Soongsil University

Time-aware Random Walk Diffusion to Improve Dynamic Graph Learning

Jinhong Jung
Soongsil University

KJDB 2023

Jong-whi Lee, and *Jinhong Jung*. "Time-aware random walk diffusion to improve dynamic graph learning." *AAAI 2023*

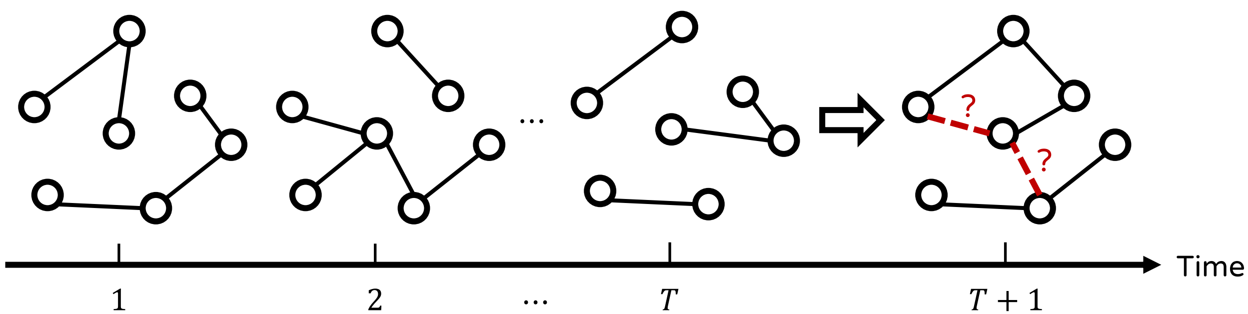
Outline

- Introduction
- Motivation
- Proposed Method
- Experiments
- Conclusion

Dynamic Graph Learning (1)

❑ Real-world graphs change over time!

- Represented as a temporal sequence of graph snapshots
  - Social networks, citation networks, web graphs, etc.
- Learning node representations on a dynamic graph is crucial in **temporal link prediction & node classification**
  - Extended to traffic forecasting & temporal knowledge completion

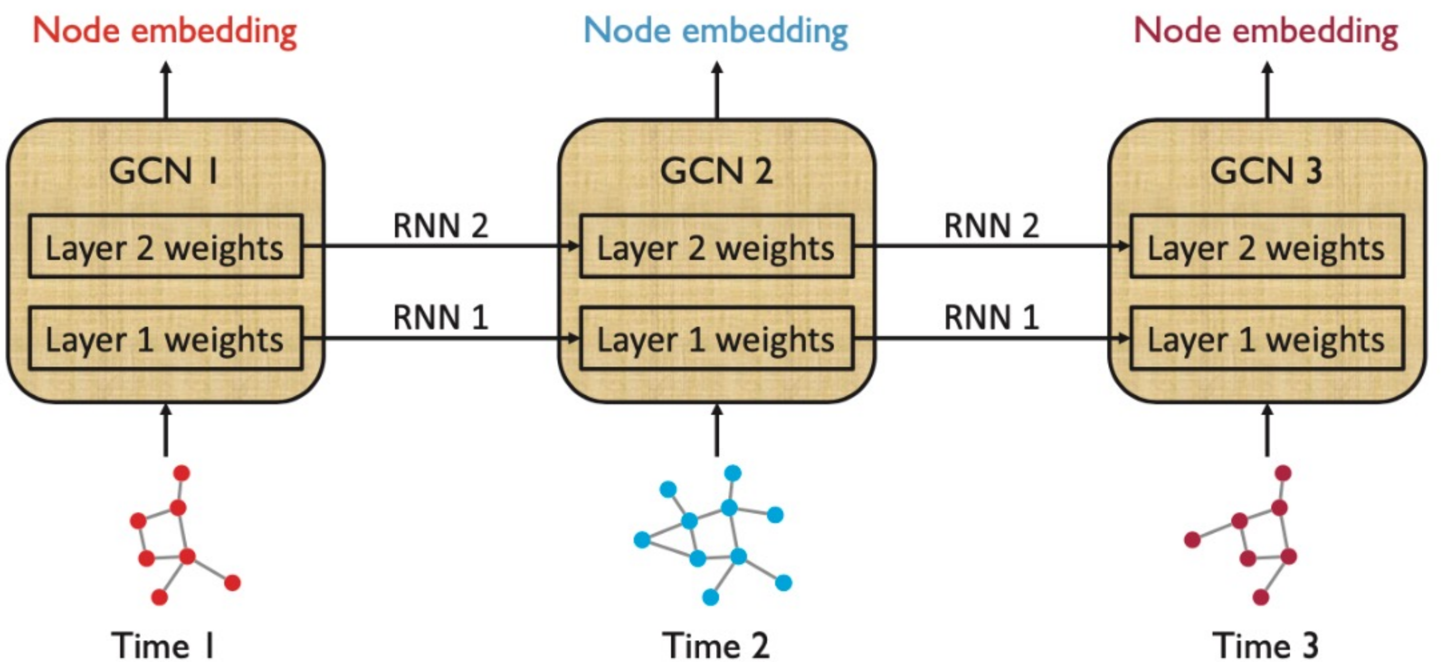


Temporal link prediction on a dynamic graph (discrete-time)

Dynamic Graph Learning (2)

❑ Dynamic Graph Neural Networks

- Combined with GCNs and RNNs (e.g., GCRN, EvolveGCN)



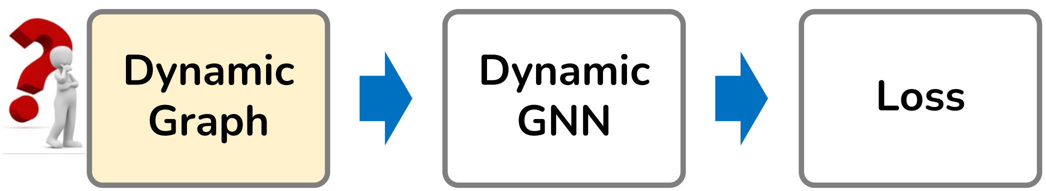
Overview of EvolveGCN

[Pareja et al., AAAI20]

Research Question

❑ **How can we augment a dynamic graph to improve dynamic graph learning?**

- Each graph snapshot is extremely sparse (i.e., few edges)
  - Not good for graph convolution
- Data augmentation is essential for ML models
  - How to augment such a dynamic graph?



How to effectively augment
the dynamic graph that changes over time?

Problem Definition

❑ Dynamic graph learning aims to learn

$$\mathbf{H}_t = \mathcal{F}_\Theta(\boldsymbol{A}_t, \mathbf{F}_t, \mathbf{H}_{t-1})$$

- $\mathcal{F}_\Theta$ is a dynamic GNN model with parameter Θ
- $\boldsymbol{A}_t$ is an adjacency matrix of a dynamic graph $\mathcal{G}$ at time t
- $\mathbf{F}_t$ and $\mathbf{H}_t$ are node features and hidden embeddings, resp.

❑ Dynamic graph augmentation

- **Input**: a sequence $\{\boldsymbol{A}_1, \cdots, \boldsymbol{A}_T\}$ of adjacency matrices in $\mathcal{G}$
- **Output**: a new sequence $\{\boldsymbol{\mathcal{X}}_1, \cdots, \boldsymbol{\mathcal{X}}_T\}$ of augmented adjacency matrices
  - **We want those new adjacency matrices to improve the performance of any dynamic GNN**

Previous Approaches

- **Most existing augmentations mainly transform spatial structure of a single static graph**
  - Drop-based methods
    - e.g., randomly drop a few of edges at each epoch
  - Diffusion-based methods
    - e.g., add new edges weighted by graph diffusion such as RWR

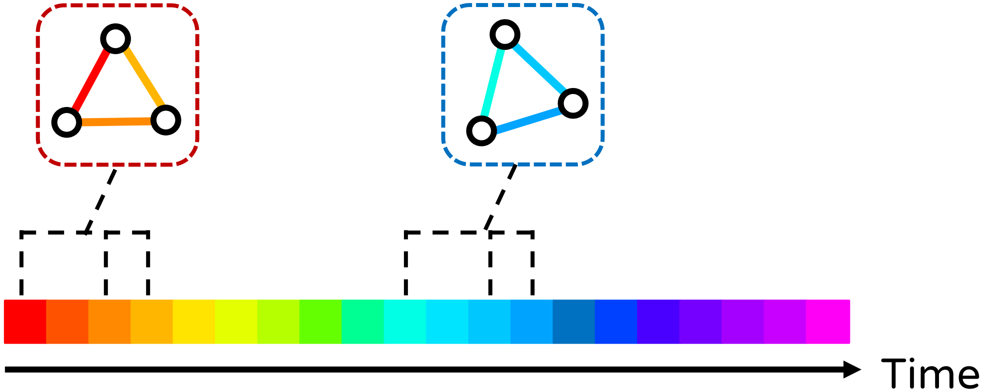
- **However, they are unsuitable for dynamic graphs**
  - Naively applying a static method to each graph snapshot could not capture temporal dynamics!

Outline

- ❑ Introduction
- ❑ Motivation
- ❑ Proposed Method
- ❑ Experiments
- ❑ Conclusion

Motivation (1)

- **Dynamic graph augmentation needs to**
  - Consider **temporal dynamics** as well as **spatial structure**
  - Inspired from **temporal and spatial localities** in graphs
- **Temporal locality**
  - Objects (e.g., triangle) tend to be **more affected by more recent edges than older ones** in dynamic graphs
    - e.g., triangles with edges close in time than with edges far in time

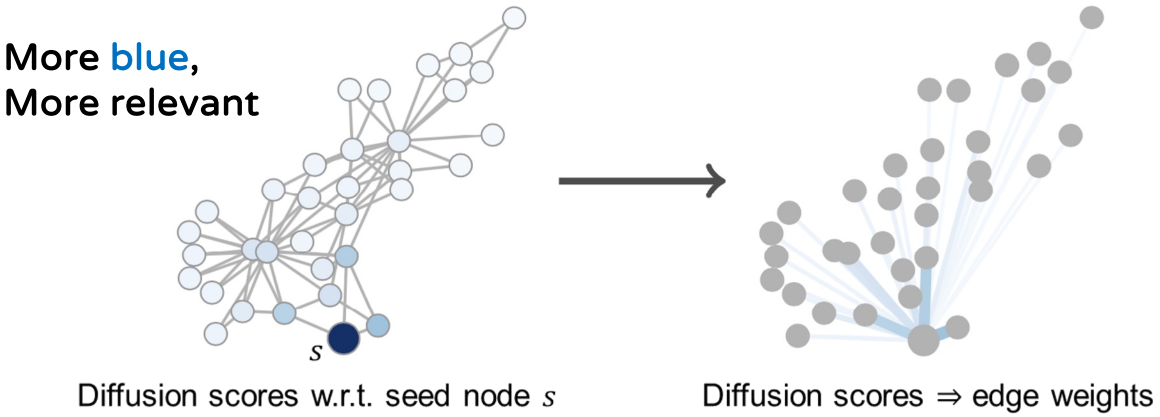


[Shin et al., ICDM17]

Motivation (2)

❑ Spatial locality

- Objects (e.g., node) tend to be **more affected by nearby nodes than distant ones**
- Graph diffusion enhances **spatial locality**
  - **Random Walk with Restart (RWR)** uses a random surfer who does random walk or restart from seed node s
    - Node-to-node proximity scores are **spatially localized** to the seed node



[Gasteiger et al., NeurIPS19]

Research Challenges

- **Previous work ignores temporal locality**
  - However, newly augmented edges need to be more affected by more recent edges
- **Graph diffusion enhances spatial locality**
  - However, it leads to a fully dense score matrix that can degrade computational efficiency

Challenges:

- **C1.** How can we augment the temporal locality as well as the spatial locality (⇒ spatio-temporal locality)?
- **C2.** How can we avoid to generate dense matrices while preserving enhanced data?

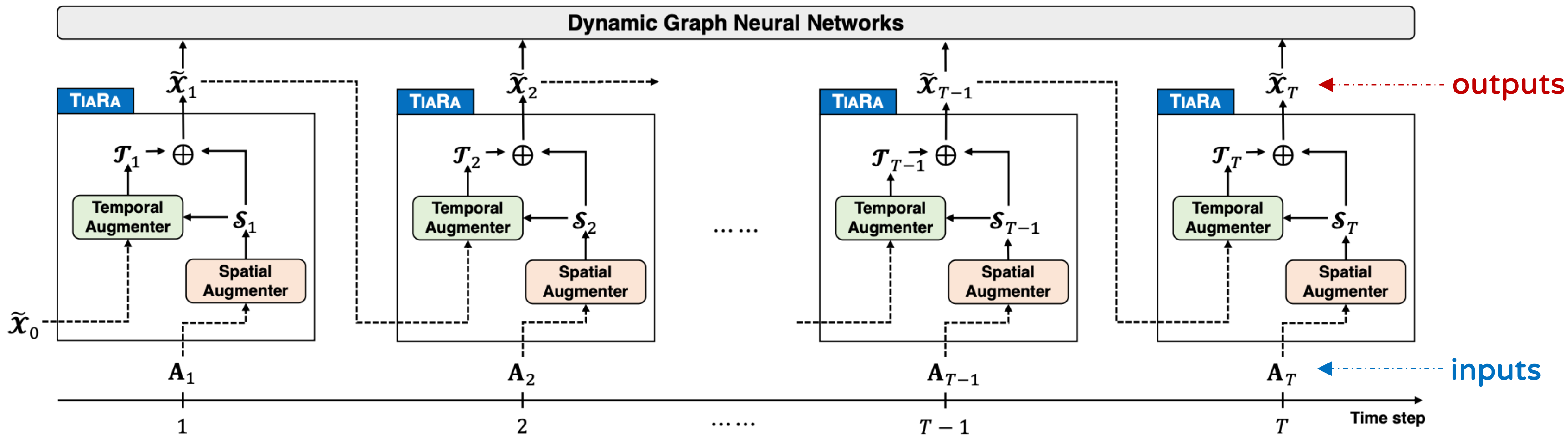
Outline

- ❑ Introduction
- ❑ Motivation
- ❑ **Proposed Method**
- ❑ Experiments
- ❑ Conclusion

Proposed Method

❑ TiaRa (Time-aware Random Walk Diffusion)

- **Aims to enhance spatio-temporal locality!**



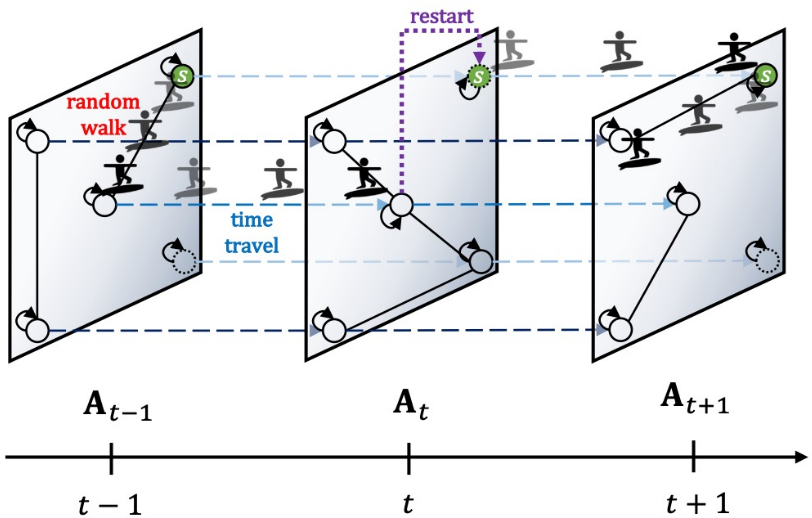
❑ Our approaches

- 1) Make an RWR's surfer time-aware
- 2) Diffuse the time-aware surfer on the dynamic graph
- 3) Sparsify the diffused results for efficiency

Time-aware RWR (TRWR)

❑ Virtually connect nodes toward the future

- Then, the surfer also can travel along the time axis
  - Not backward since future (test) data should be prevented
- Leads to diffusion scores spatio-temporally localized
  - Insert new edges based on the diffusion scores!



Classical RWR [Tong et al., ICDM06]

$$\underbrace{\mathbf{x}_s}_{\text{Diffusion scores w.r.t. } s} = \underbrace{(1-\alpha)\tilde{\mathcal{A}}^\top \mathbf{x}_s}_{\text{Random walk}} + \underbrace{\alpha \mathbf{i}_s}_{\text{Restart}}$$

⇩

Time-aware RWR

$$\underbrace{\mathbf{x}_{t,s}}_{\text{Diffusion scores w.r.t. } s \text{ at time } t} = \underbrace{(1-\alpha-\beta)\tilde{\mathcal{A}}_t^\top \mathbf{x}_{t,s}}_{\text{Random walk}} + \underbrace{\alpha \mathbf{i}_s}_{\text{Restart}} + \underbrace{\beta \mathbf{x}_{t-1,s}}_{\text{Time travel}}$$

Diffusion Matrix of TRWR

Details

Diffusion matrix $\mathcal{X}_t$ is represented as:

$$\boldsymbol{\mathcal{X}}_t = (1-\gamma)\underbrace{(\boldsymbol{\mathcal{L}}_t^{\mathrm{rwr}}\boldsymbol{I}_n)}_{\substack{\text{Spatial augmenter} \\ \boldsymbol{\mathcal{S}}_t}} + \gamma\underbrace{(\boldsymbol{\mathcal{L}}_t^{\mathrm{rwr}}\boldsymbol{\mathcal{X}}_{t-1})}_{\substack{\text{Temporal augmenter} \\ \boldsymbol{\mathcal{T}}_t}}$$

- Notations
  - $\boldsymbol{\mathcal{X}}_t = \{\boldsymbol{x}_{t,s}\}$ contains diffusion scores of TRWR w.r.t. all nodes s
  - $\gamma = \beta/(\alpha+\beta)$ is a ratio of temporal locality
  - $\boldsymbol{\mathcal{L}}_t^{\mathrm{rwr}}$ is a diffusion matrix of RWR at only time t

- In other words, $\boldsymbol{\mathcal{X}}_t$ is a linear combination of $\boldsymbol{\mathcal{S}}_t$ and $\boldsymbol{\mathcal{T}}_t$

Diffusion Matrix of TRWR

Details

❑ Theorem for dynamic graph augmentation

$$\mathcal{X}_t \propto \overbrace{\underbrace{\gamma^0 \mathcal{L}_t^{\text{rwr}} + \gamma^1 \mathcal{L}_{t \leftsquigarrow t-1}^{\text{rwr}} + \cdots + \gamma^{t-2} \mathcal{L}_{t \leftsquigarrow 2}^{\text{rwr}} + \frac{\gamma^{t-1}}{1-\gamma} \boxed{\mathcal{L}_{t \leftsquigarrow 1}^{\text{rwr}}}}_{\Longleftarrow \text{ Emphasized} \qquad\qquad \text{Decayed } \Longrightarrow}}^{\text{Augmentation of spatial and temporal localities}}$$

$\mathcal{L}_{t \leftsquigarrow 1}^{\text{rwr}} = \mathcal{L}_t^{\text{rwr}} \times \mathcal{L}_{t-1}^{\text{rwr}} \times \cdots \times \mathcal{L}_1^{\text{rwr}}$

$0 < \gamma < 1$

- **Can capture temporal locality as well as spatial locality**
  - $\mathcal{L}_t^{\text{rwr}}$ indicates a matrix in which a spatial locality is enhanced
  - $\mathcal{X}_t$ is more affected by more recent data than older ones where a temporal locality is enhanced
    - Old information is decaying over time by γ (a.k.a. *temporal decay ratio*)
  - See the detailed proof in the paper!

Calculation of TRWR

❑ Exploit Power iteration method as RWR does!

$$\mathcal{X}_t = (1-\gamma)\underbrace{(\mathcal{L}_t^{\text{rwr}} \boldsymbol{I}_n)}_{\substack{\text{Spatial augmenter} \\ \mathcal{S}_t}} + \gamma\underbrace{(\mathcal{L}_t^{\text{rwr}} \mathcal{X}_{t-1})}_{\substack{\text{Temporal augmenter} \\ \mathcal{T}_t}}$$

- Core term is $\mathcal{L}_t^{\text{rwr}}$, a typical RWR score matrix which can be calculated using Power iteration method
  - Efficient if the adjacency matrix at time t is sparse

- However, both augmenters cause $\mathcal{X}_t$ to become dense, negatively impacting the computation for the next $\mathcal{X}_{t+1}$
  - Thus, we introduce further approximation for efficiency

Sparsification

- **Set elements of $\mathcal{X}_t$ less than ϵ to zero**
  - ϵ is called filtering threshold where $0 < \epsilon < 1$
  - This sparsification follows the below intuition:
    - As scores are localized, very tiny entries are unlikely to affect a graph convolution [Gasteiger et al., NeurIPS19]
  - This significantly reduces # of non-zeros of a diffusion matrix while preserving accuracy, **thereby maintaining the efficiency of Power Iteration!**

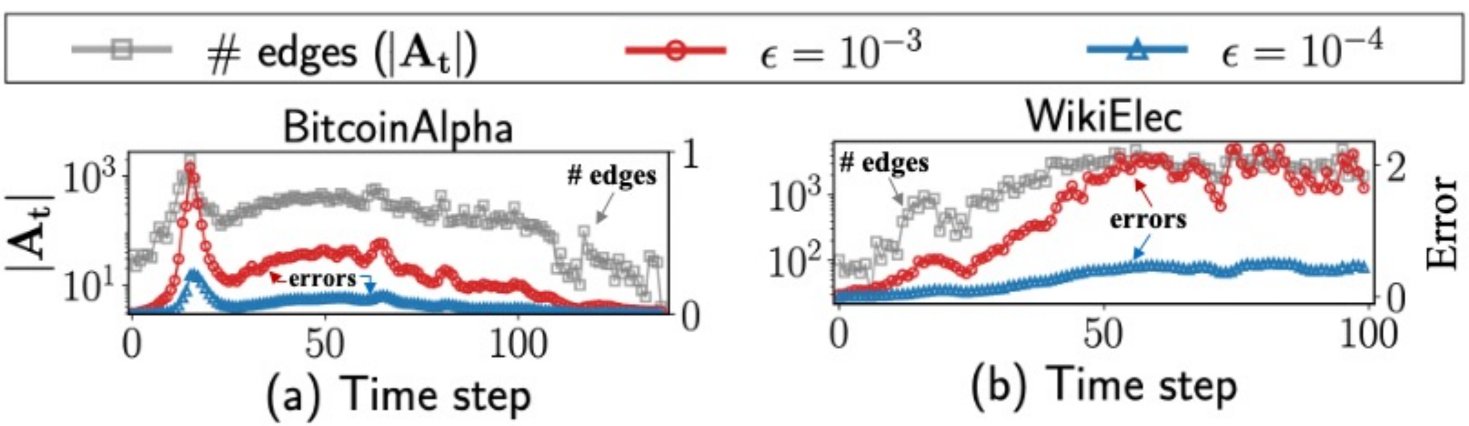
Analysis on Sparsification

ϵ: filtering threshold

❑ Analytical results of filtered $\tilde{x}_t$

- Theoretically, # of non-zeros of $\tilde{x}_t$ is $O(n/\epsilon)$
  - Where n is # of nodes, and it's much smaller than $O(n^2)$ (i.e., $\epsilon^{-1} \ll n$)

- Empirically, approximation errors don't explode over time
  - Less affected by previous errors; rather, it is ∝ # of edges



Outline

- ❑ Introduction
- ❑ Motivation
- ❑ Proposed Method
- ❑ **Experiments**
- ❑ Conclusion

Experimental Setup

Baseline augmentation methods

- DropEdge, GDC, and Merge (simply accumulating graph snapshots)

Dynamic GNN models

- GCN, GCRN and EvolveGCN (EGCN)
- Compare each GNN with and without augmentation

Datasets

	# nodes	# edges	# time steps	# labels
Datasets	n	m	T	L
BitcoinAlpha	3,783	31,748	138	2
WikiElec	7,125	212,854	100	2
RedditBody	35,776	484,460	88	2
Brain	5,000	1,955,488	12	10
DBLP-3	4,257	23,540	10	3
DBLP-5	6,606	42,815	10	5
Reddit	8,291	264,050	10	4

Temporal Link Prediction

- **Aims to predict if an edge appears in the future**
  - Augment the adjacency matrix at each time
  - Feed data from time 1 to $t-1$ into a GNN when training
  - Predict test edges at time t when evaluating

▲ improvement
▼ degradation

	AUC	BitcoinAlpha			WikiElec			RedditBody		
		GCN	GCRN	EGCN	GCN	GCRN	EGCN	GCN	GCRN	EGCN
	NONE	57.3±1.6	80.3±6.0	58.8±1.1	59.9±0.9	72.1±2.4	66.9±3.7	77.6±0.4	88.9±0.3	77.6±0.2
Augmention baseline	DROPEDGE	▼56.3±1.0	▼73.9±2.2	▼57.4±0.9	▼50.1±1.0	▼56.0±9.3	▼47.9±6.4	▼73.0±0.4	▼77.0±1.7	▼71.9±0.7
	GDC	▲57.5±1.6	▼77.3±6.5	▼57.4±1.2	▲62.8±0.8	▼67.9±1.0	▼63.1±0.7	▼74.6±0.0	▼86.4±0.3	▼73.8±0.3
	MERGE	▲66.8±2.6	▲93.1±0.4	▲61.0±9.2	▲60.6±1.7	▼68.4±3.2	▼60.7±1.3	▼69.7±0.7	▲89.8±0.5	▲80.3±0.5
	TIARA	**▲76.0±1.3**	**▲94.6±0.8**	**▲77.2±1.4**	**▲69.0±1.2**	**▲73.4±2.2**	**▲69.1±0.3**	**▲80.8±0.6**	**▲90.2±0.4**	**▲82.0±0.1**

TiaRa consistently improves the performance of dynamic GNNs, and outperforms other augmentation methods

Node Classification

❑ Aims to classify a label of a node

- A graph and features change over time
- Feed only training nodes of all time steps into a GNN
- Classify test nodes after training

▲ improvement
▼ degradation

Augmentation baseline: DropEdge, GDC, Merge

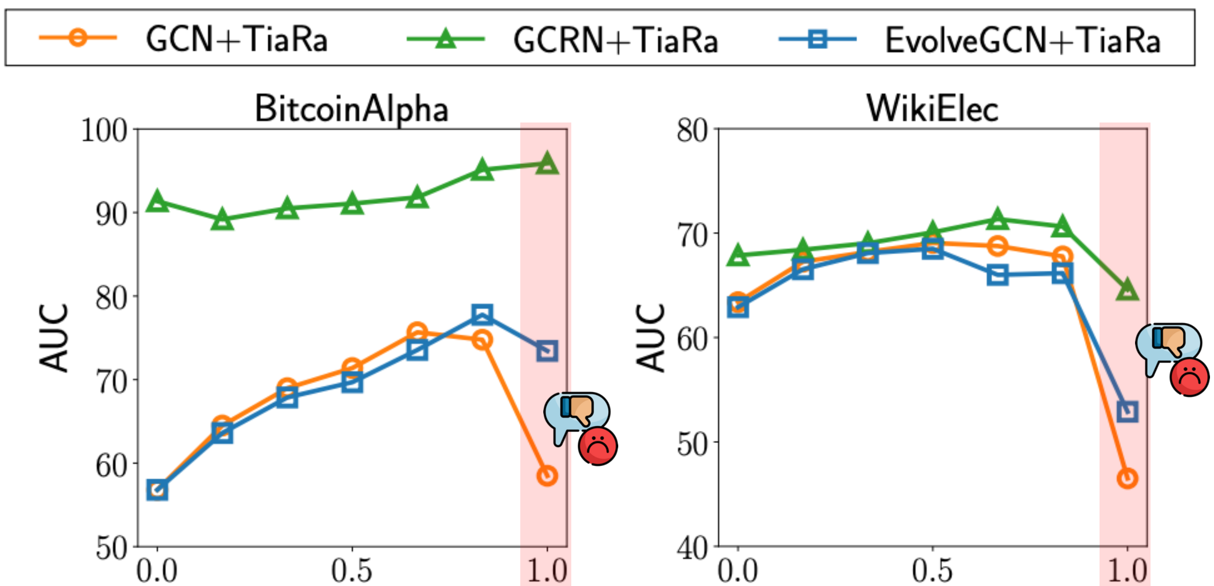
Macro F1	Brain			Reddit			DBLP-3			DBLP-5		
	GCN	GCRN	EGCN	GCN	GCRN	EGCN	GCN	GCRN	EGCN	GCN	GCRN	EGCN
None	44.7±0.8	66.8±1.0	43.4±0.7	18.2±2.9	40.4±1.6	18.6±2.3	53.4±2.6	83.1±0.6	51.3±2.7	69.6±0.9	75.4±0.7	68.5±0.6
DropEdge	▼35.2±1.7	▲67.8±0.6	▼39.7±1.8	▲**19.4±0.8**	▼40.3±1.4	▼18.0±2.7	▲55.8±1.9	▲84.3±0.6	▲52.4±1.7	▲70.5±0.5	▲75.6±0.7	▼68.0±0.7
GDC	▲63.2±1.2	▲88.0±1.5	▲67.3±1.3	▼17.5±2.3	▲41.0±1.6	▼18.5±2.8	▲53.4±2.1	▲84.7±0.5	▲52.8±2.2	▲70.0±0.7	▲75.5±1.2	▲69.1±1.0
Merge	▼34.4±3.4	▼63.2±1.6	▲53.0±0.9	▲19.3±3.0	▼39.6±0.8	▲20.4±3.0	▲54.9±3.1	▼83.0±1.4	▲53.3±1.2	▲70.8±0.4	▼74.5±0.8	▲69.7±1.6
TiaRa	▲**68.7±1.2**	▲**91.3±1.0**	▲**72.0±0.6**	▲18.4±3.0	▲**41.5±1.5**	▲**21.9±1.6**	▲**57.5±2.2**	▲**84.9±1.6**	▲**56.4±1.8**	▲**71.1±0.6**	▲**77.9±0.4**	▲**70.1±1.0**

TiaRa also works on the node classification task!

Effect of Hyperparameters (1)

❑ Effect of temporal decay ratio γ

- Mostly, AUC decreases drastically when $\gamma \rightarrow 1$
- Using the information of all time steps is a poor choice
- Important to properly mix spatial & temporal information

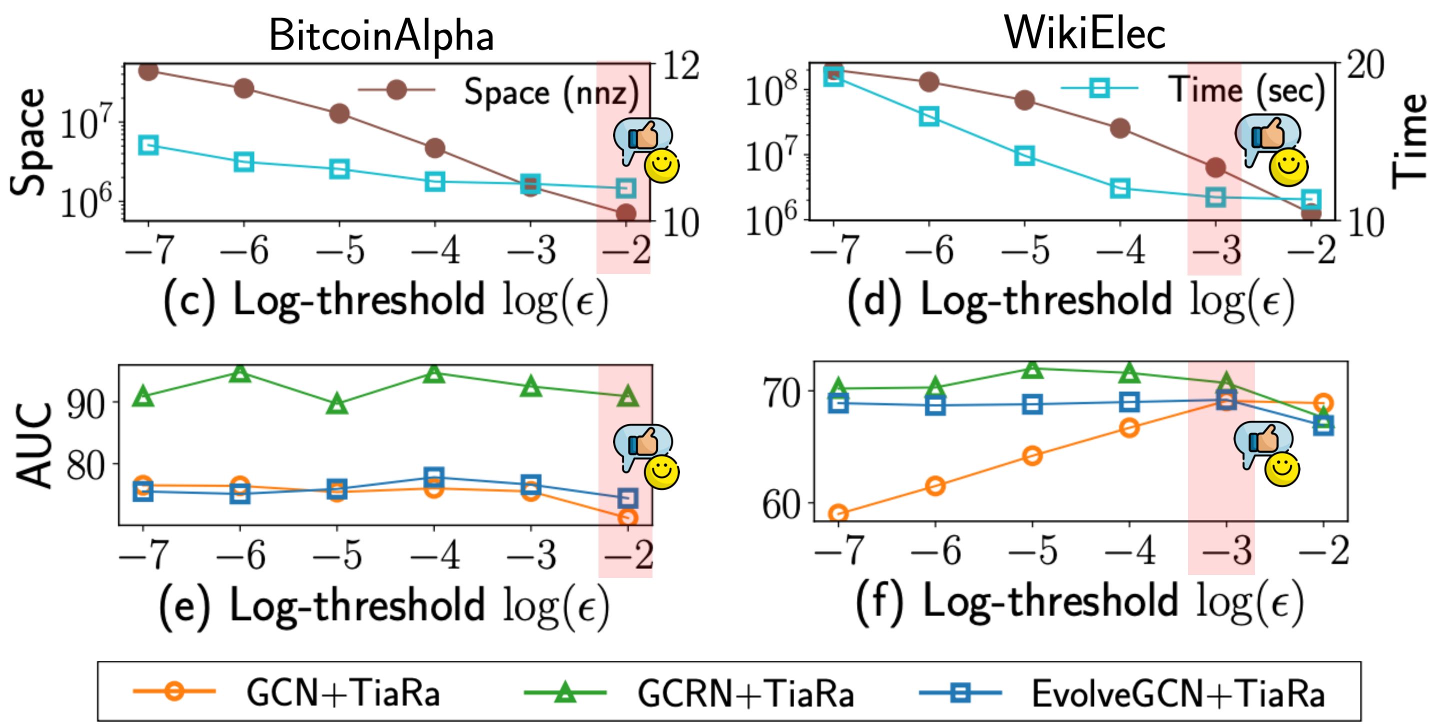


(a) Temporal decay ratio γ (b) Temporal decay ratio γ

Effect of Hyperparameters (2)

❑ Effect of filtering threshold ϵ

- Large ϵ improves efficiency while keeping accuracy



Sparsification makes TiaRa efficient and paractically usable!

Outline

- ❑ Introduction
- ❑ Motivation
- ❑ Proposed Method
- ❑ Experiments
- ❑ Conclusion

Conclusion

❑ TiaRa (Time-aware Random Walk Diffusion)

- 1) Make an RWR's surfer time-aware
- 2) Diffuse the time-aware surfer on the dynamic graph
- 3) Sparsify the diffused results for efficiency

❑ Aids dynamic GNNs in providing better accuracy

- Temporal locality as well as spatial locality are caputred
- Sparsification makes it efficient & paractically usable
- TiaRa improves the performance of dynamic GNNs on various tasks in dynamic graphs

Thank You

Jinhong Jung

Homepage: https://jinhongjung.github.io
Code: https://github.com/dev-jwel/TiaRa

SCAN ME
Homepage

Appendix

Computation of TiaRa

❑ Computing the augmented adjacency matrix $\mathcal{X}_t$

- Use Power iteration
  - Avoid matrix inversion
  - Repeatedly multiply the adjacency matrix
  - Guarantee convergence to the final answer

Algorithm 1: TIARA at time t

Require: adjacency matrix $\mathbf{A}_t$, previous time-aware diffusion matrix $\tilde{\boldsymbol{\mathcal{X}}}_{t-1}$, restart probability α, time travel probability β, number K of iterations, filtering threshold ϵ

Ensure: time-aware diffusion matrix $\tilde{\boldsymbol{\mathcal{X}}}_t^{\top}$

$\tilde{\boldsymbol{\mathcal{A}}}_t \leftarrow \mathbf{D}_t^{-1}\mathbf{A}_t$ where $\mathbf{D}_t = \text{diag}(\mathbf{A}_t\mathbf{1})$
$\boldsymbol{\mathcal{L}}_t^{\text{rwr}} \leftarrow$ POWER-ITERATION($\tilde{\boldsymbol{\mathcal{A}}}_t, \alpha, \beta, K$)
$\boldsymbol{\mathcal{S}}_t \leftarrow \boldsymbol{\mathcal{L}}_t^{\text{rwr}}$ ▷ **Spatial augmenter**
$\boldsymbol{\mathcal{T}}_t \leftarrow \boldsymbol{\mathcal{S}}_t\tilde{\boldsymbol{\mathcal{X}}}_{t-1}$ ▷ **Temporal augmenter**
$\boldsymbol{\mathcal{X}}_t \leftarrow (1-\gamma)\boldsymbol{\mathcal{S}}_t + \gamma\boldsymbol{\mathcal{T}}_t$ where $\gamma = \beta/(\alpha+\beta)$
$\tilde{\boldsymbol{\mathcal{X}}}_t \leftarrow$ filter entries of $\boldsymbol{\mathcal{X}}_t$ if their weights are $< \epsilon$
normalize $\tilde{\boldsymbol{\mathcal{X}}}_t$ column-wise
return $\tilde{\boldsymbol{\mathcal{X}}}_t^{\top}$
function POWER-ITERATION($\tilde{\boldsymbol{\mathcal{A}}}_t, \alpha, \beta, K$)
    set $c \leftarrow 1 - \alpha - \beta$ and $\mathbf{M}_t^{(0)} \leftarrow \boldsymbol{\mathcal{I}}_n$
    for $k \leftarrow 1$ to K **do**
        $\mathbf{M}_t^{(k)} \leftarrow \boldsymbol{\mathcal{I}}_n + c\tilde{\boldsymbol{\mathcal{A}}}_t^{\top}\mathbf{M}_t^{(k-1)}$
    $\boldsymbol{\mathcal{L}}_t^{\text{rwr}} \leftarrow (1-c)\mathbf{M}_t^{(K)}$ where $\mathbf{M}_t^{(K)} \approx \mathbf{L}_t^{-1}$
    normalize $\boldsymbol{\mathcal{L}}_t^{\text{rwr}}$ column-wise and **return** $\boldsymbol{\mathcal{L}}_t^{\text{rwr}}$
end function

Computational Complexity

[Appendix]

❑ Time complexity of TiaRa

- $O(n_t n/\epsilon + n_t^2 K)$ time at each time step
  - n_t: # of activated nodes (forming edges at time t)
  - n: # of total nodes
  - ϵ: filtering threshold (typically, 10^{-2} or 10^{-3})
  - K: # of power iterations
- Takes $O(n)$ time in real-world dynamic graphs
  - $n_t \ll n$, and ϵ^{-1} and K are constant
- Takes $O(n^2)$ time in dense graphs ($n_t = n$)

Datasets	n	$\lfloor \bar{n}_t \rfloor$
BitcoinAlpha	3,783	105
WikiElec	7,125	354
RedditBody	35,776	2,465
Brain	5,000	5,000
DBLP-3	4,257	782
DBLP-5	6,606	1,212
Reddit	8,291	2,071

❑ Space complexity of TiaRa

- Takes $O(n/\epsilon)$ space for augmentation at each time step

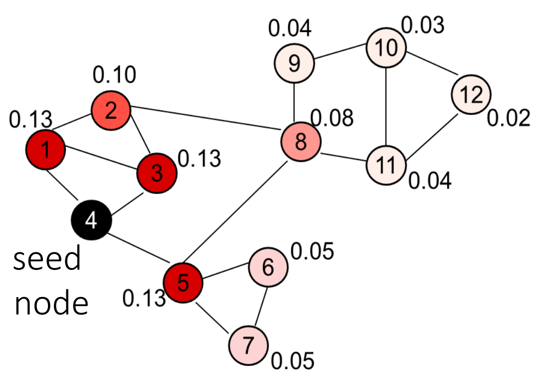
RWR Diffusion Matix $\mathcal{L}_t^{\mathrm{rwr}}$

❑ The term is derived from the equation of TRWR

$$\mathbf{x}_{t,s} = (1-\alpha-\beta)\boldsymbol{\mathcal{A}}_t^\top \mathbf{x}_{t,s} + \alpha\mathbf{i}_s + \beta\mathbf{x}_{t-1,s}$$

$$\Rightarrow (\boldsymbol{I}_n - (1-\alpha-\beta)\boldsymbol{\mathcal{A}}_t^\top)\mathbf{x}_{t,s} = \alpha\mathbf{i}_s + \beta\mathbf{x}_{t-1,s}$$

- Suppose $\mathbf{L}_t = \boldsymbol{I}_n - (1-\alpha-\beta)\boldsymbol{\mathcal{A}}_t^\top$
- Then, $\mathcal{L}_t^{\mathrm{rwr}} = (\alpha+\beta)\mathbf{L}_t^{-1}$
  - RWR scores of all pairs of nodes with restart probability $\alpha+\beta$



	Node 4
Node 1	0.13
Node 2	0.10
Node 3	0.13
Node 4	0.22
Node 5	0.13
Node 6	0.05
Node 7	0.05
Node 8	0.08
Node 9	0.04
Node 10	0.03
Node 11	0.04
Node 12	0.02

[Tong et al., ICDM06]

Input: an adjacency matrix **Output**: RWR scores w.r.t. seed