Online Appendix

A Details of Experiment Settings

Table 1: Summary of the real-world datasets

Name	# Nodes	# Edges	# time steps	# Classes	Summary
School [1]	319	2,327	10	9	Friendship
Aminer [3]	3,111	7,995	7	4	Citation
Patent [1]	12,214	41,916	10	6	Citation
DBLP [1]	28,085	150,568	10	10	Co-author
arXivAI [1]	69,854	696,819	10	5	Citation

A.1 Details of Datasets

We use five real-world time-evolving graph datasets of various types:

- *Aminer* and *arXivAI* are citation networks where nodes represent publications, and edges denote citation links between them.
- *Patent* is a patent citation network, where nodes represent patents, and edges indicate citation relationships among them.
- *School* is a friendship network where nodes represent students, and edges denote friendship interactions.
- *DBLP* is a co-authorship network, where nodes represent authors, and edges indicate co-authorship relationships between them.

Since these datasets do not provide node features, we generated 128-dimensional embeddings for node features by applying positional encoding based on the order in which the nodes appeared. To construct the time-evolving graphs, we processed the datasets to ensure that the number of incoming edges at each time step is equal. In this process, the Patent dataset was divided into a total of 7 time steps, while the remaining datasets were divided into 10 time steps each.

A.2 Details of TiGer

The GCN $f_L^{(t)}$ in M_L and $f_S^{(t)}$ in M_S are both 2-layer networks. The hidden dimension for each module is set to 64. For the proximity score additionally used in the ensemble, we employed SVD for the Patent dataset and the A.A. for the other datasets. The learning rate was searched within the range $\{1e-3, 5e-3, 1e-2, 5e-2\}$. The hyperparameters β and w_p were selected via grid search over $\{0.1, 0.2, 0.3\}$ and $\{1, 5, 10, 20\}$, respectively. The model was trained for 100 epochs. Positive and negative pseudo-labeled edges were randomly divided in a 4:1 ratio and used for training and validation, respectively.

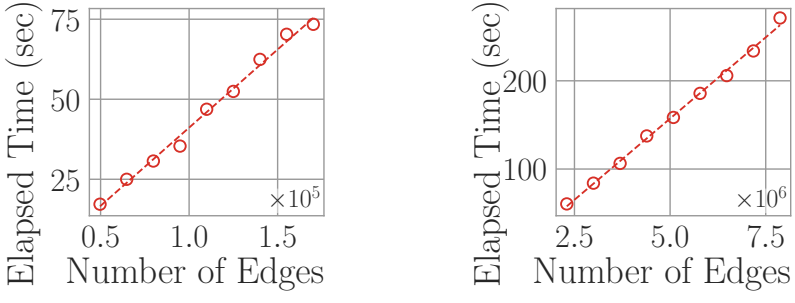
A.3 Details of Competitors

In this section, we provide descriptions of the competitors used in Sec. 4 and introduce their detailed parameter settings.

For a given pair of nodes, JACCARD is defined as the ratio of the number of common neighbors to the total number of neighbors. In the case of the A.A., it is computed as the sum of the inverse degrees of the common neighbors of the given two nodes. SVD uses a rank-100 approximation of the adjacency matrix, where the similarity score of an edge is determined by the corresponding element in the approximated adjacency matrix.

GDC employs graph diffusion based on personalized PageRank (PPR). The diffusion matrix (i.e., the output of GDC) is initially degree-normalized. To compute the similarity scores, this diffusion matrix is denormalized, and the element corresponding to each edge in the denormalized diffusion matrix is used as the similarity score for that edge. The teleport probability α, a hyperparameter for PPR, was selected via grid search over $\{0.05, 0.1, 0.15, 0.2\}$, as suggested in the original paper. TIARA extends GDC by incorporating temporal dependencies into the diffusion matrix. Specifically, it computes the diffusion matrix for a given time step as a weighted summation of the current diffusion matrix and those from previous time steps. Like GDC, TIARA also uses PPR-based graph diffusion, and the final similarity score for an edge is determined by the corresponding element in the denormalized diffusion matrix. An additional hyperparameter, β, which controls the weight of the diffusion matrix from previous time steps, was selected via grid search over $\{0.2, 0.4, 0.6, 0.8\}$.

LEO is an edge-scoring model that ensembles GNN-based modules and a node proximity-based module. We used the source code provided in the original paper and adopted the same GNN architecture as TIGER. All other parameters were tuned via grid search over the search space described in that paper.



(a) DBLP (b) arXivAI

Fig. 1: TIGER scaled **linearly** with the number of edges in the input graph.

B Scalability of TIGER

We evaluated the scalability of TIGER by measuring how its runtime changes depending on the size of the input graph. To this end, we measured the time

Table 2: (Q1) Accuracy under PGD attack. TiGer accurately identifies noise edges generated by a PGD attack. Each entry in the table represents the proportion of noise removed by each method at each time step. Note that all methods are tested under the same purification budget.

Method	$t=2$	$t=4$	$t=6$	$t=8$	$t=10$
SVD	28.76±3.45	40.36±1.13	42.34±0.62	41.17±1.98	39.66±2.10
A.A.	32.69±2.88	38.90±0.82	41.76±1.08	41.59±0.91	39.13±1.14
Jaccard	31.73±3.81	38.42±1.47	41.47±1.78	40.96±0.87	37.77±0.95
GDC	29.77±6.00	34.19±2.27	34.72±2.92	33.86±3.37	33.38±3.07
Tiara	8.28±1.78	26.26±1.69	24.11±2.86	25.04±1.70	27.76±1.40
LEO	33.18±4.89	34.50±4.66	35.40±4.75	36.85±4.28	33.81±3.66
TiGer	**34.14±2.44** (+**2.9%**)	**42.77±4.39** (+**6.0%**)	**43.60±4.08** (+**3.0%**)	**43.04±2.69** (+**3.5%**)	**40.04±2.34** (+**1.0%**)

(a) School

Method	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$
SVD	34.61±1.14	35.56±1.21	37.64±1.00	39.55±0.46	40.38±0.83
A.A.	31.96±0.08	33.89±0.38	34.74±0.67	37.12±0.50	37.59±0.57
Jaccard	31.96±0.08	33.89±0.38	34.86±0.74	37.30±0.69	37.97±0.88
GDC	8.42±0.91	12.00±2.01	12.32±2.61	12.76±1.86	13.91±1.51
Tiara	23.85±0.22	19.81±0.77	18.58±0.54	18.17±0.62	17.34±0.65
LEO	48.73±2.17	48.74±1.10	47.48±1.64	46.65±1.48	46.61±0.84
TiGer	**49.89±2.08** (+**2.4%**)	**51.13±1.37** (+**4.9%**)	**50.64±0.19** (+**6.7%**)	**51.80±0.41** (+**11.0%**)	**52.27±0.18** (+**12.1%**)

(b) Aminer

required to purify the unpurified graph at each time step for the DBLP and arXivAI datasets. As shown in Fig. 1, **TiGer scaled linearly with the size of the input graph**, as described in Sec. 3.6.

C Additional Experimental Results Supplementing Sec. 4.2

Table 2 presents the additional results regarding Q1 with the PGD attack [2]. Across all the experimental results, TiGer consistently outperforms all its competitors in all cases. These findings align with the results in Sec. 4.2 and further support the accuracy of TiGer.

References

1. Liu, M., Liu, Y., Liang, K., Tu, W., Wang, S., Zhou, S., Liu, X.: Deep temporal graph clustering. In: ICLR (2023)
2. Xu, K., Chen, H., Liu, S., Chen, P.Y., Weng, T.W., Hong, M., Lin, X.: Topology attack and defense for graph neural networks: An optimization perspective. In: IJCAI (2019)
3. Zhang, Z., Wang, X., Zhang, Z., Qin, Z., Wen, W., Xue, H., Li, H., Zhu, W.: Spectral invariant learning for dynamic graphs under distribution shifts. In: NeurIPS (2024)