Talk @ ETRI
January 22, 2020

Graph Representation Learning

Sungsu Lim
Chungnam National University

Brief Bio

Sungsu Lim

Assistant Professor
Data Intelligence Lab.
Computer Science and Engineering
Chungnam National University

Web: http://cnudi.com/
E-mail: sungsu@cnu.ac.kr

Recent Projects

Knowledge Graph Mining
Privacy-Preserving ML (with Nota Inc.)
Graph Summarization (with K. Shin, KAIST)

Selected Publications

[TKDE '19] LinkBlackHole*: Robust Overlapping Community Detection Using Link Embedding
[PRE '19] Efficient Spread Size Approximation of Opinion Spreading in General Social Network
[TIST '17] Differential Flattening: A Novel Framework for Community Detection in Multi-Layer Networks
[JSTAT '16] Motif-Based Embedding for Graph Clustering (Qualcomm Innovation Award)
[ICDE '16] BlackHole: Robust Community Detection Inspired by Graph Drawing
[ICDE '14] LinkSCAN*: Overlapping Community Detection Using the Link-Space Transformation
[TON '14] Stability of the Max-Weight Protocol in Adversarial Wireless Networks (Humantech Paper Award)

Contents

Graph Mining

- **Networks (graphs)**
  - **Node (vertex)**: individual, actor, social entity
  - **Link (edge)**: interaction between individuals

- **Popular problems**
  - Community detection,
    link prediction, recommendation,
    influential user identification,
    social contagion prediction, etc.

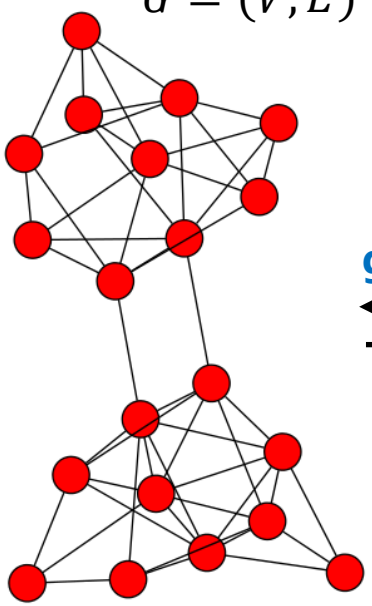
- **Importance of graph mining**
  - To **understand behaviors and interactions** through the structure and dynamics on networks

Facebook	~2B users
WeChat	~1B users
LinkedIn	~0.6B users
	~20B relations

Graph Embedding

- **Representation learning on graphs**
  - Finding embedding of nodes to **low-dimensional space**
  - **Solving the graph mining problems in a vector space**

$G = (V, E)$

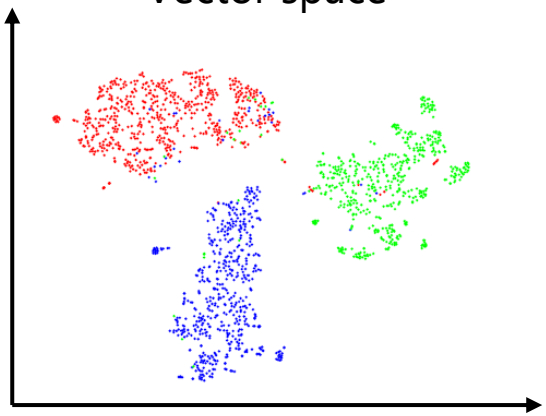


generate ⟵

⟶ **embed**

$G = (V)$

Vector space



Easy to parallel

Can apply classical ML methods

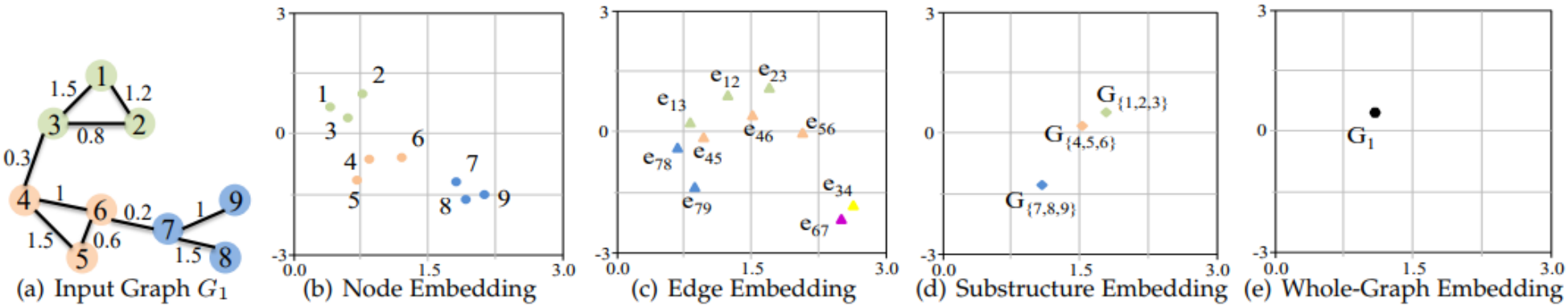
Network inference

- Community detection
- Link prediction
- Node importance
- Node classification
- Network evolution
- …

Reference: P. Cui, Network Embedding, KDD 2019 Tutorial [link]

Graph Embedding

- **Problem Formulation**
  - Input: nodes, links, substructures, graphs
  - Output: positions **preserving graph structures** (or properties)
  - Goal: **solving graph mining problems in an efficient way**

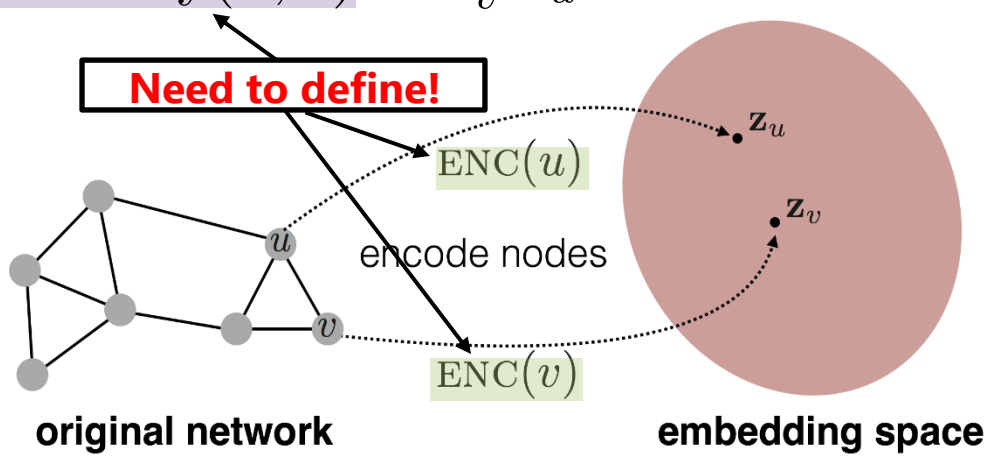


Reference: H. Cai et al., A Comprehensive Survey of Graph Embedding, TKDE 2018 [link]

Goal of Graph Embedding

- **Good graph embedding: "local" structure**
  - Input: nodes (usually)
  - Output: positions **preserving local graph structures**
  - Goal: preserving **similarity between neighboring nodes**

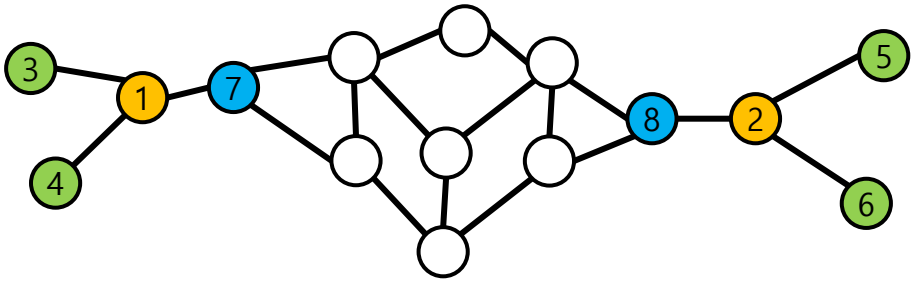
Goal: $\text{similarity}(u, v) \approx \mathbf{z}_v^\top \mathbf{z}_u$



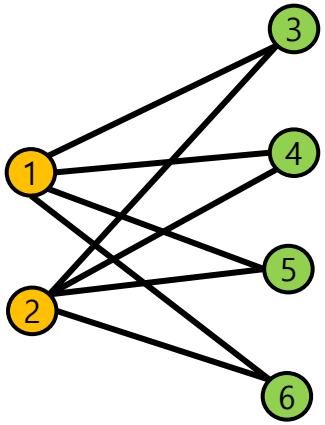
Reference: W. L. Hamilton et al., Representation Learning on Networks, WWW 2018 Tutorial [link]

Goal of Graph Embedding

- **Good graph embedding: "global" structure**
  - Input: nodes (usually)
  - Output: positions **preserving global graph structures**
  - Goal: preserving **similarity considering connectivity patterns**



Structural equivalence



Homophily

Reference: P. Cui, Network Embedding, KDD 2019 Tutorial [link]

Approaches to Graph Embedding

- **General framework**
  1. Define encoder & similarity
  2. Learn node positions in a vector space

- **Main approaches**
  1. Adjacency-based similarity [WWW 2013, CIKM 2015, KDD 2016, KDD 2018, etc.]
  2. Random walk approaches [KDD 2014, KDD 2016, KDD 2017, WWW 2018, etc.]
  3. Graph neural networks [ICLR 2017, NIPS 2017, Deep Learning on Graphs, etc.]
  4. Neighbor embedding [ICML 2014, ICDE 2016, JSTAT 2016, TKDE 2019, etc.]

➡ **Let's review them!**

Contents

- Motivations
- **Techniques**
- Our Results

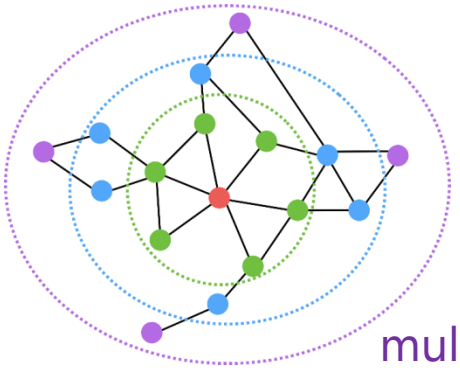
Adjacency-Based Similarity

- **"Lower"-order adjacency similarity**
  - Encoder: minimizing the difference between edge weights & vector similarities
  - Vector similarity: dot products between node positions (cosine similarity)
  - Edge weights: **one-hop** [WWW 2013] & **multi-hop** [CIKM 2015]

$$\mathcal{L} = \sum_{(u,v)\in V\times V} \left\| \boxed{\mathbf{z}_u^T \mathbf{z}_v} - \boxed{A_{u,v}^k} \right\|^2$$

from a vector space

adjacency matrix



multi-hop

where u, v: nodes in a graph, A: (weighted) adjacency matrix,

$\mathbf{z}_u, \mathbf{z}_u$: node positions, $\mathcal{L}$: loss function defined on the node positions

Adjacency-Based Similarity

- **"Higher"-order adjacency similarity**
  - Encoder: learning vector positions using eigenvalue decomposition
  - Edge weights: **higher-order** [KDD 2016] & **arbitrary order** [KDD 2018]
    - HOPE [KDD 2016]: similarity measures from social network analysis ($S = M_g^{-1} \cdot M_\ell$)
    - AROPE [KDD 2018]: similarity is a polynomial of adjacency ($S = \mathcal{F}(A) = w_1 A^1 + \cdots + w_q A^q$)

$$\min_{U^*, V^*} \left\| S - U^* V^{*T} \right\|_F^2$$

$U^*, V^* \in \mathbb{R}^{N \times d}$: embedding vectors

$w_1, \ldots, w_q$: arbitrary weights

d: dimensionality of the space

Proximity Measurement	$\mathbf{M}_g$	$\mathbf{M}_l$
Katz	$\mathbf{I} - \beta \cdot \mathbf{A}$	$\beta \cdot \mathbf{A}$
Personalized Pagerank	$\mathbf{I} - \alpha \mathbf{P}$	$(1-\alpha) \cdot \mathbf{I}$
Common neighbors	$\mathbf{I}$	$\mathbf{A}^2$
Adamic-Adar	$\mathbf{I}$	$\mathbf{A} \cdot \mathbf{D} \cdot \mathbf{A}$

↑ from social network analysis

<u>THM</u> $[\lambda, \mathbf{x}]$ is an eigenpair of $A \Rightarrow [\mathcal{F}(\lambda), \mathbf{x}]$ is an eigenpair of S (the order is different)

Random Walk Approaches

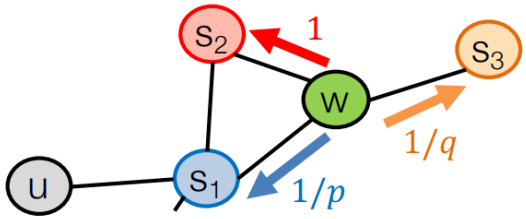
- **Random walk-based "node" similarity**
  - Encoder: minimizing the difference between RW similarities & vector similarities
  - Vector similarity: dot products between node positions (cosine similarity)
  - RW-based similarities: **language modeling** [KDD 2014] & **biased RW** [KDD 2016]
    - DeepWalk [KDD 2014]: similarity is a degree of co-occurrence in simple random walks
    - node2vec [KDD 2016]: similarity is a weighted sum of BFS-like & DFS-like random walks

nd the cold , close moon " . And neither c
the night with the moon shining so bright
in the light of the moon . It all boils dc
ly under a crescent moon , thrilled by ice
the seasons of the moon ? Home , alone ,
dazzling snow , the moon has risen full ar
d the temple of the moon , driving out of

Language sentences

$v_{92} \rightarrow v_2 \rightarrow v_3 \rightarrow v_1 \rightarrow v_{12} \rightarrow v_{73} \rightarrow$
$v_{37} \rightarrow v_{34} \rightarrow v_9 \rightarrow v_1 \rightarrow v_{10} \rightarrow v_{94} \rightarrow$
$v_{73} \rightarrow v_{64} \rightarrow v_5 \rightarrow v_1 \rightarrow v_{12} \rightarrow v_1 \rightarrow$
$v_{75} \rightarrow v_{14} \rightarrow v_6 \rightarrow v_1 \rightarrow v_{13} \rightarrow v_{61} \rightarrow$

Random walk in a graph



- **BFS-like** walk: Low value of p
- **DFS-like** walk: Low value of q

BFS-like walk: local microscopic view
DFS-like walk: global macroscopic view

Random Walk Approaches

- **Random walk-based "structure/role" similarity**
  - Encoder: minimizing the difference between RW similarities & vector similarities
  - Vector similarity: dot products between node positions (cosine similarity)
  - RW-based similarities: **structural identity** [KDD 2017] & **roles** [WWW 2018]
    - struc2vec [KDD 2017]: similarity measures the structural distance between nodes
    - VERSE [WWW 2018]: similarity measures using personalized PageRank, SimRank, etc.

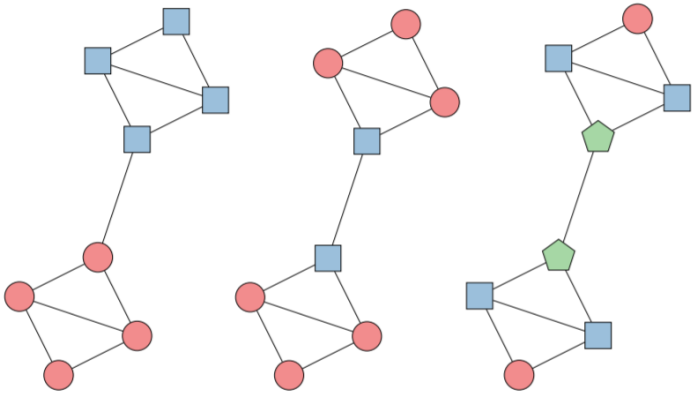
$R_k(u)$: set of nodes at distance k from u

$g(D_1, D_2)$: distance between degree sequences

Structural distance:

$$\boxed{f_k(u,v) = f_{k-1}(u,v) + g(R_k(u), R_k(v))}$$

⇒ construct a multilayer graph (for each k) to encode structural similarity



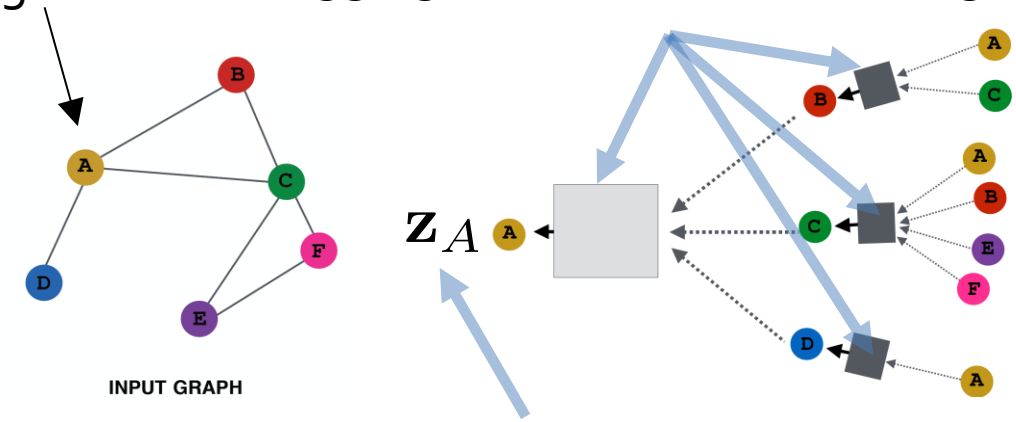
(a) Community structure (b) Roles (c) Structural equivalence

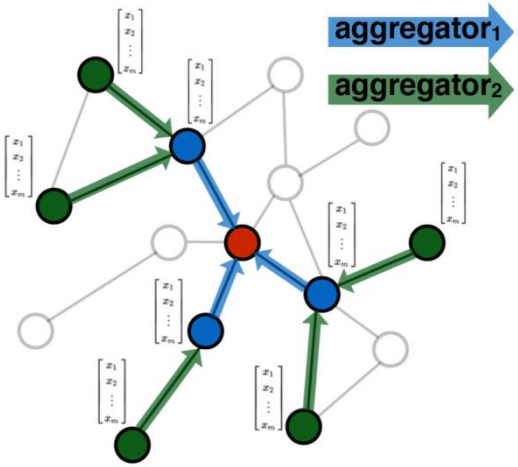
Graph Neural Networks

- **Graph convolutional networks**
  - Embedding using **convolutional networks** [ICLR 2017]
  - Every node defines a unique computation graph
  - Same aggregation parameters are shared for all nodes

1) Sample neighborhood &
aggregate information from neighbors

Target node



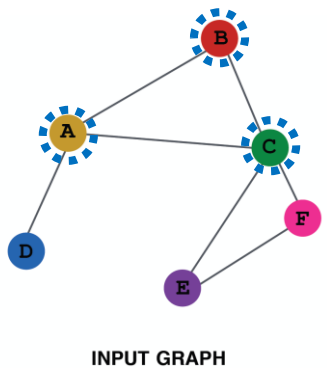


2) Define a loss function on the embeddings

Reference: W. L. Hamilton et al., Representation Learning on Networks, WWW 2018 Tutorial [link]

Graph Neural Networks

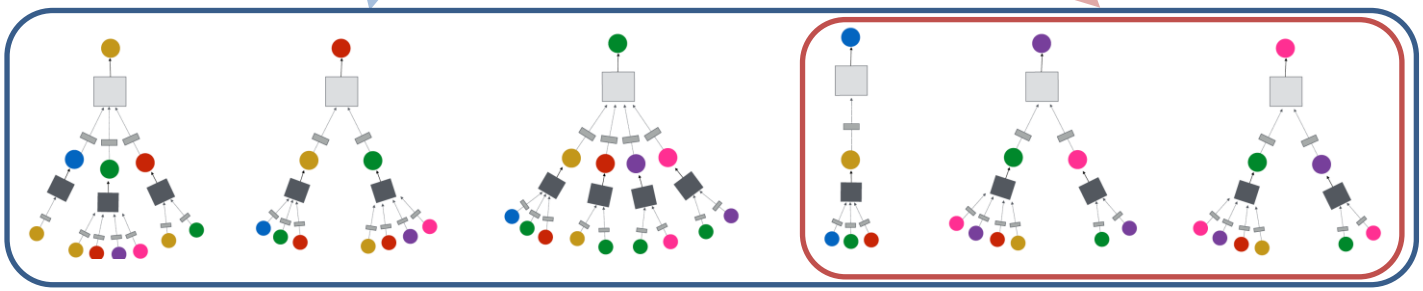
- **Graph convolutional networks**
  - Embedding using convolutional networks
  - **Inductive capability**: can be generalized to completely unseen data [NIPS 2017]



3) Train on a set of nodes, i.e., a batch of compute graphs

4) Generate embeddings for nodes as needed

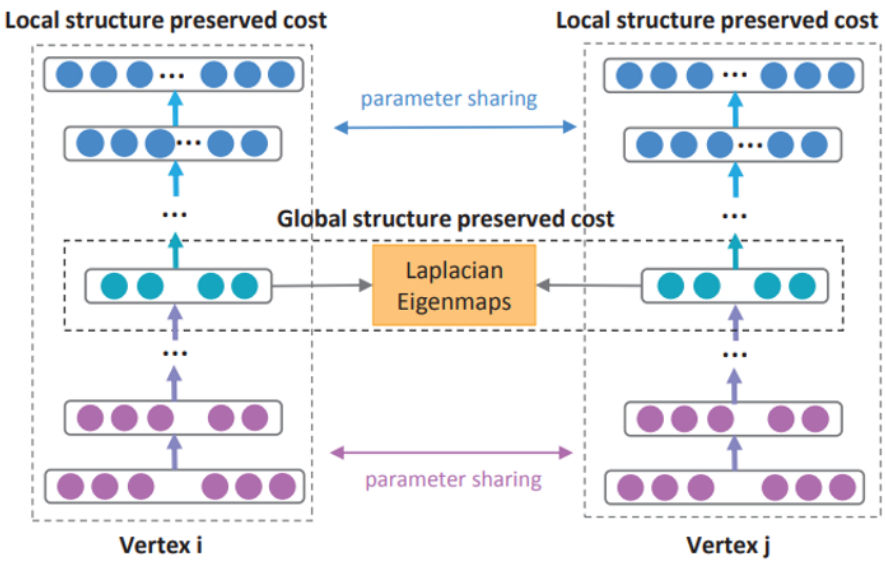
Even for nodes we never trained on

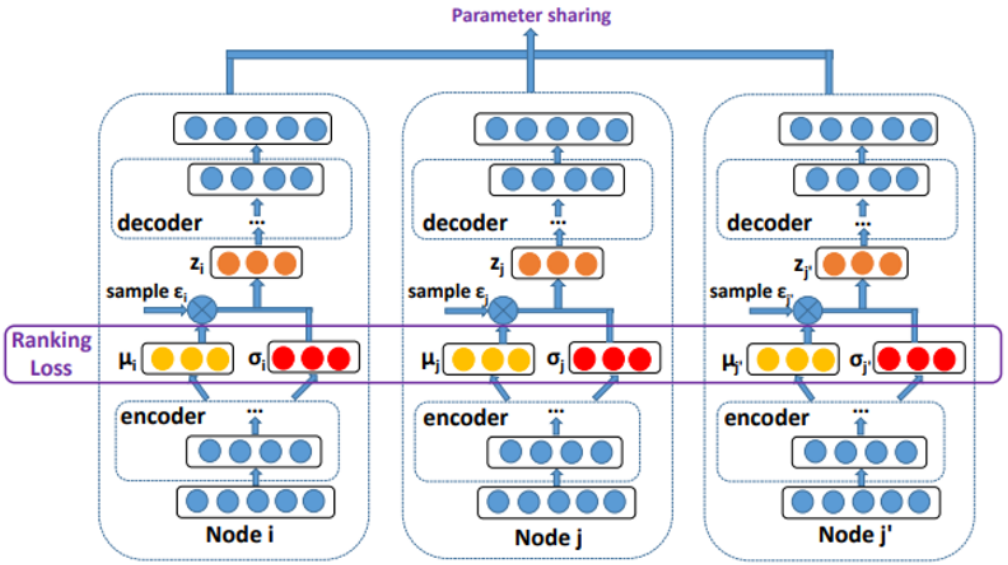


Reference: W. L. Hamilton et al., Representation Learning on Networks, WWW 2018 Tutorial [link]

Graph Neural Networks

- **Deep learning on graphs**
  - Interest in graph neural networks has exploded over the past years [ArXiv 2019]
    - SDNE [KDD 2016]: **AE** (Autoencoder) is used to learn node representations
    - DVNE [KDD 2018]: **VAE** (Variational AE) is used to learn node representations





Reference: Z. Zhang et al., Deep Learning on Graphs: A Survey [link]

Neighbor Embedding

- **Divergence minimization on graphs**
  - Encoder: **minimizing the sum of divergences** for each node ($\sum_u D(\boldsymbol{x}_u|\boldsymbol{y}_u)$)
  - It covers Laplacian eigenmap, elastic embedding, SNE and its variants on graphs, force-directed embedding, etc. [ICML 2014]

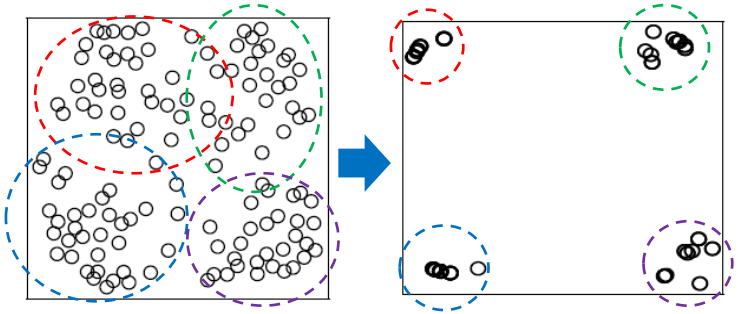
- **Force-directed graph embedding**
  - Find node positions that minimize

$$\mathcal{E}(p|G) = \sum_{\{u,v\}\in V\times V} \left(\underbrace{\frac{w_{u,v}}{a+1}\|p(u)-p(v)\|^{a+1}}_{\text{Attraction}} - \underbrace{\frac{w_u w_v}{r+1}\|p(u)-p(v)\|^{r+1}}_{\text{Repulsion}} \right),$$

    where w_u, w_v: node weights, $w_{u,v}$: edge weights, a, r: attraction/repulsion weights
    $p(u), p(v)$: node positions, $\mathcal{E}$: energy defined on the node positions
  - Attraction: **adjacent nodes are located close to each other**

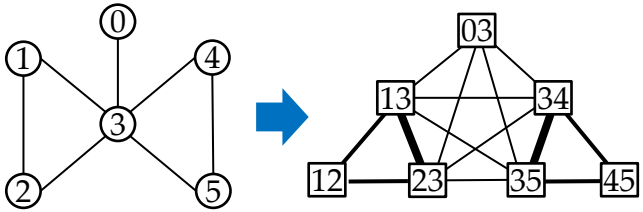
Neighbor Embedding

- **BlackHole embedding**
  - BlackHole embedding [ICDE 2016]: set $w_u = \frac{w_u}{\sqrt{\sum_k w_k}}$ and $a - r \gtrsim 0$
  - Due to the **strong attraction**, it **attracts the nearby nodes into a black hole** & conventional algorithm finds communities of node positions (and nodes)
  - LinkBlackHole* [TKDE 2019]: it finds overlapping communities by finding communities in the link-space graph of the original graph



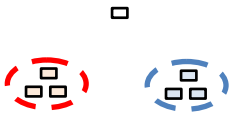
Force-directed embedding

BlackHole embedding



Original Graph

Link-Space Graph



Link Embedding

Contents

- Motivations
- Techniques
- **Our Results**

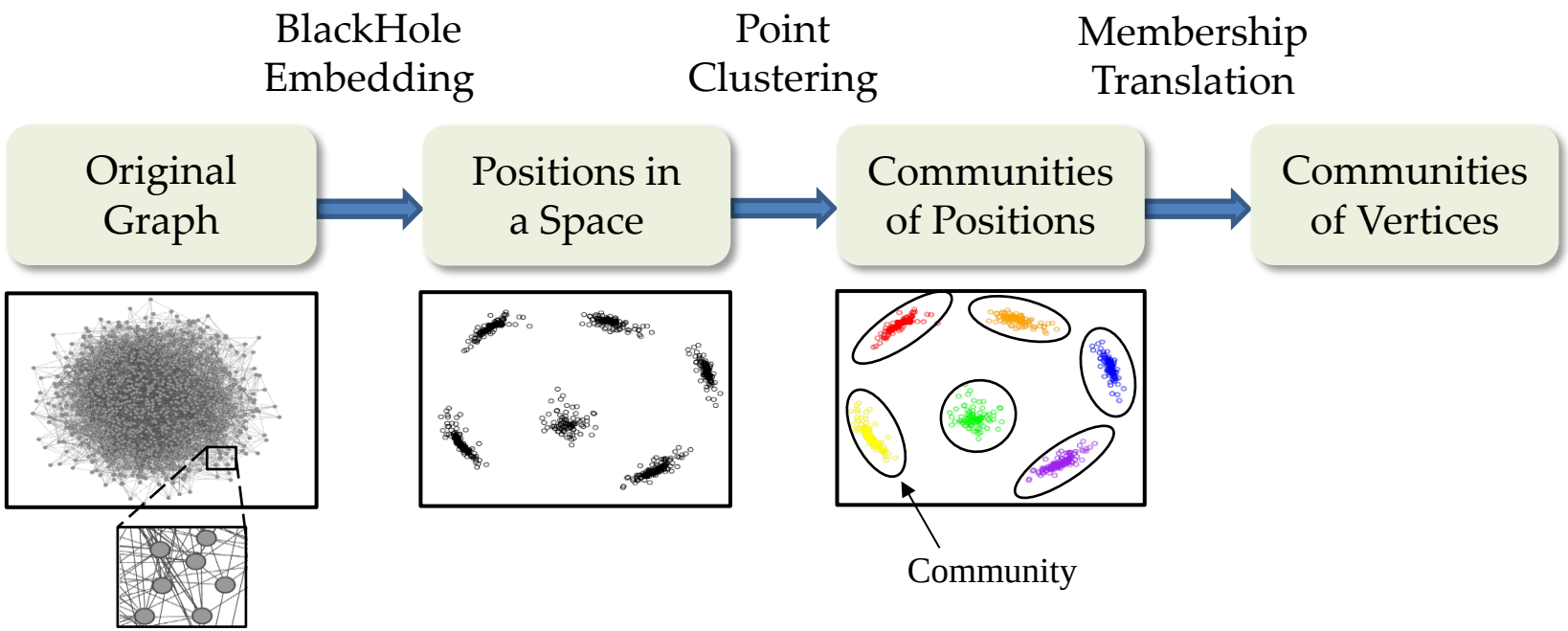
Part I
BlackHole embedding

BlackHole: Robust Community Detection Inspired by Graph Drawing
IEEE ICDE 2016

Joint work with J. Kim (NTU, Singapore) and J.-G. Lee (KAIST)

Proposed Algorithm: BlackHole

- Proposing the ***BlackHole embedding*** that transforms a given graph into the points in a low-dimensional space
- Developing an algorithm that performs clustering on the embedded space, which enables us to discover **highly mixed communities**

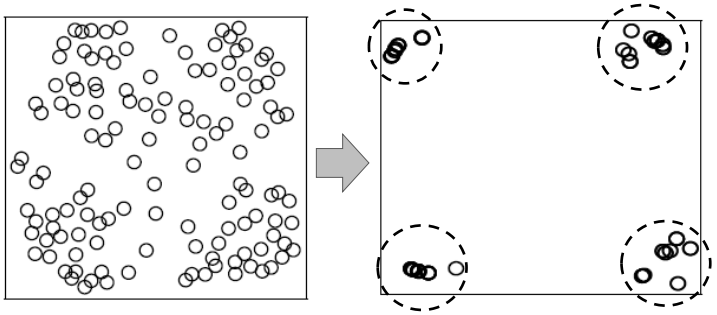


Phase I: BlackHole Embedding

- Solving an energy minimization problem through multiple iterations to determine vertex positions

$$\min_{p:V\to S} \sum_{\{i,j\}\in V^{(2)}} \left(\underbrace{\frac{w_{ij}}{a+1}\|p(i)-p(j)\|^{a+1}}_{\textbf{Attraction}} - \underbrace{\frac{w_i w_j}{r+1}\|p(i)-p(j)\|^{r+1}}_{\textbf{Repulsion}} \right)$$

  - LinLog: $a = 0, r = -1$
    - Conventional design
  - ***BlackHole***: $a = -0.95, r = -1$
    - Relatively strong repulsion
    - Exponential growth in attraction



LinLog **BlackHole**

Design of Our Repulsive Forces

- Increasing the order of repulsive forces ($r = -1$)
  - cf., Fruchterman-Reingold ($r = -1$)
    Davidson-Harel ($r = -3$)
    LinLog ($r = -1$)

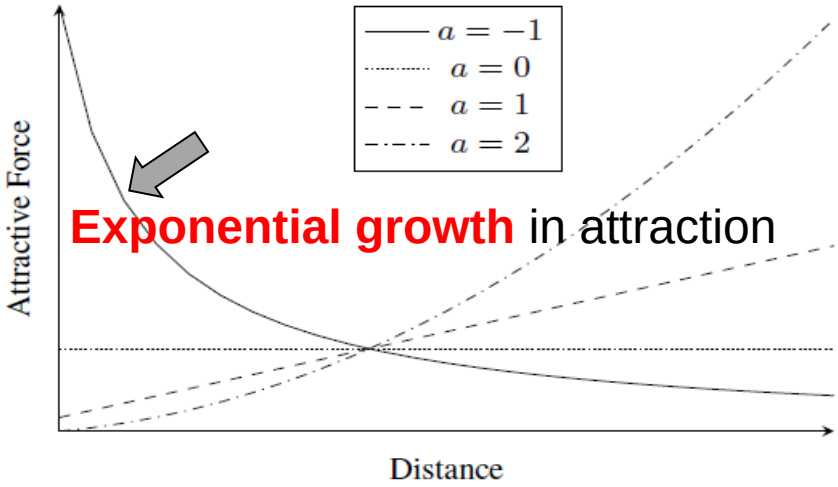
$\Rightarrow$ Making positions more separable in early stages

How to **break** balls?
Increase the **force**!

Design of Our Attractive Forces

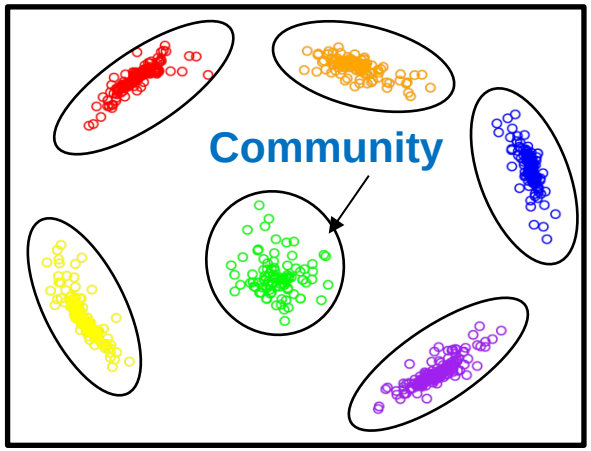
- Increasing the attractive force **much stronger** as the connected vertices **get closer to each other** $(a \approx -1)$
  - cf., Fruchterman-Reingold $(a = 2)$
    Davidson-Harel $(a = 1)$
    LinLog $(a = 0)$

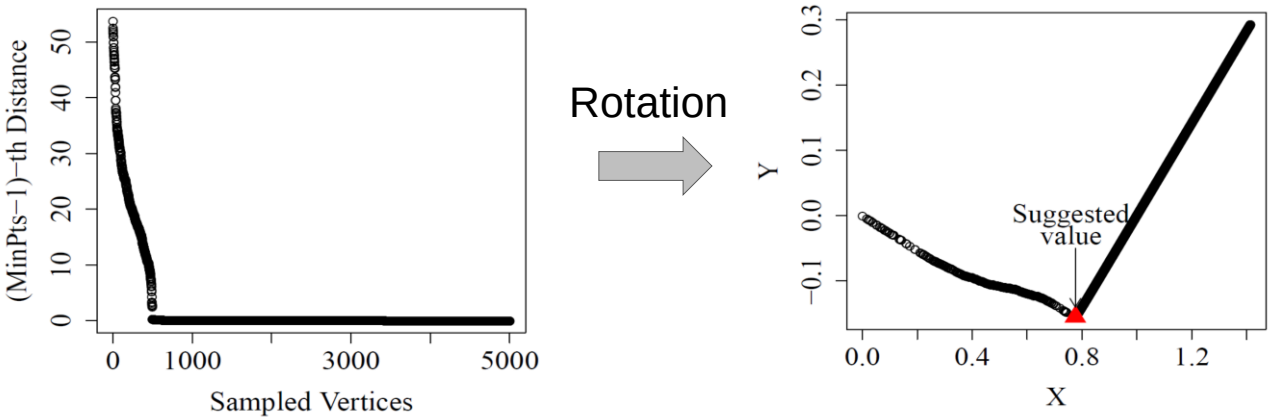
$\Rightarrow$ Attracting the nearby vertices into a ***black hole***



Phase II: Clustering

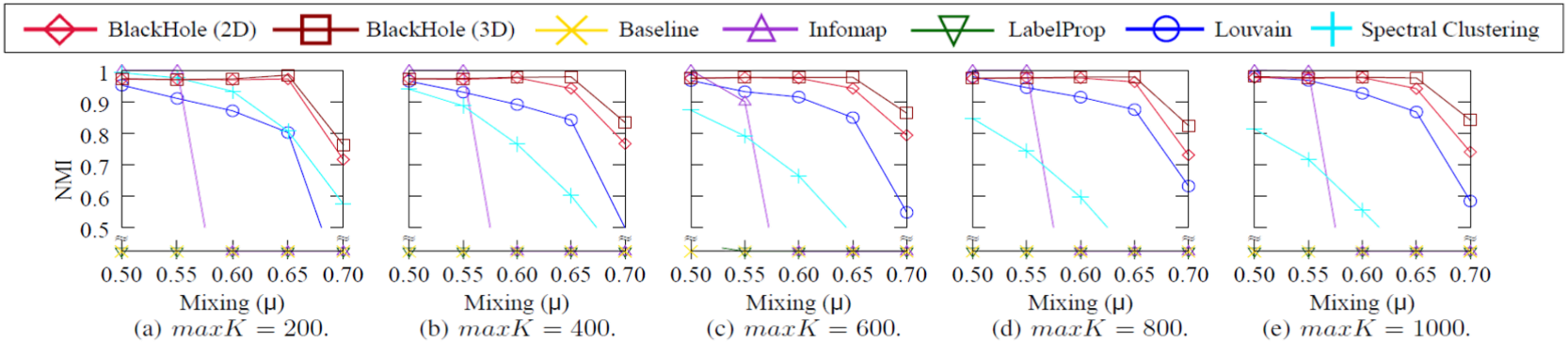
- Applying conventional clustering algorithms to the vertex positions obtained in Phase I
- Adopting **DBSCAN**
  - The two parameters ε and *MinPts* are determined by the heuristic



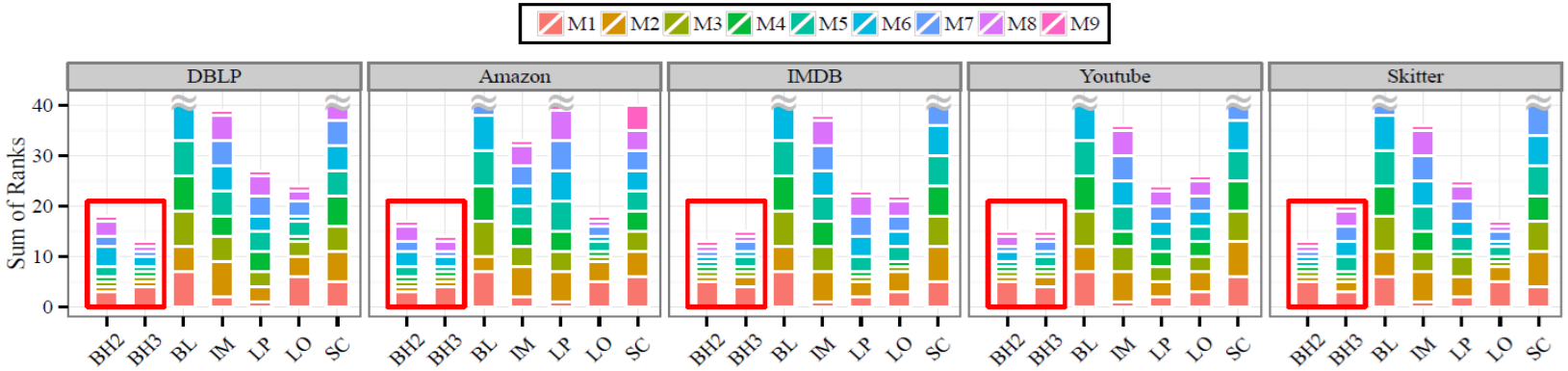


Performance Evaluation

- Providing higher quality for both synthetic and real-world networks



Effect of mixing parameter (fraction of inter-community cuts)



Cumulative ranks of M1~M9 for each algorithm

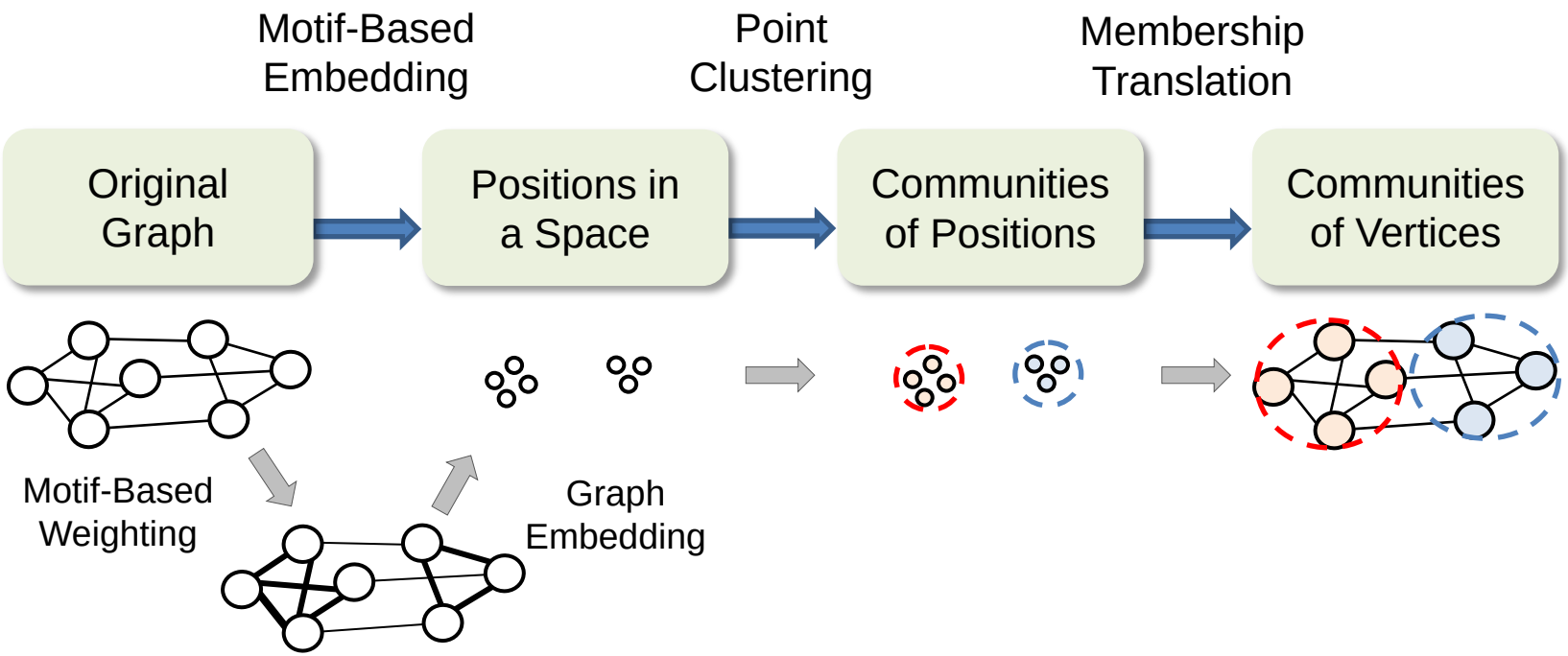
Part II
Motif-based embedding

Motif-Based Embedding for Graph Clustering
Journal of Statistical Mechanics: Theory and Experiments, 2016

Joint work with J.-G. Lee (KAIST)

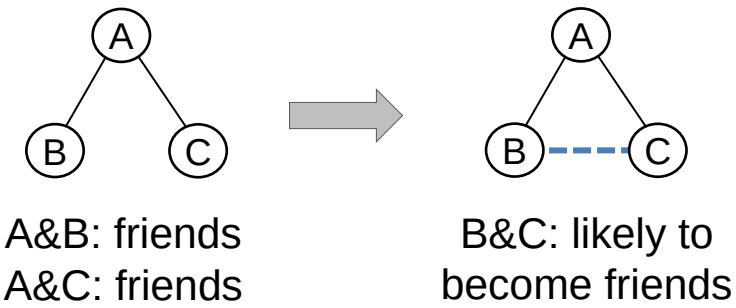
Proposed Algorithm: Motif-Based Embedding

- Proposing the ***motif-based embedding*** that transforms a given graph into points in a low-dimensional space
- Developing an algorithm that performs clustering on the embedded space, which enables us to discover **higher-order graph substructure**

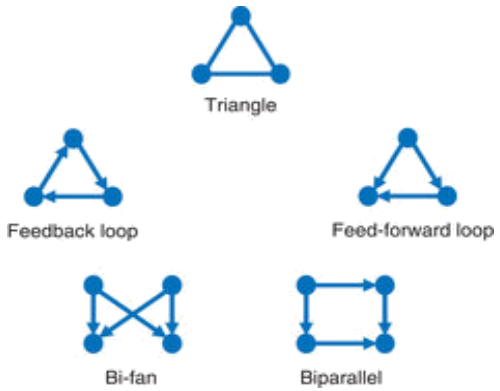


Network Motifs

- Higher-order graph substructures
  - The relationships involve **multiple vertices** within clusters, *e.g.*, triangles in social networks
  - By incorporating the motifs, the ***motif-based weighting*** method reflects motif substructures in a given graph



Triadic closure



Motifs in biological networks

Motif-Based Embedding

- Solving an energy minimization problem through multiple iterations to determine vertex positions

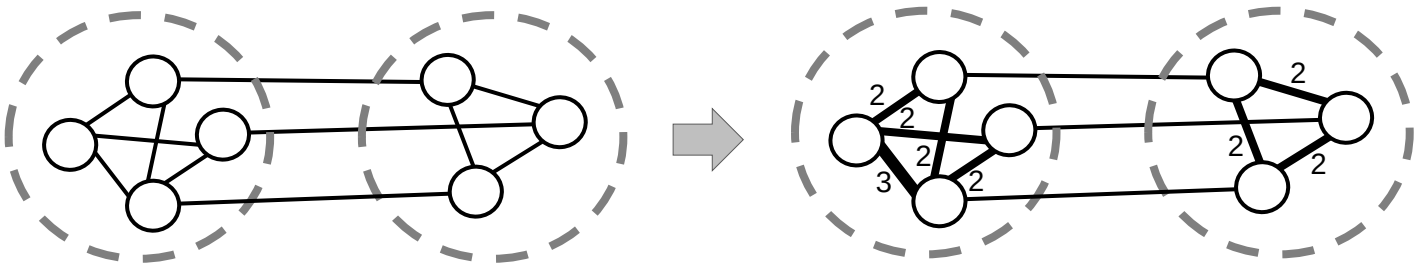
$$\min_{p:V\to S} \sum_{\{i,j\}\in V^{(2)}} \underbrace{\frac{\boldsymbol{f(i,j)}}{a+1}\|p(i)-p(j)\|^{a+1}}_{\textbf{Attraction}} - \underbrace{\frac{w_i w_j}{r+1}\|p(i)-p(j)\|^{r+1}}_{\textbf{Repulsion}}$$

  - Force-directed embedding: $f(i,j) = w_{ij}$
    - Conventional design
  - ***Motif-Based Embedding***: $f(i,j) = w_{ij} + m_{ij}$
    - Weights calculated from edges and motifs
    - Strong attraction within motifs

Motif-Based Weighting

- Weighting every pair of vertices by $f: V \times V \to \mathbb{R}$, with **strong attraction** within the motifs of interest
  - **Example:**
    $f: \{i, j\} \mapsto w_{ij} + m_{ij}$
    w_{ij}: the existence of an edge $\{i, j\}$
    m_{ij}: # of motifs containing i & j together

Many motifs (triangles) are within a cluster but not across clusters



Mixing: 0.43 Mixing: 0.26

  - The lower the mixing (fraction of sum of inter-community weights) is, the more detectable the community structure is

Performance Evaluation: Synthetic Networks

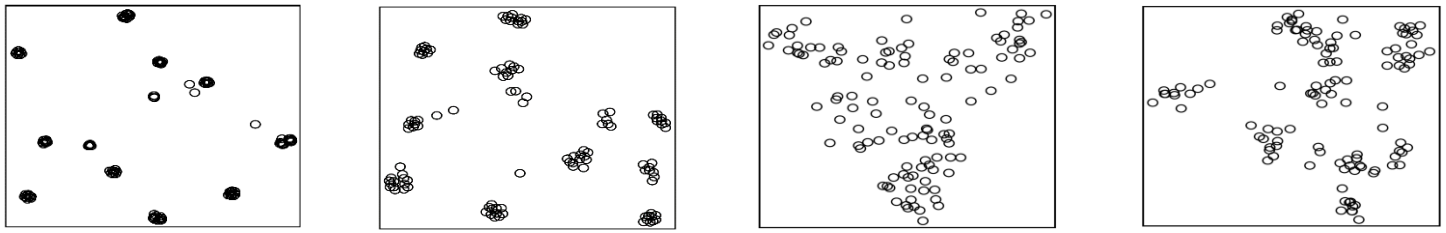
- **Embedding methods**
  - (i) Motif-based embedding (2-dim. space) ⇦ ***Proposed method***
  - (ii) Force-directed embedding (2-dim. space)
  - (iii) Spectral embedding (k-dim. space)
  - (iv) Spectral embedding with motif-based weights (k-dim. space)
- **Used motifs**
  - Triangles for all synthetic networks
- **NMI results** (clustering: embedding + k-means)

	Proposed	Method (ii)	Method (iii)	Method (iv)
Mixing = 0.4	**0.99**	0.89	0.70	0.73
Mixing = 0.5	**0.84**	0.79	0.37	0.41
Mixing = 0.6	**0.58**	0.47	0.13	0.26

Performance Evaluation: Real Networks

- **Used motifs**
  - Social graphs (Football network): triangles
  - Bipartite graphs (Malaria network): wedges
- **Accuracy**

	Proposed	Method (ii)	Method (iii)	Method (iv)
Social graphs	**0.93**	0.91	0.75	0.89
Bipartite graphs	**1**	0.52	0.01	**1**



(i) Proposed embedding | (ii) Edge-based embedding | (iii) Spectral embedding | (iv) Spectral embedding with motif-based weights

Embedding results of each algorithm for the football network

Part III
Link embedding

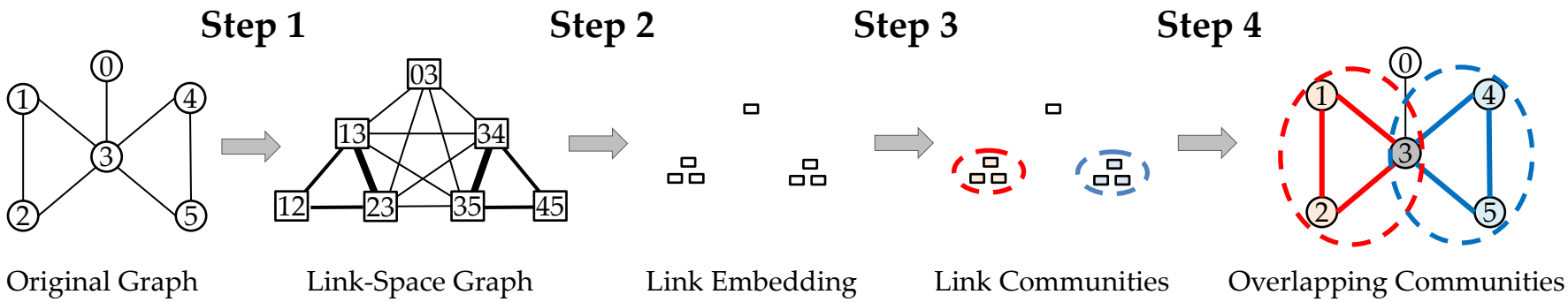
LinkBlackHole: Robust Overlapping Community Detection Using Link Embedding*

IEEE TKDE 2019 (Prior work: IEEE ICDE 2014 & 2016)

Joint work with J. Kim (ETRI), B.S. Lee (U. Vermont), and J.-G. Lee (KAIST)
(Prior work: collaborated with S. Ryu (ADD), S. Kwon (Samsung), and K. Jung (SNU))

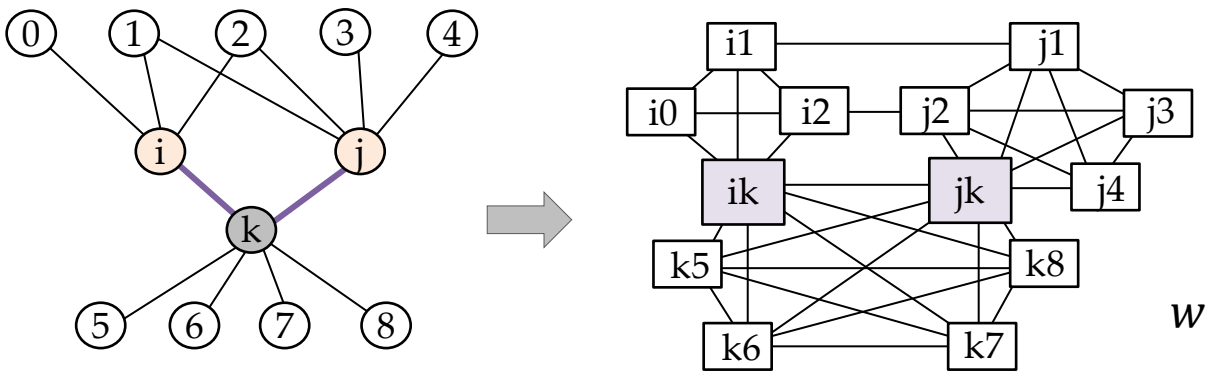
Proposed Algorithm: LinkBlackHole*

- Step 1: **Link-space transformation** on the original graph
- Step 2: **BlackHole embedding** on the link-space graph
- Step 3: Clustering for the vertex positions of the link-space graph
- Step 4: Membership translation procedure
  - 4-1: Disjoint communities of the vertex positions of the link-space graph
  - 4-2: Disjoint communities of the links of the original graph
  - 4-3: Overlapping communities of the vertices of the original graph



Phase I: Link-Space Transformation

- **Topological structure**
  - Link (original graph) → node (link-space graph)
  - Two incident links (original graph) → link (link-space graph)
- **Weights**
  - Similarity between links (original graph) → weight of a link (link-space graph)



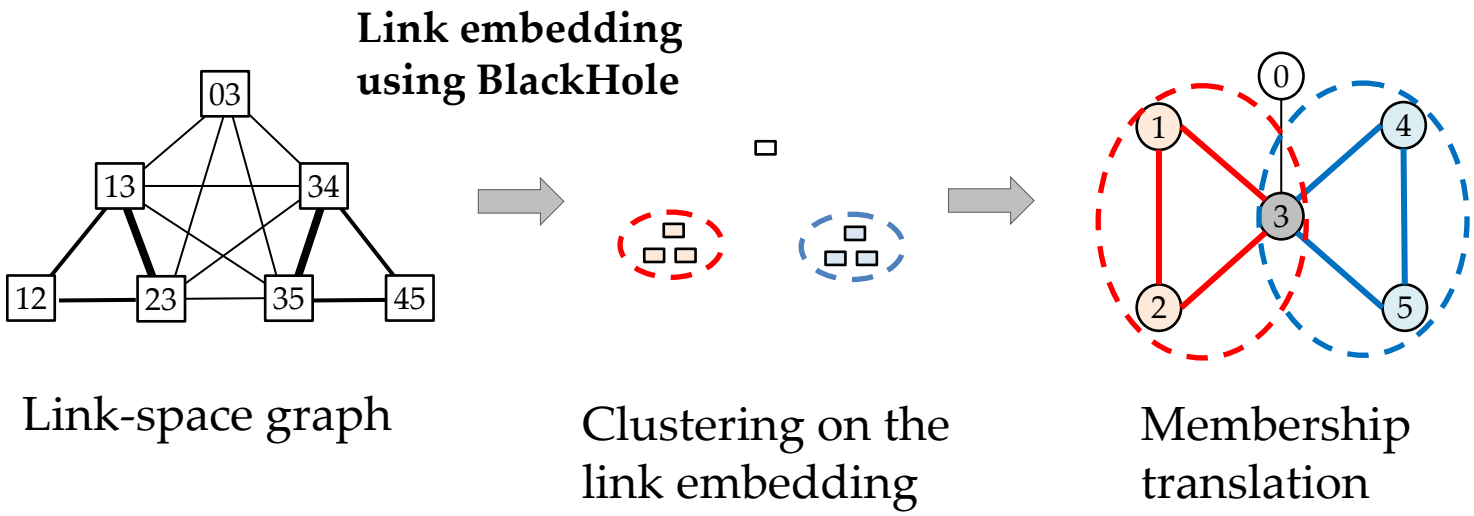
Original graph

Link-space graph

$$w(v_{ik}, v_{jk}) = \sigma(e_{ik}, e_{jk})$$

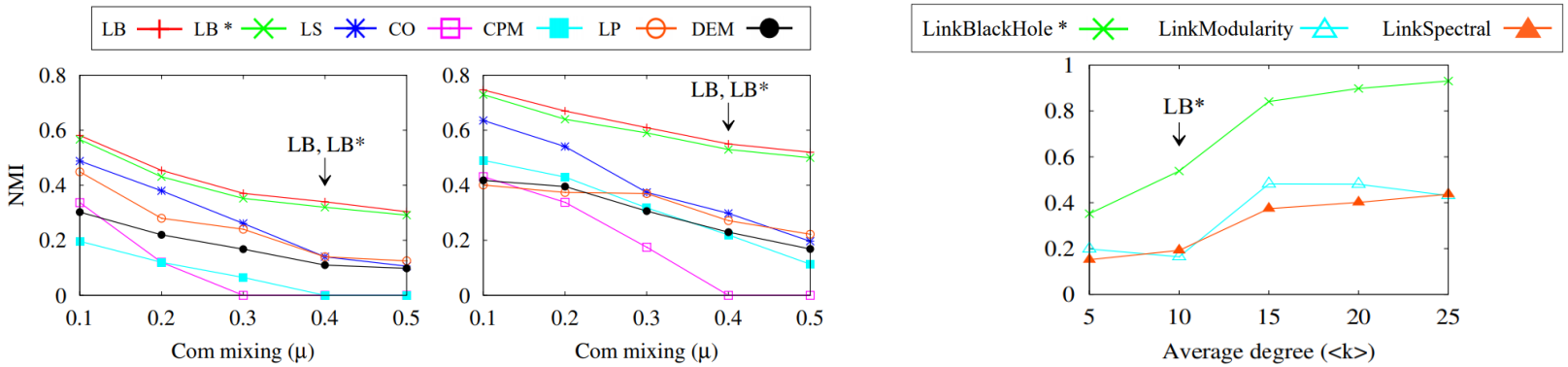
Phase II: BlackHole Embedding + Clustering

- **Link Embedding**: Applying a BlackHole embedding to the link-space graph
- Using structural clustering (SCAN) that can assign a node into hubs or outliers (neutral membership)

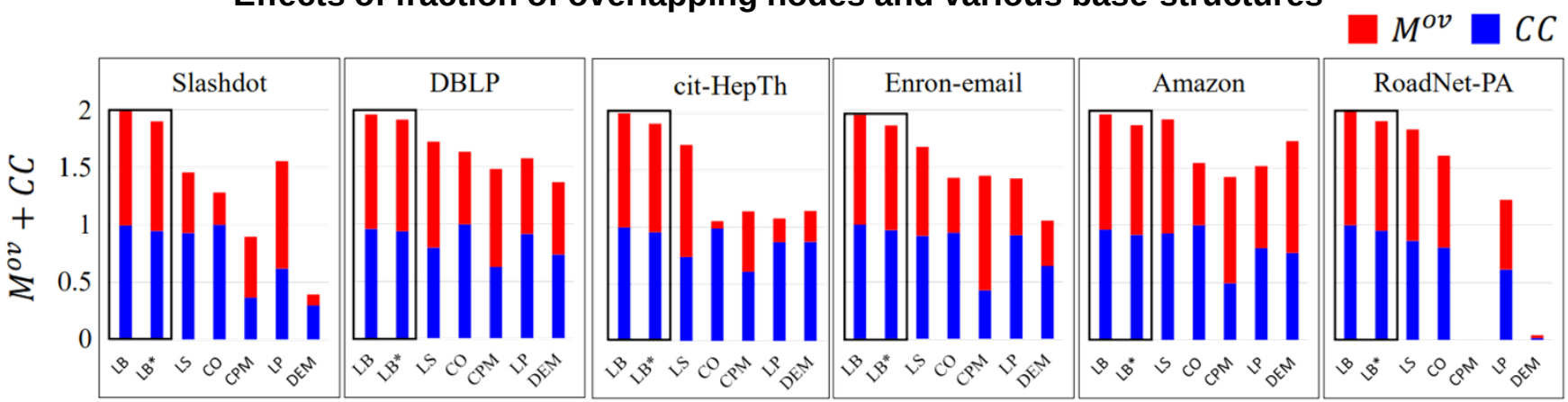


Performance Evaluation

- Providing higher quality for both synthetic and real-world networks



Effects of fraction of overlapping nodes and various base-structures



Normalized measure of (Quality + Coverage) for each algorithm

Thank You Very Much!

Any Questions?