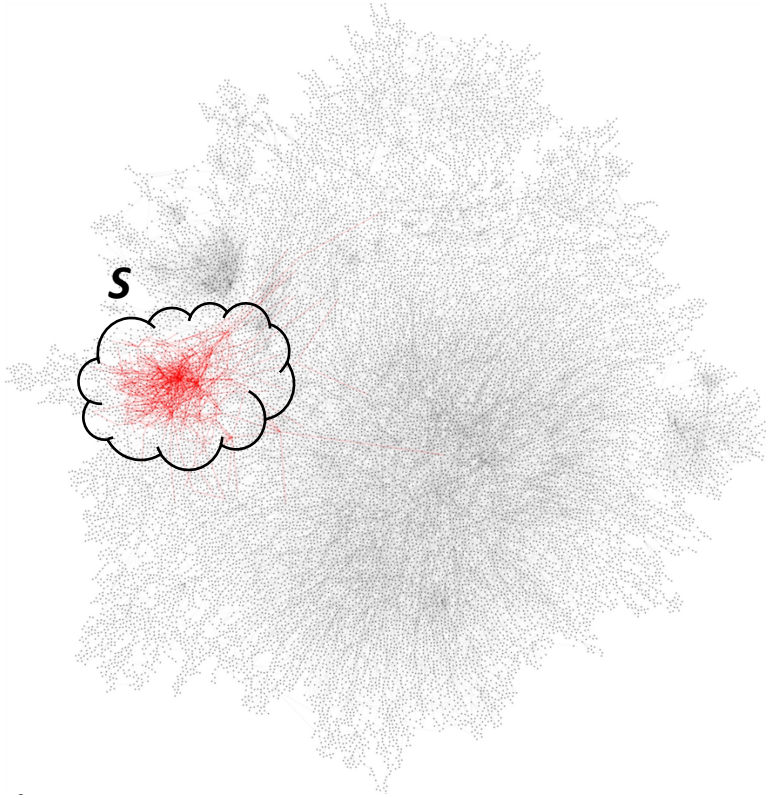
Leo Torres

S. Yu, L. Torres, S. Alfeld, T. Eliassi-Rad, Y. Vorobeychik. **POTION: Optimizing Graph Structure for Targeted Diffusion**. In *Proceedings of the 2021 SIAM Data Mining Conference* (SDM), May 2021, pp. 154-162.

Setup

- $G = (V, E)$ is a connected, weighted or unweighted, undirected graph with no self-loops
- $n = |V|$ = number of nodes in *G*
- *A* = adjacency matrix of *G*
- Eigenvalues of *A*: $\lambda_1(A) \geq \cdots \geq \lambda_n(A)$

Problem definition

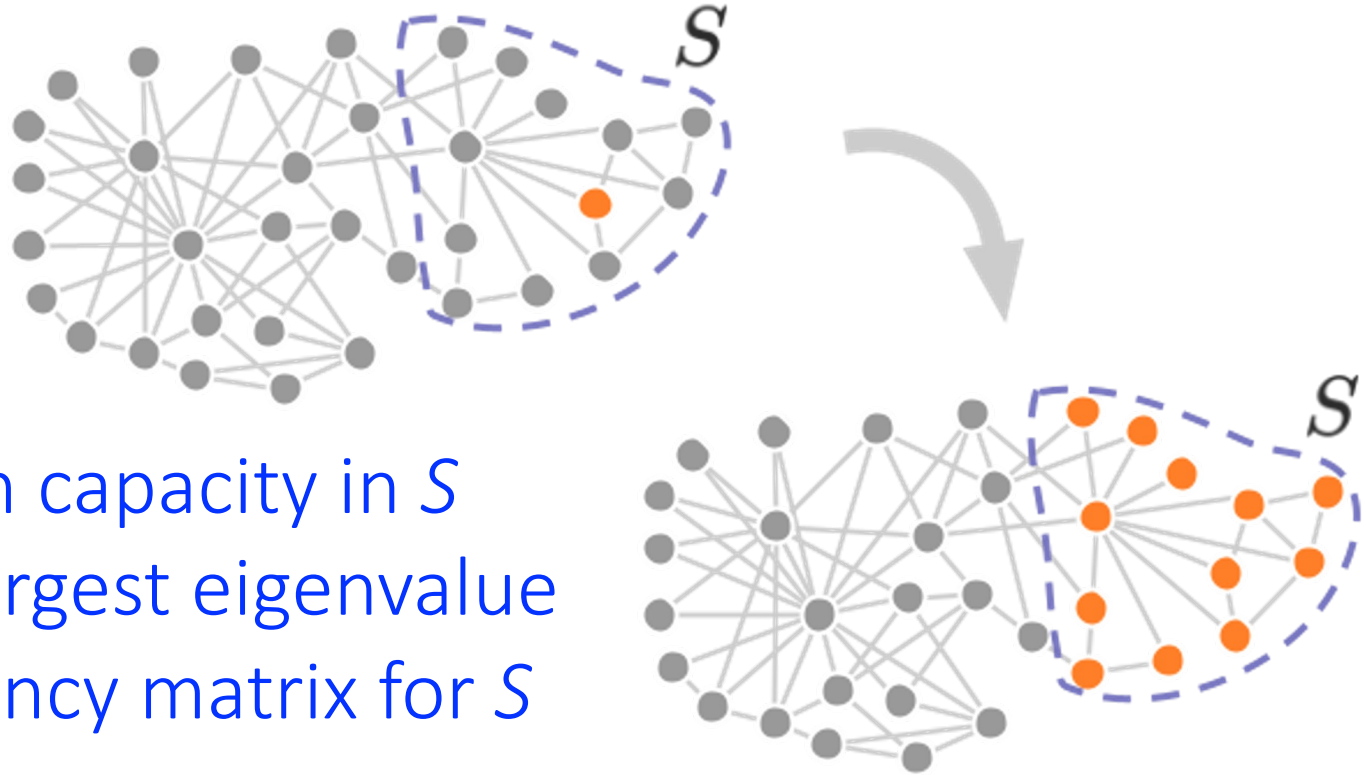


- 😈 targets the subgraph S
  - A_S = adjacency matrix of S
- Let $S' = G \backslash S$
- Attacker modifies the structure of *G*, which results in

$$\tilde{A} = A + \Delta = \tilde{A}_S + \tilde{A}_{S'}$$

Attacker's objectives

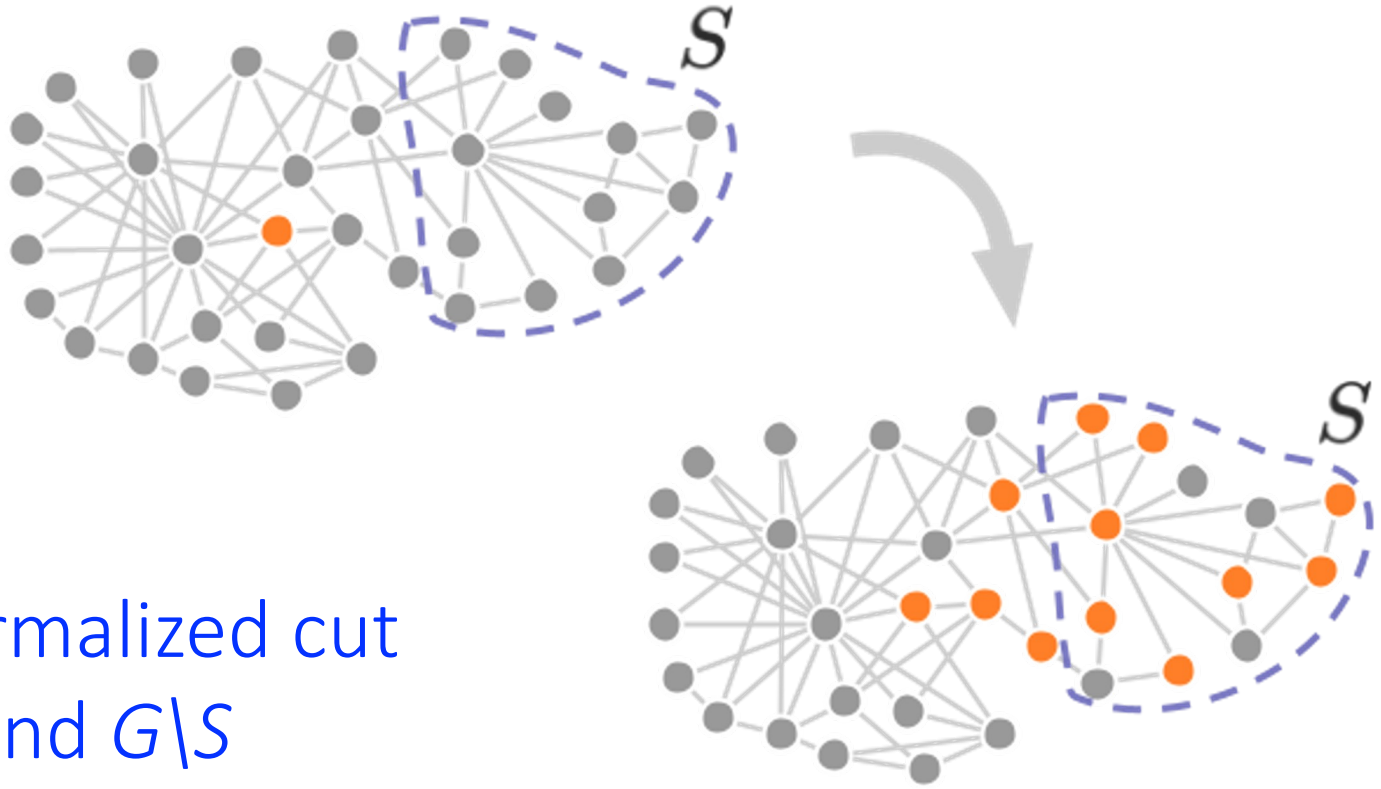
1. If the virus starts in *S*, it should create an epidemic.



Increase path capacity in *S*
≡ increase largest eigenvalue
of the adjacency matrix for *S*

Attacker's objectives

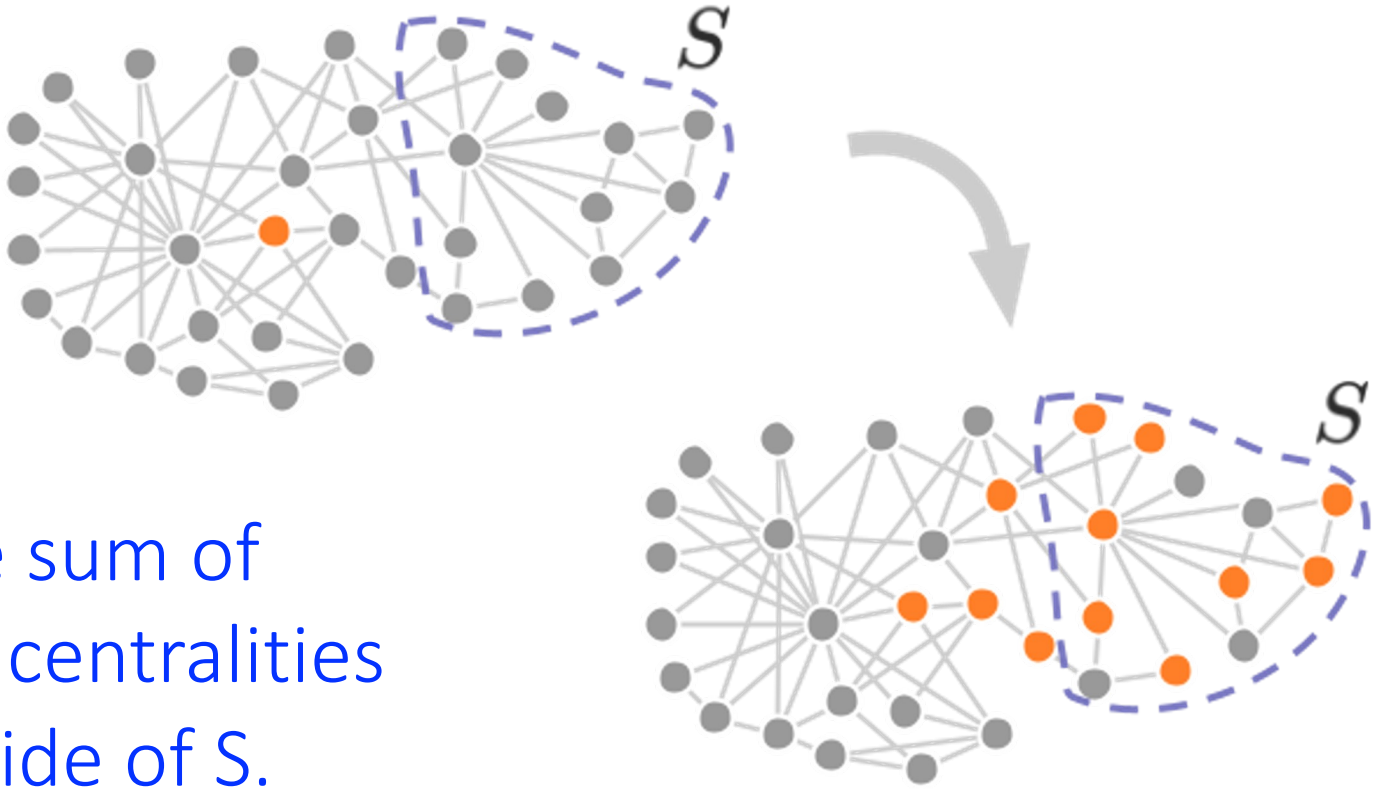
2. If the virus starts in $G \backslash S$, it should reach S.



Increase normalized cut between S and $G \backslash S$

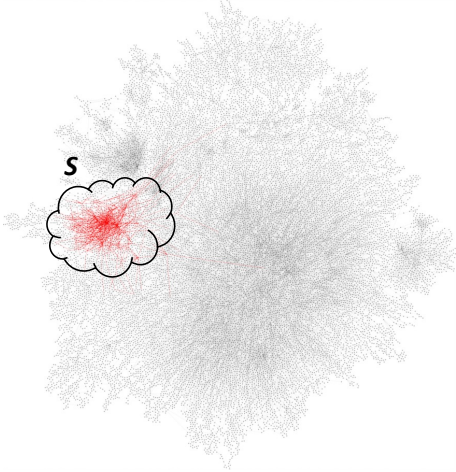
Attacker's objectives

3. Limit the harm to *G/S.*



Increase the sum of eigenvector centralities of nodes inside of S.

POTION formulation



path capacity in S

eigenvector centrality of S

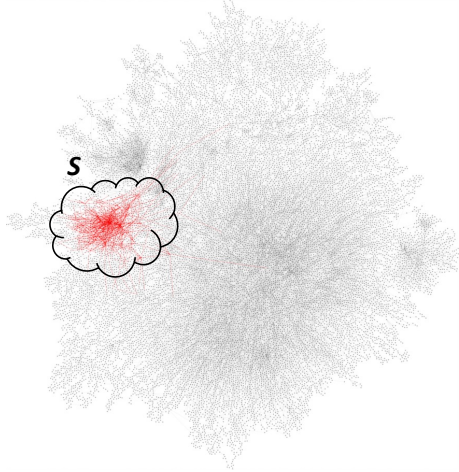
normalized cut between S and S'

$$\max_{\tilde{\boldsymbol{A}}} \quad \alpha_1 \lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}}) + \alpha_2 \sigma(\mathcal{S}) + \alpha_3 \phi(\mathcal{S})$$

$$s.t. \qquad \tilde{\boldsymbol{A}} \in \mathcal{P} = \left\{ \tilde{\boldsymbol{A}} \;\middle|\; \begin{array}{l} |\lambda_i(\tilde{\boldsymbol{A}}) - \lambda_i(\boldsymbol{A})| \leq \epsilon,\ i = 1, \ldots, n, \\ \tilde{\boldsymbol{A}} = \tilde{\boldsymbol{A}}^\top, \tilde{\boldsymbol{A}}_{ii} = 0,\ \forall i = 1, \ldots, n \end{array} \right\},$$

where the relative importance of the terms is balanced by the nonnegative constants $\alpha_1, \alpha_2, \alpha_3$, and the restrictions $\tilde{\boldsymbol{A}} = \tilde{\boldsymbol{A}}^\top$ and $\tilde{\boldsymbol{A}}_{ii} = 0, \forall i = 1, \ldots, n$ ensure that $\tilde{\boldsymbol{A}}$ is a valid adjacency matrix.

POTION-ALG



Rayleigh quotient + the Power method
Intuitively, argmax approx. a sequence of diff. operations

A differentiable function
involving the Laplacian matrix

$$\max_{\tilde{\boldsymbol{A}}} \quad \alpha_1 \lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}}) + \alpha_2 \sigma(\mathcal{S}) + \alpha_3 \phi(\mathcal{S})$$

$$s.t. \qquad \tilde{\boldsymbol{A}} \in \mathcal{P} = \left\{ \tilde{\boldsymbol{A}} \;\middle|\; \begin{array}{l} |\lambda_i(\tilde{\boldsymbol{A}}) - \lambda_i(\boldsymbol{A})| \leq \epsilon,\ i = 1, \ldots, n, \\ \tilde{\boldsymbol{A}} = \tilde{\boldsymbol{A}}^\top, \tilde{\boldsymbol{A}}_{ii} = 0,\ \forall i = 1, \ldots, n \end{array} \right\},$$

where the relative importance of the terms is balanced by the nonnegative constants $\alpha_1, \alpha_2, \alpha_3$, and the restrictions $\tilde{\boldsymbol{A}} = \tilde{\boldsymbol{A}}^\top$ and $\tilde{\boldsymbol{A}}_{ii} = 0, \forall i = 1, \ldots, n$ ensure that $\tilde{\boldsymbol{A}}$ is a valid adjacency matrix.

Pseudospectra theory: equivalent to compute $\lambda_1(\tilde{\boldsymbol{A}} - \boldsymbol{A})$

Power method

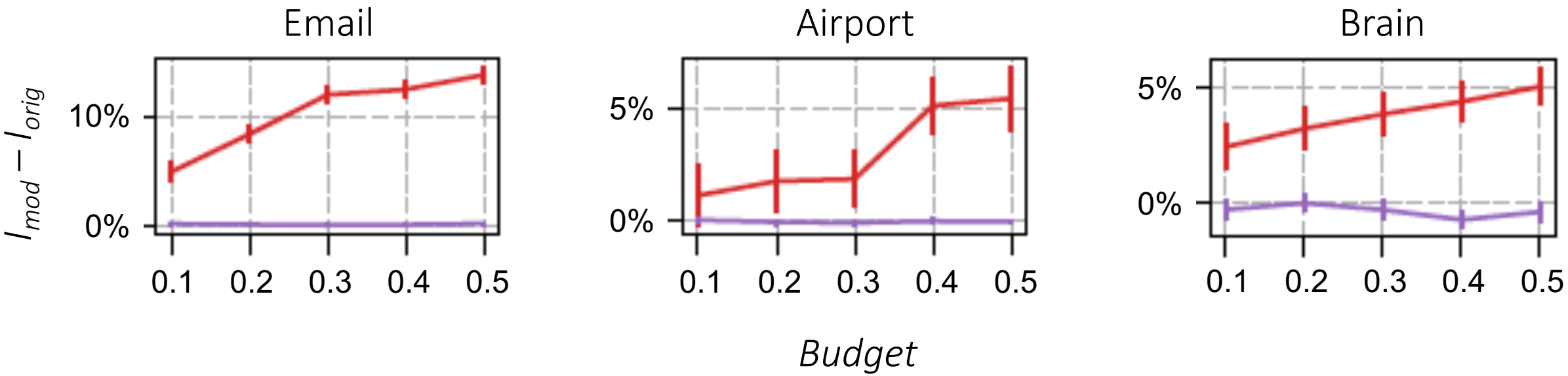
POTION-ALG optimizes 😈's the objective function

- Projected gradient ascent has two major issues:
  1. The objective function involves terms that do not have an explicit functional representation in the decision variables
  2. The projection step is quite expensive, as it involves projecting into a spectral norm ball, which entails an expensive SVD operation
- We leverage Rayleigh quotients and pseudospectrum theory to overcome these hurdles and have a differentiable function of the adjacency matrix

Algorithm 1 POTION-ALG

Input: $\boldsymbol{A}, \epsilon, \{\eta_i\}_{i=1}$ ▷ $\{\eta_i\}_{i=1}$ is a schedule of step sizes
Initialize: $i = 1, \tilde{\boldsymbol{A}}_1 = \boldsymbol{A}, B_1 = 0$ ▷ B_i: the amount of budget used just before step i
while True **do**
  Set $\boldsymbol{\Delta}_i$ to the gradient of $\alpha_1\lambda_1(\tilde{\boldsymbol{A}}_{\mathcal{S}}) + \alpha_2\sigma(\mathcal{S}) + \alpha_3\phi(\mathcal{S})$ w.r.t. to $\tilde{\boldsymbol{A}}_i$
  Set the diagonal entries of $\boldsymbol{\Delta}_i$ to zeros
  if $\|\boldsymbol{\Delta}_i\| = \mathbf{0}$ **then** ▷ a local optimum is found
    return $\tilde{\boldsymbol{A}}_i$
  end if
  if $B_i + \|\eta_i\boldsymbol{\Delta}_i\|_2 \leq \epsilon$ **then** ▷ one-step look ahead
    $\tilde{\boldsymbol{A}}_{i+1} = \tilde{\boldsymbol{A}}_i + \eta_i\boldsymbol{\Delta}_i$, $B_{i+1} = B_i + \|\eta_i\boldsymbol{\Delta}_i\|_2$, $i = i+1$
  else
    return $\tilde{\boldsymbol{A}}_i$
  end if
end while

POTION effectively achieves targeted diffusion in G_S without affecting $G_{S'}$



	Email	Airport	Brain
#nodes	986	1572	638
#edges	16064	17214	18625

Diffusion dynamics: SIS

POTION is applicable to various epidemic processes

- SIS
- SIR
- SEIR
- Random walk based spreading dynamics

Necessary condition for successful attacks in the form of a lower bound on the attacker's budget ϵ

THEOREM 5.1. *Given an instance* `TargetDiff`$(\mathcal{S}, G, \epsilon)$, $I(G_{\mathcal{S}})$ *is estimated by* $\hat{I}(G_{\mathcal{S}}) = \sum_{i \in \mathcal{S}} 1 - \delta/(\beta d_i)$. *Suppose we have an upper bound* $|\hat{I}(G_{\mathcal{S}}) - I(G_{\mathcal{S}})| \leq \tau$, *the degrees of nodes in* $\mathcal{S}$ *are increased, i.e.,* $\tilde{d}_i \geq d_i$ *for* $i \in \mathcal{S}$, *and* $\delta/\beta \leq d_{min}$. *In order to have* $I(\tilde{G}_{\mathcal{S}}) - I(G_{\mathcal{S}}) > 2\tau$, *the budget* ϵ *must satisfy:*

$$\epsilon \geq \sqrt{\frac{|\mathcal{S}|}{n}} \left(\frac{\sum_{i \in \mathcal{S}} d_i^2}{|S|} - \frac{\left(\sum_{i \in \mathcal{S}} d_i\right)^2}{|S|^2} \right)^{1/2}. \tag{5.12}$$

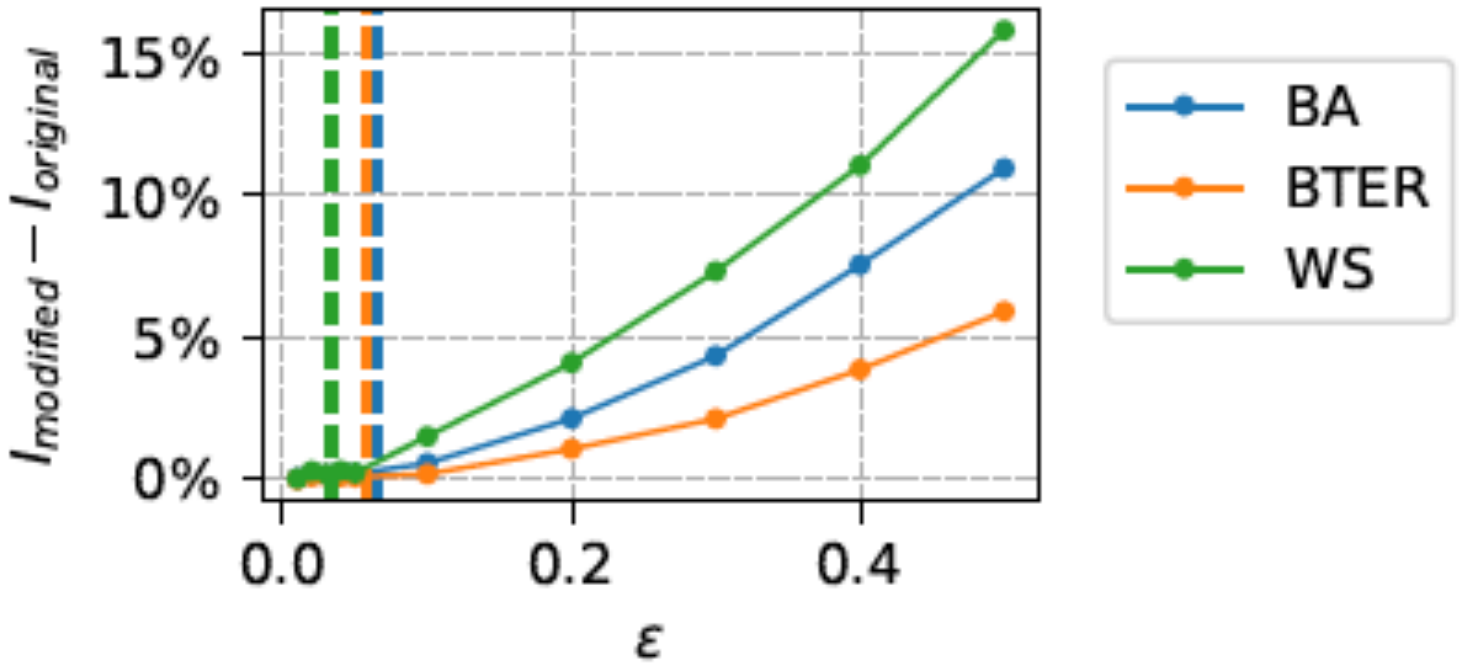
$I(G_S)$ = *Impact on* G_S

β = *attack probability over a communication*

δ = *healing probability once infected*

To have epidemic spread: $\lambda \geq \frac{\delta}{\beta}$

Certified robustness results

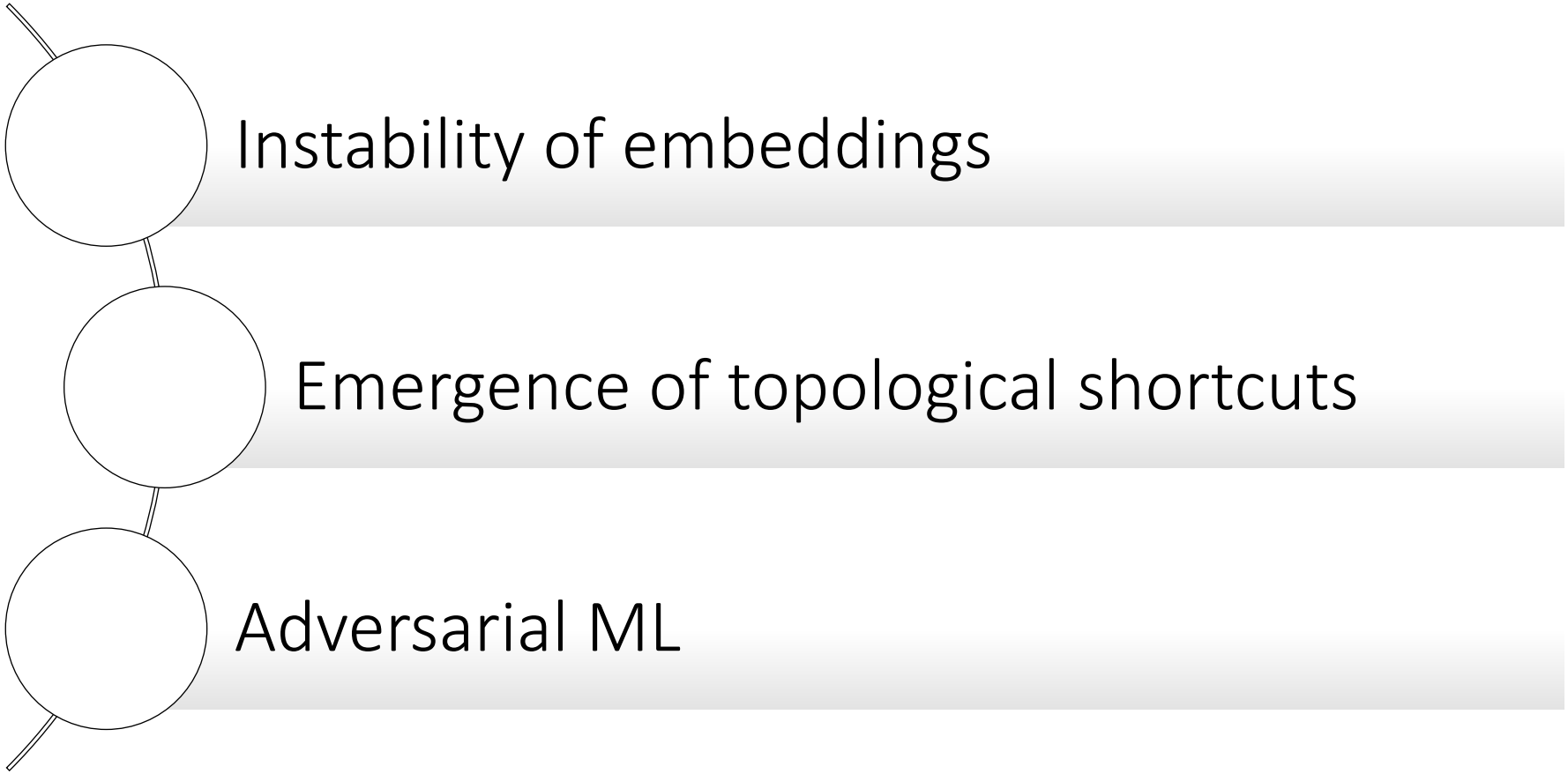


- Dashed lines mark the lower bounds from Eq. (5.12)
- Solid lines represent infectious ratios within target subgraphs.

We found …

- POTION: a model for targeted diffusion attack by optimizing graph structures
- POTION-ALG: an efficient algorithm for solving POTION by leveraging Rayleigh quotient and pseudospectra theory
- Provided a condition for certifying that a targeted subgraph is immune to such attacks
- Demonstrated effectiveness of POTION and POTION-ALG on synthetic and real-world networks

Three pitfalls of using ML-based optimization



Thank you!

Any questions?

Slides at http://eliassi.org/tina-ipam-aid23.pdf