Directed Network Embedding with Virtual Negative Edges

09.08.2022

Hyunsik Yoo

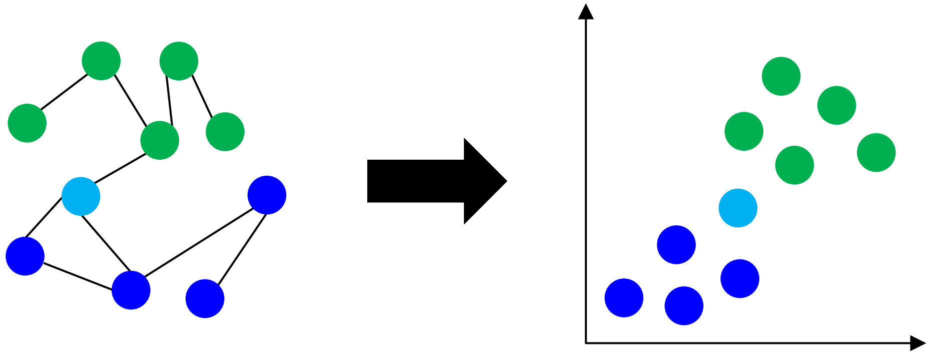
* This is a joint work with Prof. Sang-Wook Kim and Dr. Yeon-Chang Lee at **Hanyang Univ.**, and Prof. Kijung Shin at **KAIST**, published in **ACM WSDM 2022**

Table of Contents

- **Background**
  - Network Embedding (NE)
  - Directed Network Embedding (DNE)
- **Motivation**
- **Proposed Approach**
  - STEP 1: Inferring the Degree of Negativity
  - STEP 2: Selecting Virtual Negative Edges
  - STEP 3: Modeling a Signed Directed Network
  - STEP 4: Learning Source and Target Embeddings
- **Evaluation**
- **Conclusions**

Background

- **Network embedding (NE)**
  - Represents nodes in a given network as low-dimensional vectors that preserves the structural properties of the network
    - *e.g.*, proximity between nodes

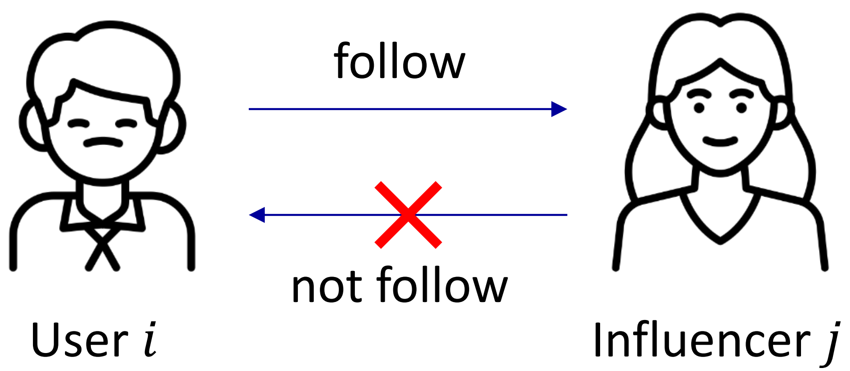


  - Can be used as informative features of nodes in various downstream network mining tasks
    - Link prediction
    - Node clustering/classification
    - Recommendation

Background (cont'd)

A directed network

- A directed edge from node i to j expresses an asymmetric relationship (or proximities) between two nodes
- A toy example on Instagram

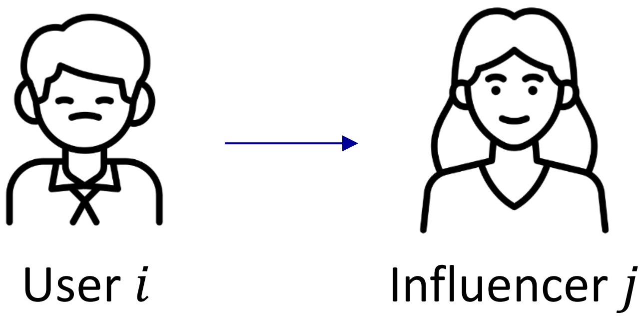


To capture such asymmetric relationships accurately, various directed NE methods have been proposed

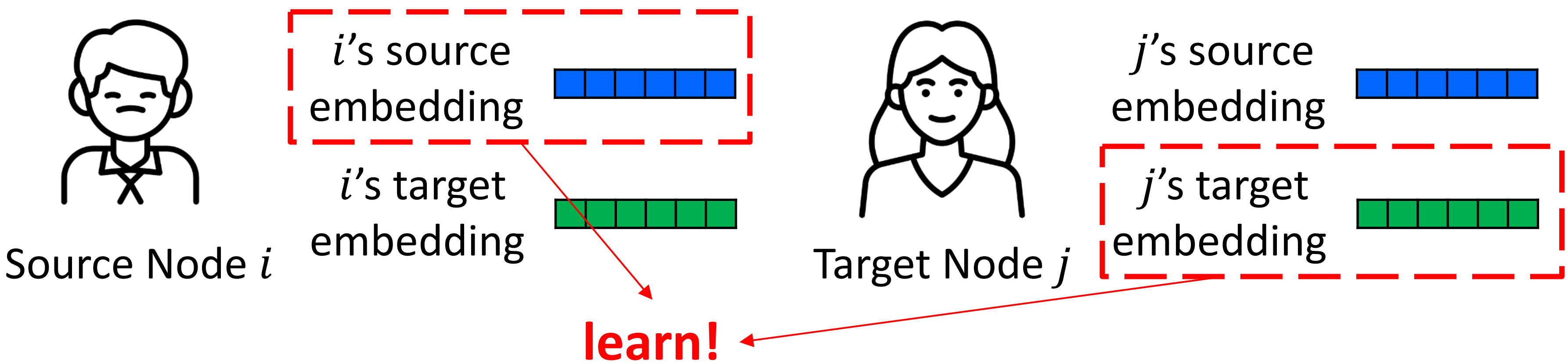
- APP [AAAI'17], ATP [AAAI'19], NERD [ECML-PKDD'19], GravityAE/VAE [CIKM'19], DiGCN [NeurIPS'20]

Directed NE Methods

Given a directed edge from i to j,

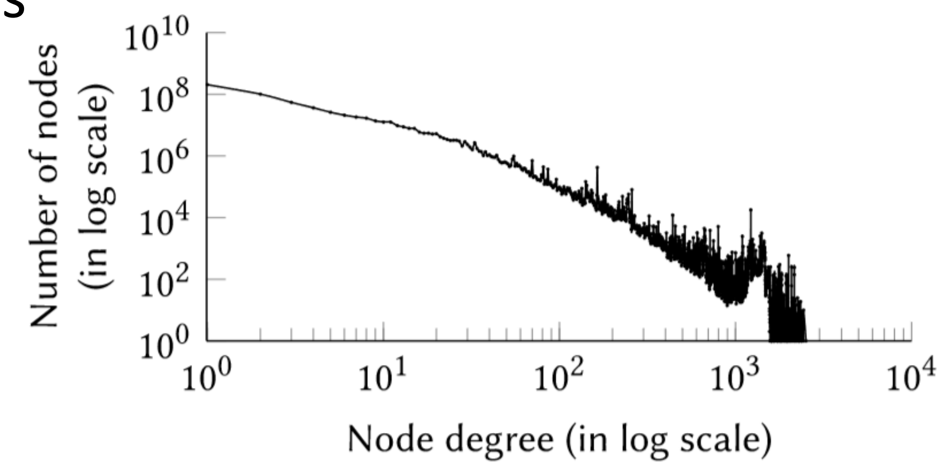


- Distinguish the source node i and the target node j according to their roles in the edge
- Learn a source embedding and a target embedding, which preserve the node's properties as sources and targets



Motivation

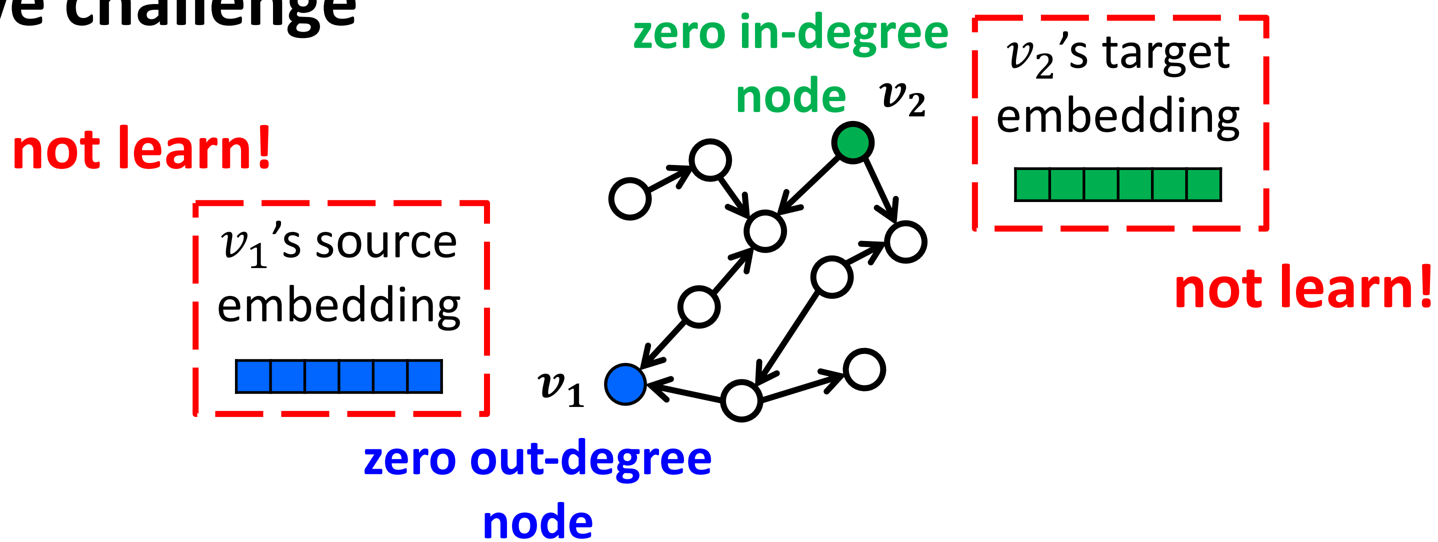
- **Sparsity of real-world networks**
  - Follow power-law degree distribution, which indicates there are a small number of hub nodes and a large number of non-hub nodes



  - **Most nodes have extremely low out- and in-degrees!**

Motivation (cont'd)

- **Challenge of directed NE methods**
  - They hardly learn the source/target embedding of low out-/in-degree nodes
  - Thus, they easily fail to capture the properties of low out- and in-degree nodes as sources and targets, respectively
- **Since a considerable fraction (34.86%/34.29%) of nodes have a zero out- and in-degree, it aggravates the above challenge**



Our Idea: Data Augmentation

- **NE's intrinsic difficulty is its lack of information when embedding low out- and in-degree nodes in a sparse directed network**

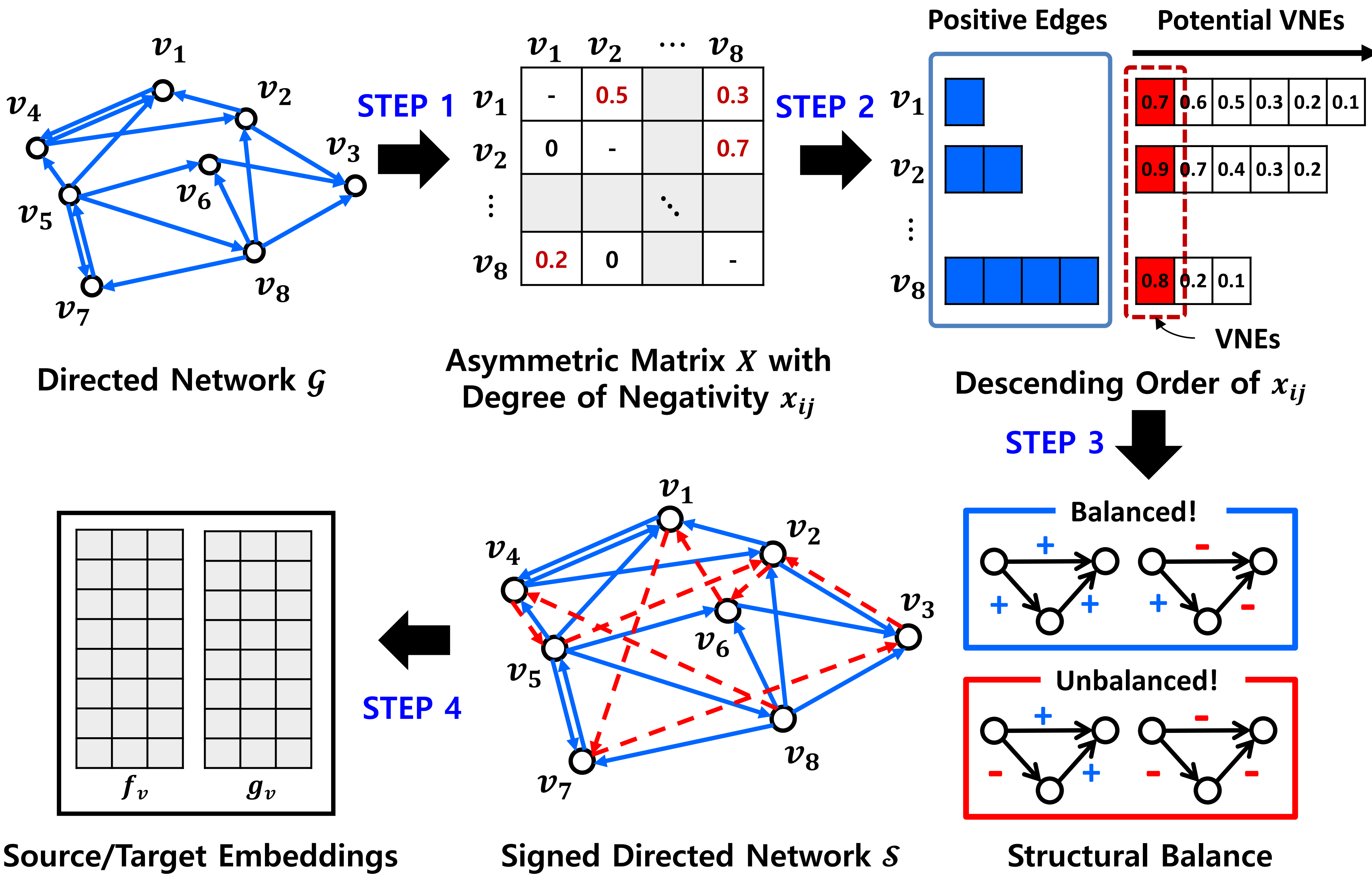
- **New concept: Virtual Negative Edges (VNEs)**
  - Represent latent negative relationships between nodes

- **We propose a novel DIrected NE approach with VIrtual Negative Edges, named DIVINE**
  - Carefully determine the number and location of VNEs to be added to the input network
  - Learn embeddings by exploiting both edge types

Why Virtual Negative Edges?

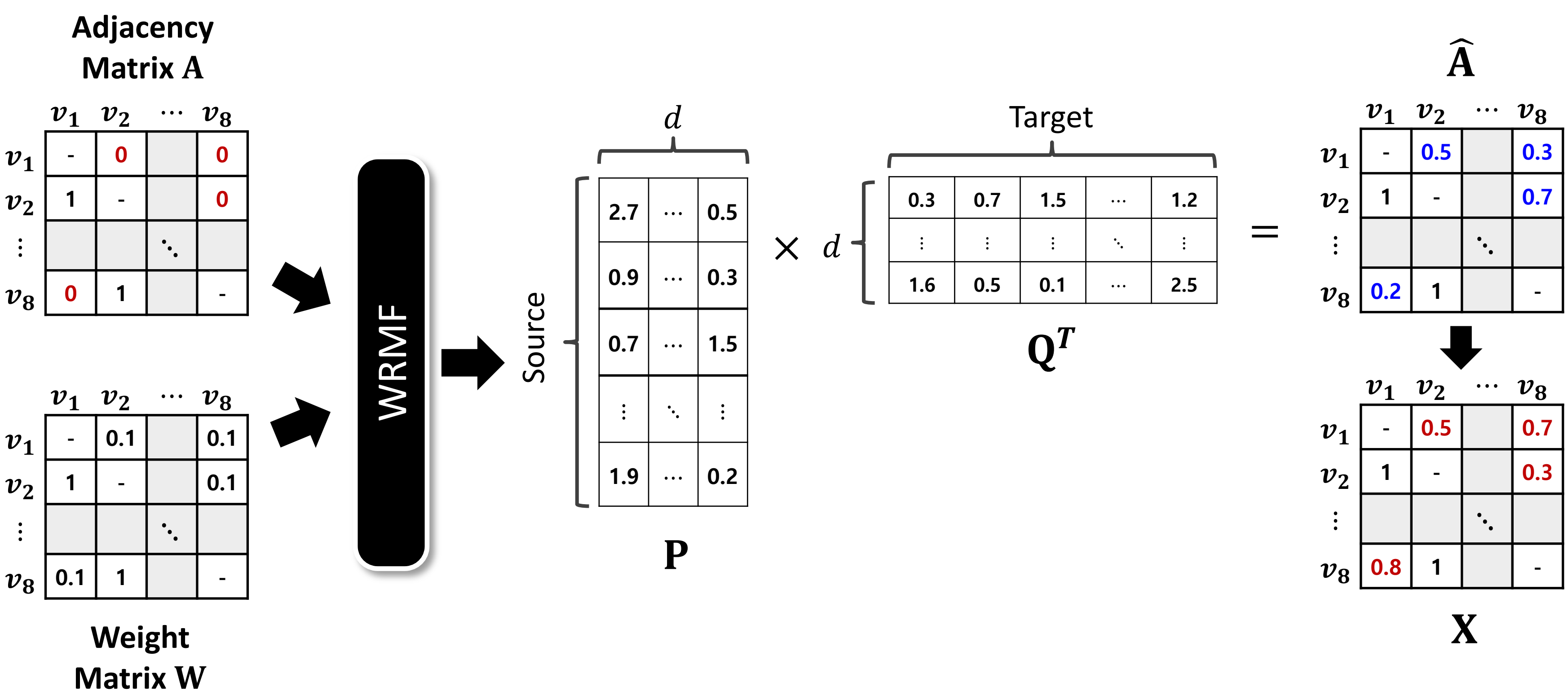
- **Adding virtual edges (VEs) facilitates the utilization of information expressed in the form of VEs**
  - It has been proven useful for various graph mining tasks
    - e.g., node classification [Klicpera et al. NeurIPS'19, Zhao et al. AAAI'21], community detection [Kang et al. CIKM'20]
  - They only focused on positive edges (VPEs)
- **However, we confirmed that VNEs provide information more useful to directed NE methods than VPEs**
  - The information inherent in VNEs is more difficult for directed NE methods to utilize (than that in VPEs) unless it is explicitly provided in the form of VEs

Overview of DIVINE



STEP1: Inferring the Degree of Negativity

- Quantify the degree of positivity of all pairs of nodes based on *weighted regularized matrix factorization*
- Consider that the lower the degree of positivity is, the higher the degree of negativity is



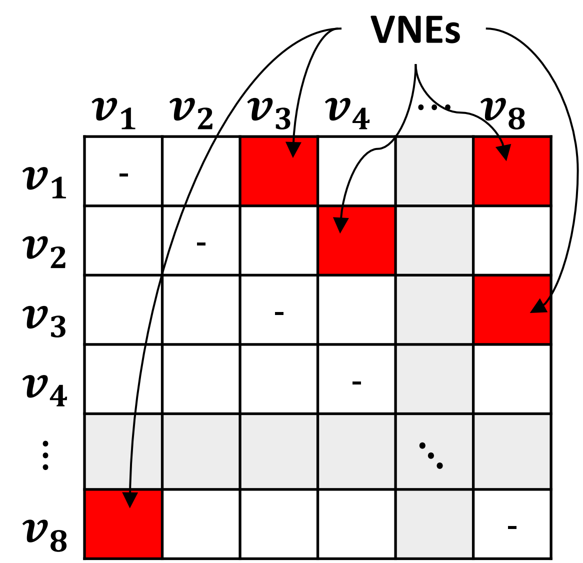
STEP 2: Selecting VNEs

- **Propose two strategies: global/local selection**
- **Global selection**
  - Select VNEs with high degrees of negativity among all potential VNEs (i.e., non-existent edges)

	v_1	v_2	v_3	v_4	$\cdots$	v_8
v_1	-	0.4	0.8	0		0.7
v_2	0	-	0	0.5		0.2
v_3	0.3	0.1	-	0.3		0.9
v_4	0	0	0.4	-		0.5
$\vdots$					$\ddots$	
v_8	0.7	0	0	0.2		-

Potential VNEs	
$v_3 \to v_8$	0.9
$v_1 \to v_3$	0.8
$v_8 \to v_1$	0.7
$v_1 \to v_8$	0.7
$v_2 \to v_4$	0.5
$v_4 \to v_8$	0.5
$\vdots$	$\vdots$

(1) Sorting in descending order of the degree of negativity

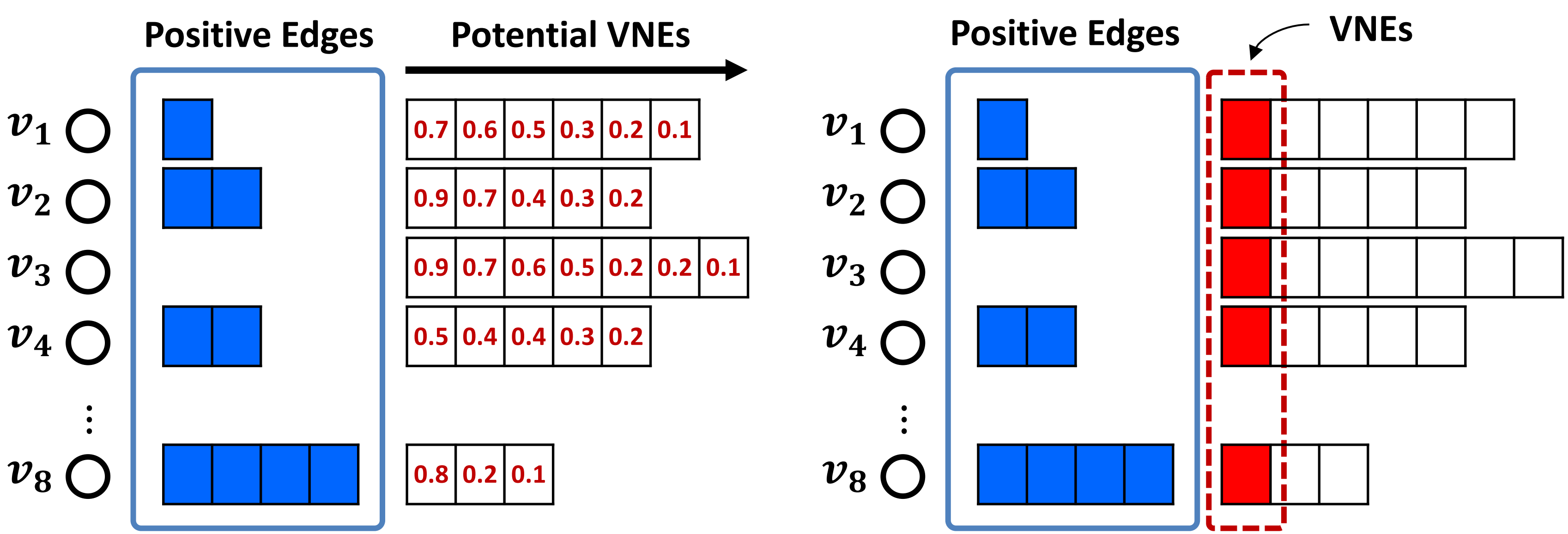


(2) Selecting VNEs (e.g., a pre-defined number=5)

STEP 2: Selecting VNEs (cont'd)

☐ Local selection

- Select an equal number of VNEs with high degrees of negativity for each node



(1) Sorting in descending order of the degree of negativity

(2) Selecting VNEs per source node (e.g., a pre-defined number=1)

Step 3: Modeling a Signed Directed Network

- **Determine the total number of VNEs to be added**

$$|\mathcal{E}^-| = |\mathcal{E}^+| \times \boldsymbol{\theta}$$

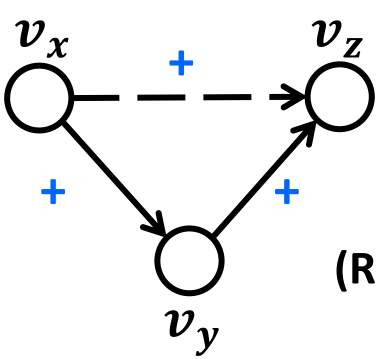
  - $|\mathcal{E}^-|$: the total number of VNEs
  - $|\mathcal{E}^+|$: the total number of positive edges
  - $\boldsymbol{\theta}$: a parameter that determines $|\mathcal{E}^-|$

- **Intuitively, it is natural to set $\boldsymbol{\theta}$ to a small value**
  - In most real-world signed networks, the number of negative edges is significantly smaller than that of positive edges
  - e.g., Wiki-election dataset
    - Positive edges : negative edges = 79% : 21%

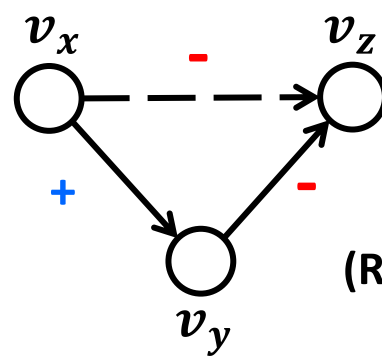
How to determine the number of VNEs?

- Deal with this issue based on a well-known property of signed networks, i.e., structural balance

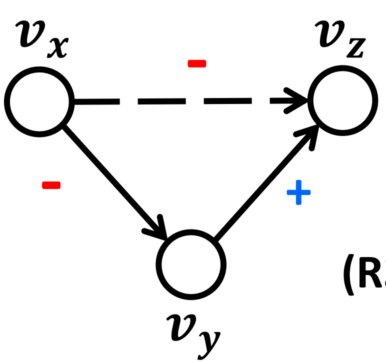
How well the edge signs in a given signed network ***follow the balance theory***?



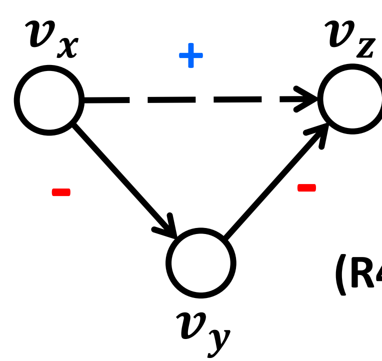
(R1) A **friend** of my **friend** is my **friend**



(R2) An **enemy** of my **friend** is my **enemy**



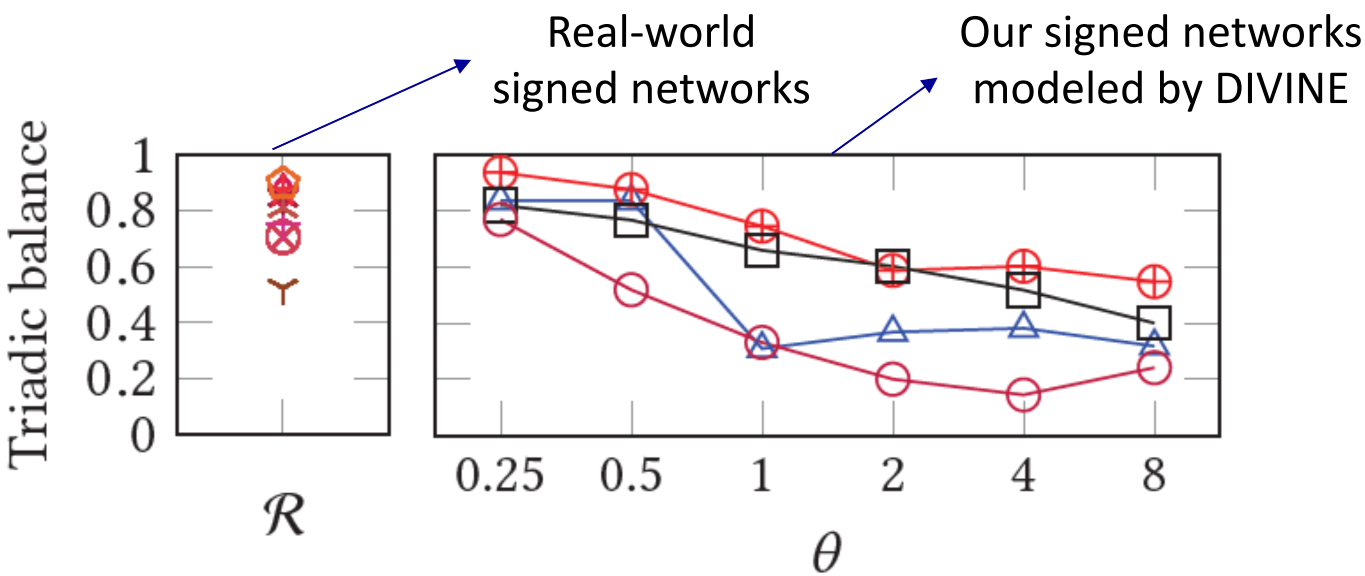
(R3) A **friend** of my **enemy** is my **enemy**



(R4) An **enemy** of my **enemy** is my **friend**

How to determine the number of VNEs?

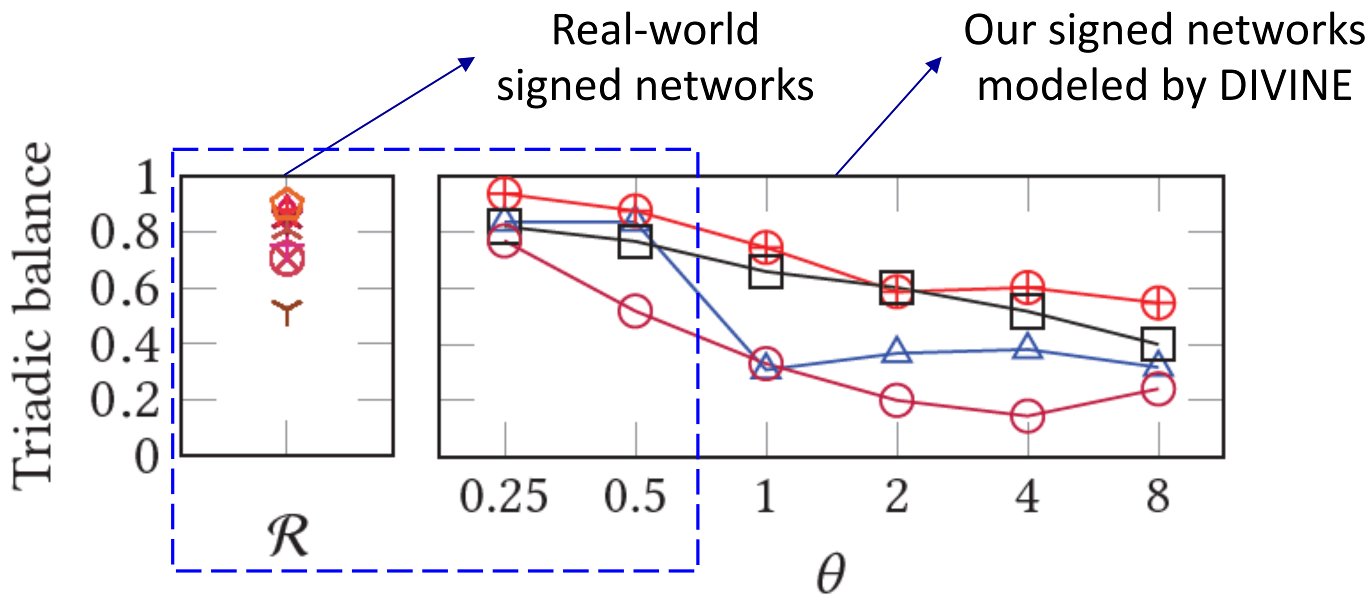
- **The effect of the parameter θ on the structural balance in our signed directed networks**



- **Observation 1**: edge signs in real-world signed networks follow the rules of balance theory well
- **Observation 2**: as θ increases, our signed networks contain more uncertain VNEs, so the edge signs do not follow well the rules of balance theory

How to determine the number of VNEs?

- **Based on this observation,**

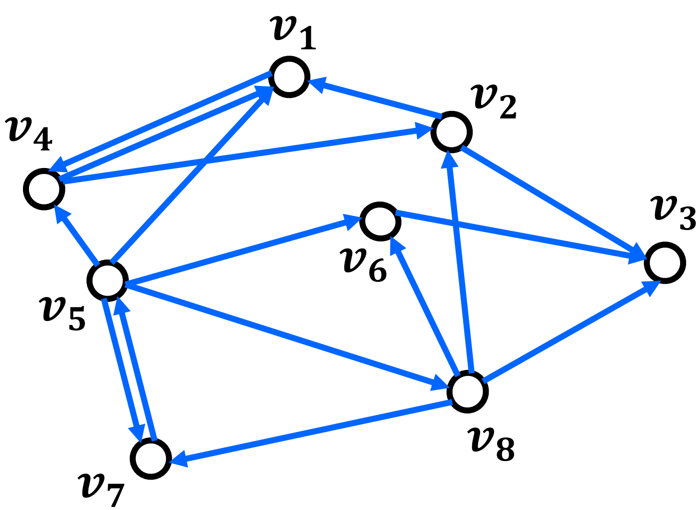


  - We set θ to a value around 0.25 or 0.5 where the structural balance of both real-world and our signed networks become similar

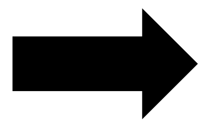
- **We will also show empirically that such values of θ lead to high accuracy of DIVINE in link prediction tasks.**

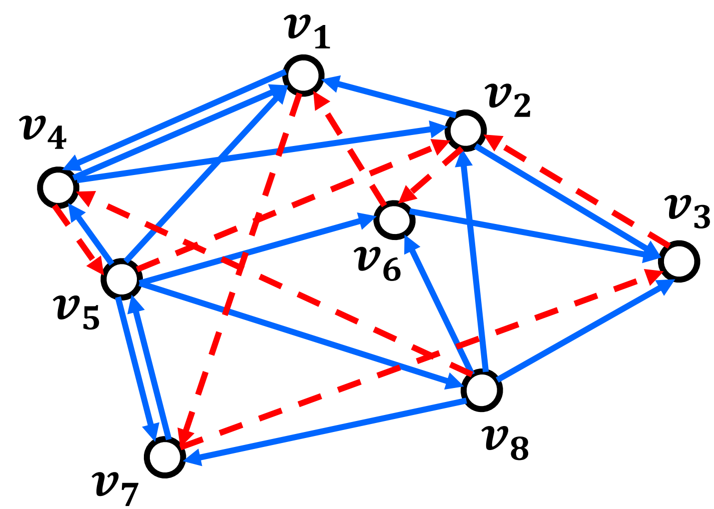
Step 3: Modeling a Signed Directed Network

- **Build a signed directed network composed of both the existent positive edges and the VNEs**



Directed Network $\mathcal{G}$

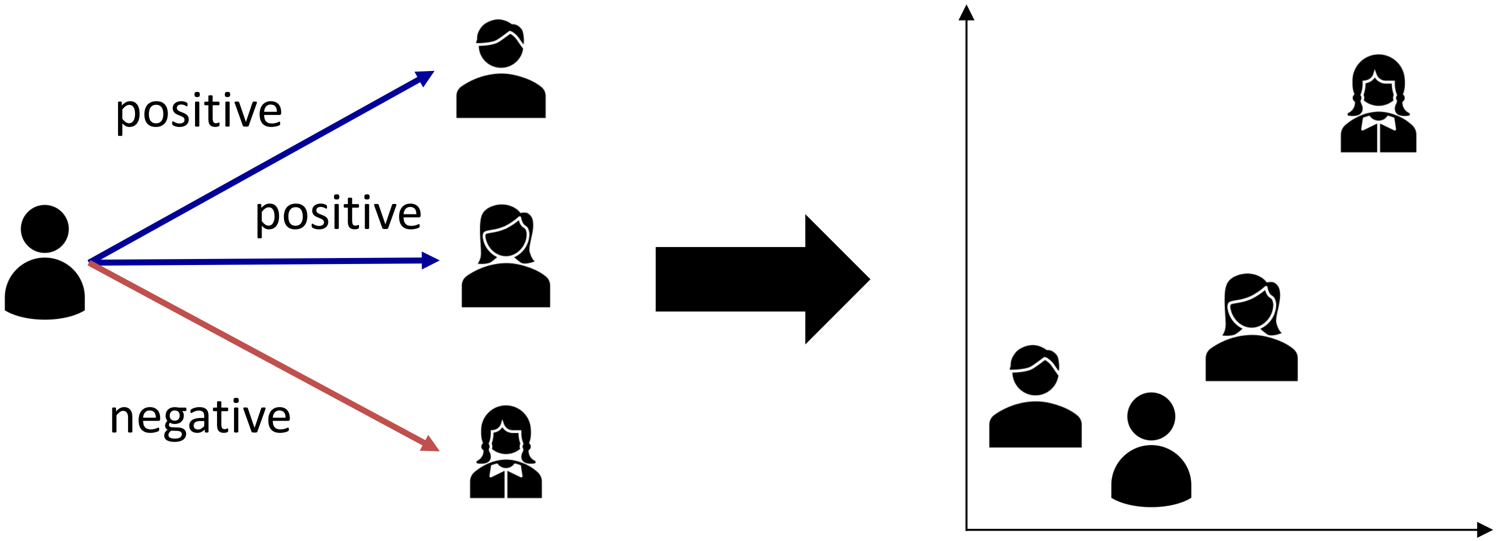




Signed Directed Network $\mathcal{S}$

Step 4: Learning Source/Target Embeddings

- **Incorporate recent signed NE methods into our DIVINE**
  - Nodes with the positive edges to be close to each other
  - Nodes with the negative edges to be distant from each other



DIVINE can be equipped with any signed NE methods!

Experimental Setup

Datasets

Datasets	GNU	Wiki-Vote	JUNG	EAT
Nodes	6,301	7,115	6,120	23,132
0 out-degree	59.35%	15.21%	1.35%	63.54%
0 in-degree	4.11%	64.49%	66.43%	2.16%
Edges	20,777	103,689	50,535	312,320
Reciprocity	0.00%	5.64%	0.90%	9.50%
Density	0.05%	0.20%	0.13%	0.06%
Types	P2P	Election	Software	Word

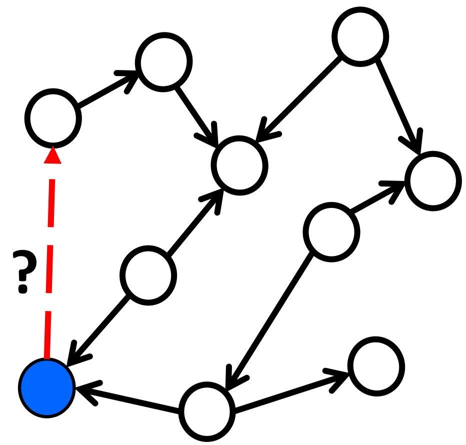
- Gnutella (GNU): a peer-to-peer network
- Wiki-Vote: an online voting network
- JUNG: a software class dependency network
- Edinburgh Associative Thesaurus (EAT): a lexical network

Experimental Setup

- Two variants of DIVINE
  - DIVINE-I employing SIDE [WWW'18]
  - DIVINE-T employing STNE [ICDM'19]
- Nine competitors
  - 3 undirected NE methods
    - DeepWalk [KDD'14]
    - LINE [WWW'15]
    - Node2Vec [KDD'16]
  - 6 directed NE methods
    - APP [AAAI'17]
    - ATP [AAAI'19]
    - NERD [ECML-PKDD'19]
    - GravityAE [CIKM'19]
    - GravityVAE [CIKM'19]
    - DiGCN [NeurIPS'20]

Evaluation Task: Link Prediction (LP)

- **How accurately we can predict the directed edges removed from the input directed network?**



- **Evaluation protocol**
  - Split the edges into training (80%) and test (20%) sets
    - Consider the existent edges as positive examples
    - Consider the same number of randomly-sampled non-existent edges as negative examples
  - Measure classification accuracy using *area under curve* (AUC)

LP Task for Directed Networks

- **How accurately the directions of the unidirectional edges in the input network can be predicted?**

- **Evaluation protocol (sampling negative examples)**
  - Sample *k*% of the unidirectional positive examples and consider the edges with the opposite directions as negative examples
  - Sample the remaining (100-*k*)% of negative examples uniformly at random among non-existent edges

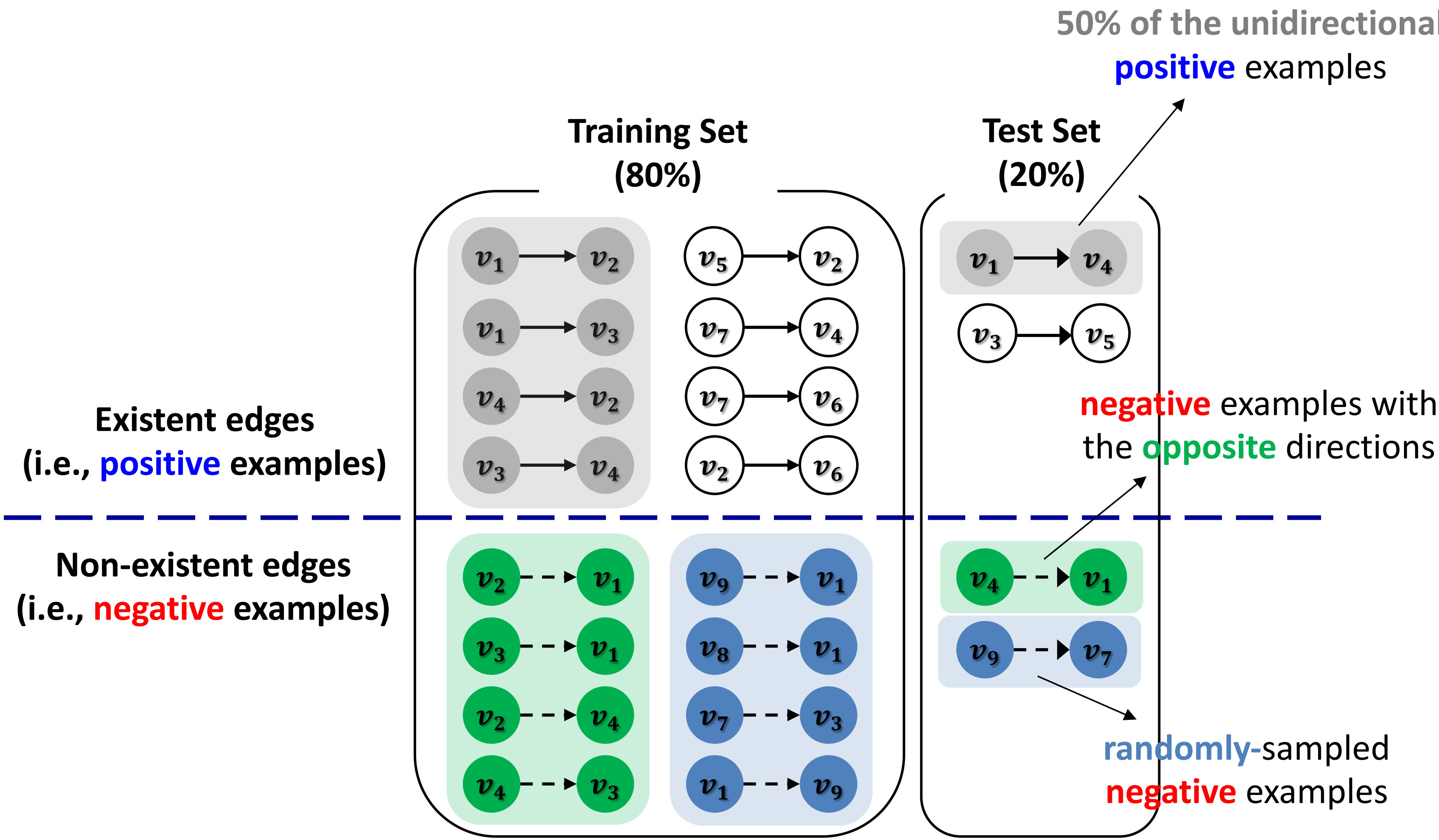
Three types of LP task according to the ratio (i.e., *k*%)

(1) k=0, Uniform LP (U-LP) (2) k=50, Mixed LP (M-LP)

(3) k=100, Biased LP (B-LP)

LP Task for Directed Networks

Example of M-LP (i.e., *k*=50)



Questions to Be Answered

- RQ1: How should the degree of negativity be inferred in DIVINE?
- RQ2: How should the locations of VNEs be decided in DIVINE?
- RQ3: How should VNEs be distributed to nodes in DIVINE?
- RQ4: How many VNEs should be added in DIVINE?
- RQ5: Does DIVINE outperform its competitors for directed NE?
- RQ6: Is DIVINE effective for embedding low-degree nodes?

Results for RQ2

☐ Effectiveness of the local selection strategy

Datasets	GNU	Wiki-Vote	JUNG	EAT
DIVINE(Global)	0.923	0.839	0.978	0.802
DIVINE(Local)	**0.943**	**0.966**	**0.994**	**0.917**
DIVINE(Local$_{vari}$)	0.920	0.838	0.986	0.813

Employing the *global selection*

Employing the *local selection*

- DIVINE(Global) vs DIVINE(Local)
  - ☐ Giving VNEs to all nodes (i.e., local) is more beneficial than giving those to only a small fraction of nodes (i.e., global)
  - ☐ Found that DIVINE(Global) added VNEs to only 35%, 44%, 53%, and 34% of nodes in GNU, Wiki-Vote, JUNG, and EAT, respectively

Results for RQ2 (cont'd)

Effectiveness of the local selection strategy

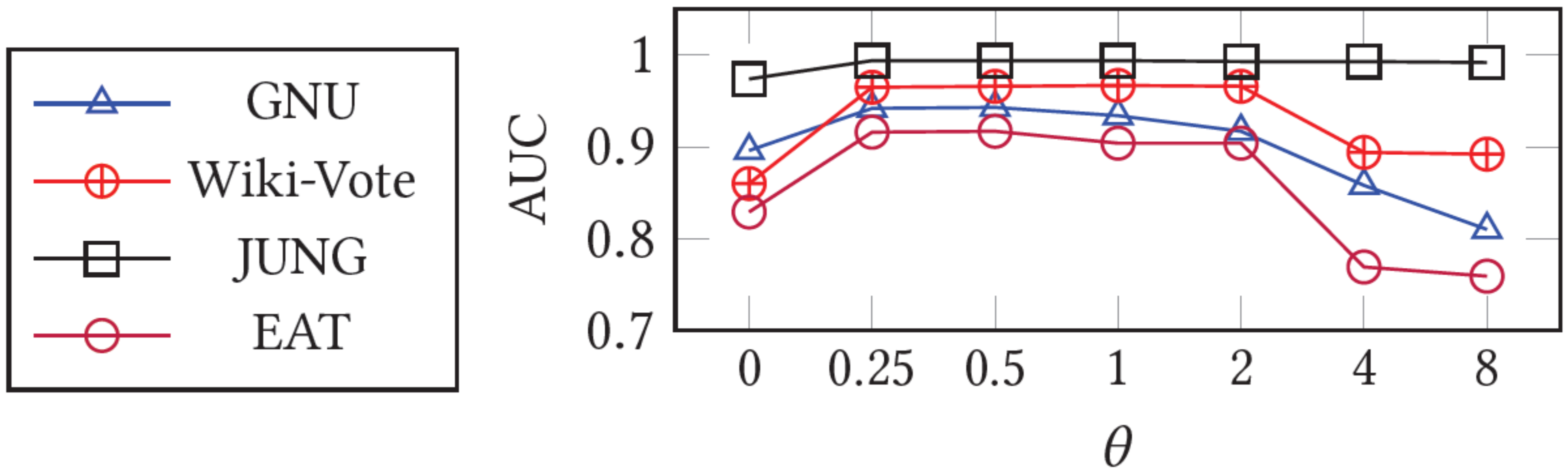
Datasets	GNU	Wiki-Vote	JUNG	EAT
DIVINE(Global)	0.923	0.839	0.978	0.802
DIVINE(Local)	**0.943**	**0.966**	**0.994**	**0.917**
DIVINE(Local$_{vari}$)	0.920	0.838	0.986	0.813

Employing the *local selection,* but **not selecting** VNEs from **zero out-degree** nodes

- DIVINE(Local) vs DIVINE(Local$_{vari}$)
  - Giving VNEs to all nodes including zero out-degree nodes effectively mitigates the lack of information

Results for RQ4

☐ Accuracy changes with varying θ



- DIVINE achieves the best AUC when $0.25 \leq \theta \leq 0.5$, which is similar to that of the structural balance
- Setting θ so that a signed directed network follows the rules of balanced theory well helps improve the AUC of DIVINE

Results for RQ5

☐ Comparison with nine competitors

Datasets	Types	Undirected NE			Directed NE					
		DeepWalk	Node2Vec	LINE	APP	GravityAE	GravityVAE	NERD	ATP	DiGCN
GNU	U-LP	0.644±0.005	0.639±0.005	0.710±0.003	0.617±0.006	0.634±0.013	0.723±0.005	0.773±0.003	0.758±0.002	0.768±0.002
	M-LP	0.618±0.007	0.600±0.005	0.772±0.004	0.606±0.003	0.648±0.016	0.750±0.007	0.809±0.006	0.813±0.004	0.836±0.003
	B-LP	0.654±0.012	0.679±0.008	0.859±0.005	0.634±0.007	0.710±0.017	0.822±0.008	0.851±0.007	0.877±0.004	0.917±0.002
Wiki-Vote	U-LP	0.890±0.002	0.880±0.003	0.864±0.007	0.823±0.002	0.871±0.008	0.906±0.002	0.901±0.006	0.824±0.004	0.826±0.001
	M-LP	0.883±0.002	0.894±0.002	0.886±0.002	0.676±0.004	0.878±0.017	0.905±0.005	0.890±0.007	0.891±0.002	0.850±0.002
	B-LP	0.922±0.002	0.944±0.002	0.944±0.001	0.686±0.006	0.922±0.017	0.950±0.005	0.897±0.007	0.966±0.001	0.917±0.002
JUNG	U-LP	0.880±0.009	0.948±0.003	0.936±0.003	0.939±0.002	0.946±0.039	0.954±0.002	0.955±0.002	0.951±0.002	0.955±0.001
	M-LP	0.902±0.007	0.956±0.003	0.957±0.002	0.950±0.002	0.944±0.033	0.968±0.003	0.963±0.002	0.968±0.002	0.971±0.002
	B-LP	0.950±0.006	0.982±0.001	0.989±0.001	0.930±0.001	0.976±0.027	0.991±0.002	0.979±0.001	0.990±0.001	0.994±0.001
EAT	U-LP	0.831±0.001	0.832±0.002	0.824±0.001	0.772±0.001	0.836±0.009	0.839±0.004	0.864±0.002	0.855±0.002	0.831±0.001
	M-LP	0.682±0.001	0.759±0.001	0.827±0.001	0.701±0.001	0.791±0.033	0.815±0.001	0.825±0.002	0.882±0.001	0.860±0.001
	B-LP	0.614±0.001	0.819±0.001	0.863±0.001	0.630±0.002	0.838±0.029	0.851±0.003	0.802±0.002	0.915±0.001	0.901±0.001

- No single competitor consistently outperforms the other competitors

Results for RQ5

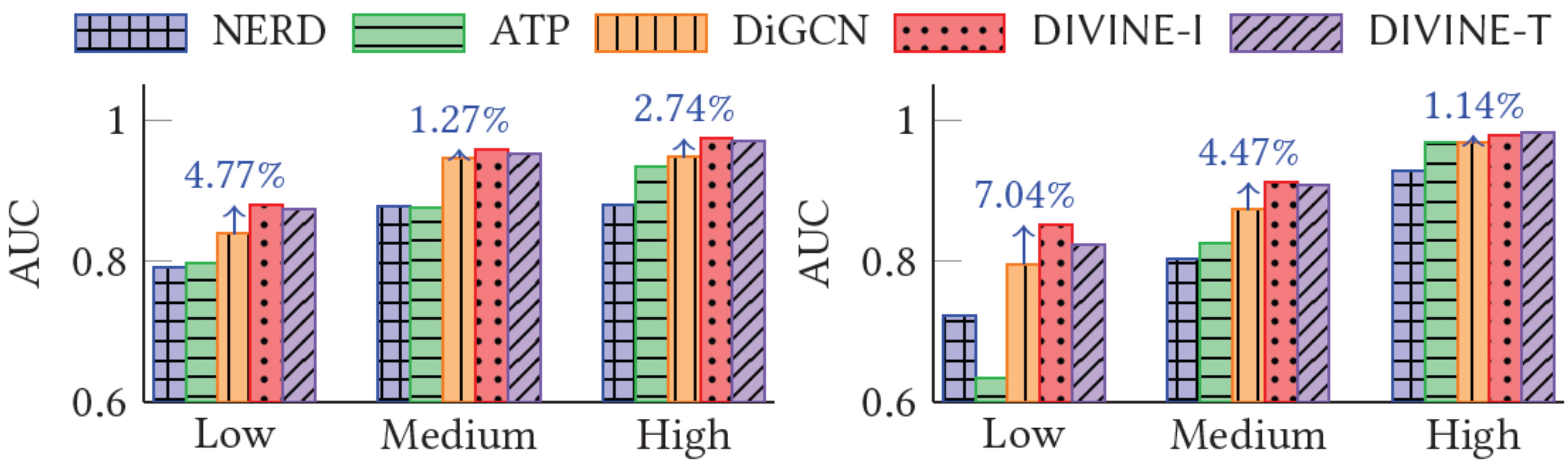
☐ Comparison with nine competitors

Datasets	Types	Directed NE APP	GravityAE	GravityVAE	NERD	ATP	DiGCN	DIVINE-I	DIVINE-T
GNU	U-LP	0.617±0.006	0.634±0.013	0.723±0.005	0.773±0.003	0.758±0.002	0.768±0.002	0.784±0.006	**0.798±0.002**
	M-LP	0.606±0.003	0.648±0.016	0.750±0.007	0.809±0.006	0.813±0.004	0.836±0.003	**0.858±0.010**	0.857±0.002
	B-LP	0.634±0.007	0.710±0.017	0.822±0.008	0.851±0.007	0.877±0.004	0.917±0.002	**0.943±0.008**	0.937±0.003
Wiki-Vote	U-LP	0.823±0.002	0.871±0.008	0.906±0.002	0.901±0.006	0.824±0.004	0.826±0.001	0.910±0.002	**0.929±0.001**
	M-LP	0.676±0.004	0.878±0.017	0.905±0.005	0.890±0.007	0.891±0.002	0.850±0.002	0.918±0.003	**0.933±0.001**
	B-LP	0.686±0.006	0.922±0.017	0.950±0.005	0.897±0.007	0.966±0.001	0.917±0.002	0.966±0.004	**0.971±0.001**
JUNG	U-LP	0.939±0.002	0.946±0.039	0.954±0.002	0.955±0.002	0.951±0.002	0.955±0.001	0.948±0.002	**0.960±0.002**
	M-LP	0.950±0.002	0.944±0.033	0.968±0.003	0.963±0.002	0.968±0.002	0.971±0.002	0.969±0.001	**0.976±0.001**
	B-LP	0.930±0.001	0.976±0.027	0.991±0.002	0.979±0.001	0.990±0.001	0.994±0.001	0.994±0.001	**0.996±0.001**
EAT	U-LP	0.772±0.001	0.836±0.009	0.839±0.004	0.864±0.002	0.855±0.002	0.831±0.001	0.880±0.006	**0.888±0.001**
	M-LP	0.701±0.001	0.791±0.033	0.815±0.001	0.825±0.002	0.882±0.001	0.860±0.001	0.881±0.007	**0.889±0.001**
	B-LP	0.630±0.002	0.838±0.029	0.851±0.003	0.802±0.002	0.915±0.001	0.901±0.001	0.917±0.006	**0.921±0.002**

- Both versions of DIVINE significantly and consistently outperform all competitors in all LP tasks on all datasets
- DIVINE is most accurate in the task of predicting the edge directions (i.e., B-LP)

Results for RQ6

☐ Effectiveness in embedding low-degree nodes

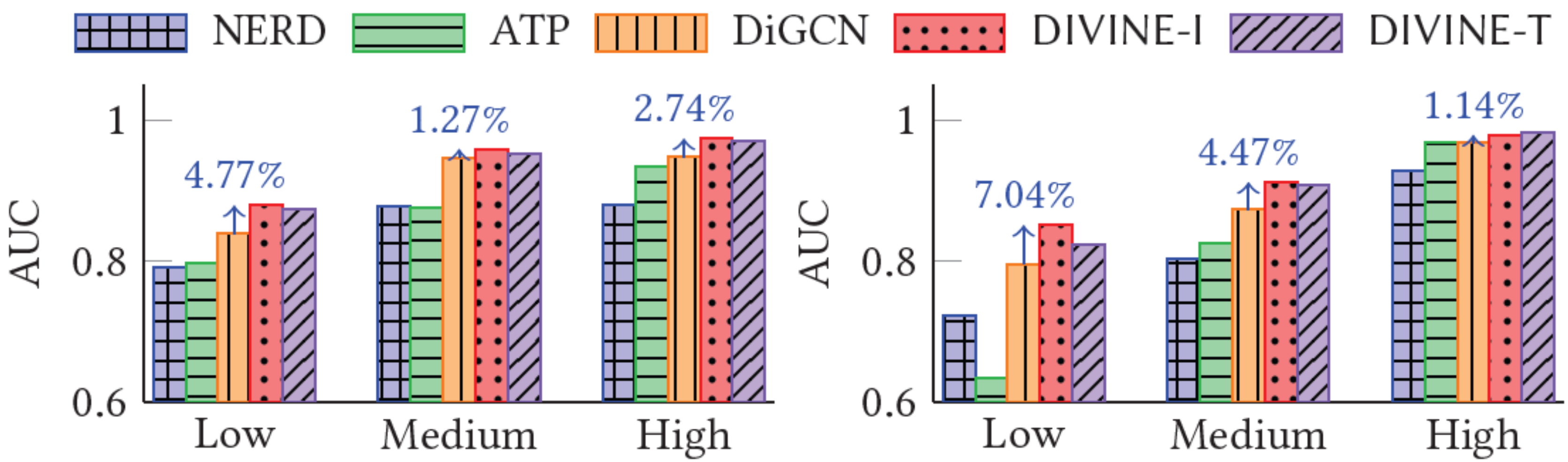


(a) Out-degree-based (b) In-degree-based

- Out-degree-based: divide all nodes in the test set into low, medium, and high groups according to their out-degree
- In-degree-based: divide all nodes in the test set into low, medium, and high groups according to their in-degree

Results for RQ6 (cont'd)

□ Effectiveness in embedding low-degree nodes



(a) Out-degree-based (b) In-degree-based

- DIVINE consistently outperform all the competitors
- The performance gain is largest in the low-degree groups
- DIVINE successfully address the lack of information about low out- and in-degree nodes

Conclusions

- **We pointed out that the existing directed NE methods face difficulties in accurately preserving asymmetric proximities between nodes in a sparse network**
- **Under DIVINE, we proposed three ideas to selectively add VNEs**
  - Inferring the degree of negativity
  - Using the local selection strategy to distribute VNEs to all nodes
  - Determining the number of VNEs based on the theory of structural balance
- **DIVINE significantly outperforms its 9 state-of-the-art competitors in 3 LP tasks on 4 real-world datasets**

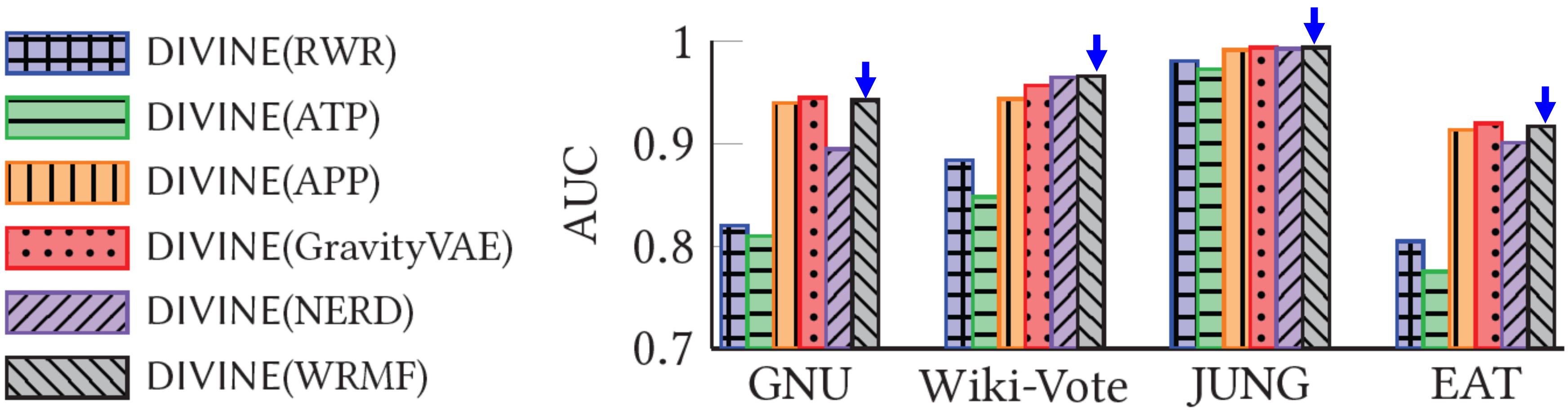
Thank You !

Contact: hy40@Illinois.edu

Appendix

Results for RQ1

☐ Comparisons of methods for inferring the degree of negativity



- When it is equipped with WRMF, DIVINE consistently achieves high AUC in all datasets

Results for RQ3

- **Effectiveness of adding an equal number of VNEs to each source node**

Datasets	GNU	Wiki-Vote	JUNG	EAT
DIVINE(Prop.)	0.922	0.862	0.976	0.806
DIVINE(InverseProp.)	0.915	0.951	**0.994**	0.760
DIVINE(Uniform)	**0.943**	**0.966**	**0.994**	**0.917**

  - DIVINE(Prop.) sets the number of VNEs from each node **proportionally to its out-degree**
  - DIVINE(InverseProp.) sets the number of VNEs from each node **inverse proportionally to its out-degree**
  - DIVINE(Uniform) sets **an equal number** of VNEs to all nodes

Results for RQ3 (cont'd)

☐ Effectiveness of adding an equal number of VNEs to each source node

Datasets	GNU	Wiki-Vote	JUNG	EAT
DIVINE(Prop.)	0.922	0.862	0.976	0.806
DIVINE(InverseProp.)	0.915	0.951	**0.994**	0.760
DIVINE(Uniform)	**0.943**	**0.966**	**0.994**	**0.917**

- DIVINE(Uniform) consistently outperforms the others
- Treating all source nodes equally by adding an equal number of VNEs to them helps learn accurate embeddings most

Why Virtual Negative Edges? (cont'd)

☐ Comparisons of several methods for adding VEs

Datasets	Types	Based on GDC	Based on the degree of negativity		
		VPEs	VPEs	VNEs	VPEs+VNEs
GNU	U-LP	0.756	0.778	0.784	0.788
	M-LP	0.822	0.846	0.858	0.859
	B-LP	0.911	0.934	0.943	0.944
Wiki-Vote	U-LP	0.871	0.879	0.910	0.911
	M-LP	0.875	0.892	0.918	0.916
	B-LP	0.903	0.954	0.966	0.967
JUNG	U-LP	0.936	0.951	0.948	0.951
	M-LP	0.942	0.969	0.969	0.970
	B-LP	0.979	0.992	0.994	0.994
EAT	U-LP	0.820	0.849	0.880	0.881
	M-LP	0.830	0.845	0.881	0.882
	B-LP	0.867	0.880	0.917	0.917

- Adding VNEs achieves superior AUC over adding VPEs
- Adding VPEs in addition to VNEs resulted in marginal additional gains.

Inferring the Degree of Negativity (cont'd)

Equations

- Objective function

$$\mathcal{L}(\mathbf{P},\mathbf{Q}) = \sum_{i,j} w_{ui}\left\{\left(a_{ij} - \mathbf{P}_{i(\cdot)}(\mathbf{Q}_{j(\cdot)})^{\top}\right)^2 + \lambda\left(\left\|\mathbf{P}_{i(\cdot)}\right\|_F^2 + \left\|\mathbf{Q}_{j(\cdot)}\right\|_F^2\right)\right\}$$

- Updates elements in the matrices $\mathbf{P}$ and $\mathbf{Q}$

$$\mathbf{P}_{i(\cdot)} = \mathbf{A}_{i(\cdot)}\widetilde{\mathbf{W}}_{i(\cdot)}\mathbf{Q}\left\{\mathbf{Q}^{\top}\widetilde{\mathbf{W}}_{i(\cdot)}\mathbf{Q} + \lambda\left(\sum_j w_{ij}\right)\mathbf{I}\right\}^{-1}$$

$$\mathbf{Q}_{j(\cdot)} = (\mathbf{A}_{(\cdot)j})^{\top}\widetilde{\mathbf{W}}_{(\cdot)j}\mathbf{P}\left\{\mathbf{P}^{\top}\widetilde{\mathbf{W}}_{(\cdot)j}\mathbf{P} + \lambda\left(\sum_i w_{ij}\right)\mathbf{I}\right\}^{-1}$$

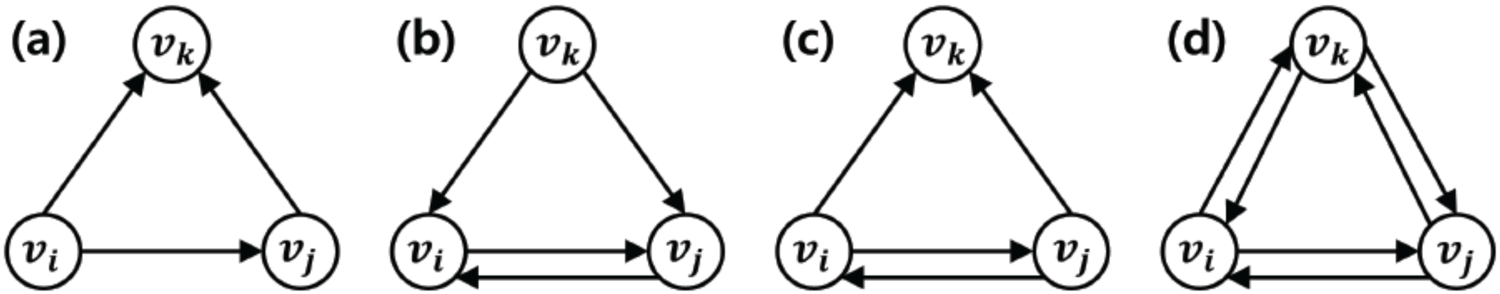
  - $\widetilde{\mathbf{W}}_{i(\cdot)}$ is a diagonal matrix with elements of $\mathbf{W}_{i(\cdot)}$ on the diagonal
  - Matrix **I** is an identity matrix

- Final value

  - $\widehat{\mathbf{A}} \approx \mathbf{A} = \mathbf{P}\mathbf{Q}^{\top}$ ➡ $\boldsymbol{x_{ij} = 1 - \frac{\hat{a}_{ij} - \|\widehat{\mathbf{A}}\|_{min}}{\|\widehat{\mathbf{A}}\|_{max} - \|\widehat{\mathbf{A}}\|_{min}}}$

Triadic Balance [Aref et al. Sci. Rep.'20]

- **New measure that assesses the structural balance of the signed "directed" network**



  - Collect all the transitive triads consisting of at least one or multiple triangles where the directions of three edges satisfy the transitivity
  - Measure the ratio of balanced ones among all the collected transitive triads

* Triad: a set of three nodes with at least one directed edge between each pair of them

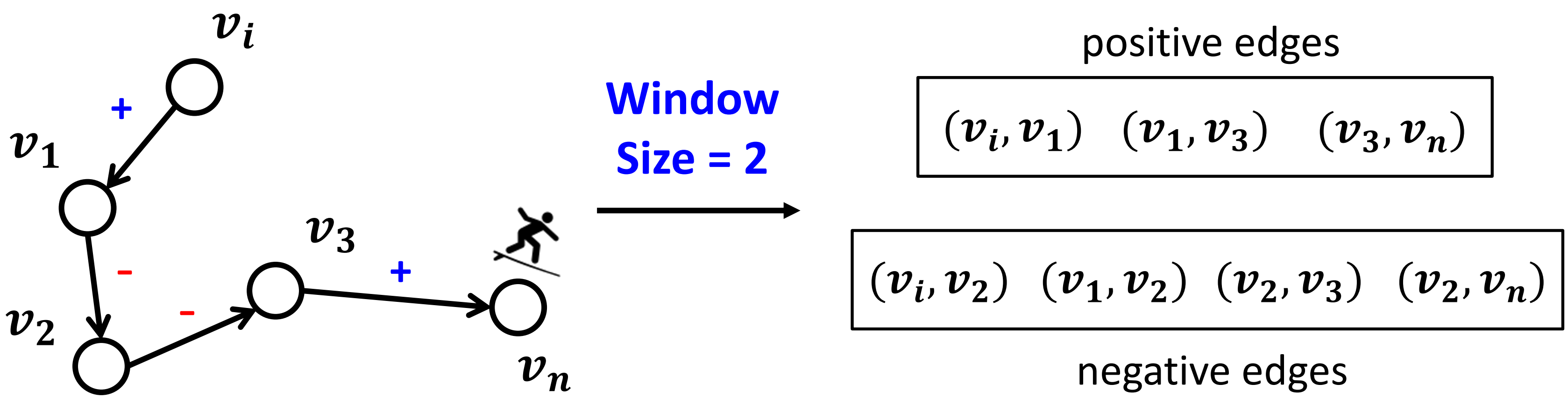
How to determine the number of VNEs?

Real-world signed networks

Datasets	Reddit	Wiki-election	Bitcoin OTC	Bitcoin Alpha	Highland	College-A	College-B	College-C
Nodes	18,313	7,118	5,881	3,783	16	21	17	20
Edges	120,792	103,675	35,592	24,186	116	94	83	81
Positive Edges	111,891	81,318	32,029	22,650	58	51	41	41
Negative Edges	8,901	22,357	3,563	1,536	58	43	42	40

- **Reddit** represents connections between users of two subreddits from Jan 2014 to April 2017
- **Wiki-election** contains approval/disapproval votes for electing admins in Wikipedia from 2003 to 2013
- **Bitcoin OTC** and **Bitcoin Alpha** represent the record of reputation/trust of users on a Bitcoin trading platform
- **Highland** represents alliance structure among three tribal groups
- **College-A**, **College-B**, and **College-C** represent preference rankings of a group of girls in an Eastern college

SIDE [Kim et al. WWW'18]



- **Perform a directed random walk** that start from each node v_i by following out-going edges
- **Generate a sequence** $\{v_i \rightarrow v_1 \rightarrow \cdots \rightarrow v_n\}$ with edge signs
- **Sample each directed node pair** (v_i, v_j) where v_i (i.e., source) precedes v_j (i.e., target) in the sequence **within a window size**
- **Determine the sign of each** (v_i, v_j) by combining the edge signs in the sequence from v_i to v_j **based on balance theory**

SIDE [Kim et al. WWW'18] (cont'd)

$$\mathcal{L}(\mathbf{f},\mathbf{g}) = \sum_{(v_i,v_j)\in\mathcal{O}} \left[-\log \mathcal{P}\left(\boldsymbol{v_i},\boldsymbol{v_j}\right) + \sum_{k=1}^{\alpha} -\log \mathcal{P}(\boldsymbol{v_i},\boldsymbol{v_k}) \right] + \mathcal{R}(\delta)$$

- **For each (v_i, v_j) with a positive sign,**
  - Maximize the proximity between $\boldsymbol{v_i}$'s source embedding and $\boldsymbol{v_j}$'s target embedding
- **For each (v_i, v_j) with a negative sign,**
  - Minimize the proximity between $\boldsymbol{v_i}$'s source embedding and $\boldsymbol{v_j}$'s target embedding