Exploiting Temporal Importance for Adversarial Attacks on Dynamic Graphs

Brilian Surya Budi[1,†], Toshiyuki Amagasa[1,*,†]

[1]University of Tsukuba, Tsukuba, Ibaraki, Japan

Abstract

Graph neural networks (GNNs) have emerged as powerful tools for modeling dynamic graphs, yet their vulnerability to adversarial attacks remains underexplored. Existing attack methods on dynamic graphs lack explicit mechanisms to incorporate temporal importance into perturbation allocation. We propose TIDGA (Temporal Importance-aware Dynamic Graph Attack), a novel framework that incorporates temporal importance weighting to guide perturbation placement on discrete-time dynamic GNNs. TIDGA introduces two weighting mechanisms: (1) temporal position weight (α), which prioritizes earlier snapshots based on the intuition that early perturbations have more time to propagate through subsequent snapshots, and (2) edge persistence factor (β), which measures structural stability using Jaccard similarity. We evaluate TIDGA on three benchmark datasets, and the results demonstrate that TIDGA achieves statistically significant improvement over the comparative methods.

Keywords

Adversarial attack, discrete-time dynamic graph, graph neural network (GNN)

1. Introduction

Graph-structured data has become ubiquitous across numerous domains, and *Graph Neural Networks* (GNNs) have emerged as a powerful paradigm for learning from such data [1, 2, 3], with applications spanning social network analysis [4, 5], recommendation systems [6], and fraud detection [7, 8, 9, 10, 11]. While these foundational architectures operate on static graphs, real-world networks are inherently dynamic, with edges and nodes evolving over time [12, 13]. This has driven the formalization of discrete-time dynamic graphs, represented as ordered sequences of snapshots $\mathcal{G} = \{G_1, G_2, \ldots, G_T\}$ [14], and the development of architectures that combine graph convolution with recurrent mechanisms to capture both structural and temporal patterns.

Despite their effectiveness, GNNs are vulnerable to adversarial perturbations [15, 16]. While static GNN vulnerability has been extensively studied through surrogate-model-guided perturbations [17], gradient-based methods [18, 19], meta-learning [20], and continuous optimization [21], the vulnerability of discrete-time dynamic graph models remains comparatively underexplored. TGA [22] proposed greedy perturbation selection based on gradient magnitude across snapshots, and TD-PGD [23] extended projected gradient descent to the temporal setting. However, existing methods select edges based on gradient magnitude alone, ignoring temporal factors such as snapshot position and edge persistence. This work addresses this gap by incorporating temporal importance awareness into the perturbation optimization process.

Understanding this vulnerability is practically important: in safety-critical deployments such as fraud detection and social network monitoring, adversaries may manipulate historical interaction patterns to evade detection. Adversarial evaluation serves as a red-teaming tool for assessing model reliability before deployment, and studying effective perturbation strategies reveals which temporal patterns dynamic GNNs rely on, informing more robust architecture design.

Adversarial perturbations on dynamic graphs can be viewed as a form of graph transformation, where deliberate structural modifications to edge sets across temporal snapshots are designed to maximally

Published in the Proceedings of the Workshops of the EDBT/ICDT 2026 Joint Conference (March 24-27, 2026), Tampere, Finland
*Corresponding author.
briliansurya@kde.cs.tsukuba.ac.jp (B. S. Budi); amagasa@cs.tsukuba.ac.jp (T. Amagasa)
0000-0001-8093-1339 (B. S. Budi); 0000-0003-0595-2230 (T. Amagasa)

degrade model prediction performance. By studying how such transformations propagate through sequential architectures, we gain insights into the sensitivity of dynamic GNNs to structural changes at different temporal positions.

We propose **TIDGA** (Temporal Importance-aware Dynamic Graph Attack), a framework that weights perturbations based on their temporal significance. Our contributions are as follows:

- Temporal importance weighting mechanism that integrates two components: (1) temporal position weight, which prioritizes perturbations at earlier snapshots where they propagate through more subsequent computations, and (2) edge persistence factor, which considers edge stability across snapshots.
- Empirical analysis across three datasets and two victim model architectures, identifying that temporal weighting improves attack effectiveness on dense graphs with parameter-evolving architectures.

The rest of this paper is organized as follows. Section 2 introduces the preliminary definitions. Section 3 surveys the related studies. Section 4 presents the proposed method. We evaluate the performance in Section 5, followed by the results and discussions in Section 6. Section 7 concludes this paper.

2. Preliminaries

Definition 1 (Discrete-Time Dynamic Graph)**.** A discrete-time dynamic graph is an ordered sequence of graph snapshots $\mathcal{G} = \{G_1, G_2, \ldots, G_T\}$, where each snapshot $G_t = (V_t, A_t)$ consists of a node set V_t and adjacency matrix A_t.

Let $\{G'_1, G'_2, \ldots, G'_T\}$ denote the perturbed snapshots.

Definition 2 (Adversarial Attack on Dynamic Graphs)**.** Let $\mathcal{M}$ be a victim dynamic GNN and f_M its embedding function that generates node embeddings of G_t given $\mathcal{G}_{1:t-1}$. Let y_{task} be the ground-truth labels for a given task (for link prediction, these correspond to binary labels representing link existence at timestep T). The attacker aims to introduce structural perturbations $S_t = A'_t - A_t$ at each timestep $t < T$ such that model inference at timestep T for the target edges E_{tg} deteriorates. Following [23], the optimization problem is:

$$\begin{aligned} \max_{A'_1, A'_2, \ldots, A'_{T-1}} \quad & \mathcal{L}_{task}(\hat{y}_{task}(f_M(A'_{1:T-1})), y_{task}, E_{tg}) \\ \text{subject to} \quad & \sum_t \|A'_t - A_t\|_0 \leq k \end{aligned} \tag{1}$$

where $\hat{y}_{task}$ is the predicted labels, $\mathcal{L}_{task}$ is a task-specific loss (binary cross-entropy for link prediction), and k is the perturbation budget.

3. Related Work

Dynamic Graph Neural Networks. Several architectures combine graph convolution with temporal modeling for discrete-time dynamic graphs. EvolveGCN [24] employs recurrent units to evolve GCN weight matrices across timesteps. GC-LSTM [25] embeds graph convolution within LSTM gates, jointly considering structural context in gating decisions. DySAT [26] applies self-attention over both structural neighborhoods and historical snapshots, while TGAT [27] uses temporal encoding with attention mechanisms for continuous-time graphs. These architectures differ in how temporal information propagates: EvolveGCN through weight evolution, GC-LSTM through gated embeddings, and attention-based methods through learned attention scores. Architectures that evolve model parameters across timesteps create stronger temporal dependencies, a distinction with implications for adversarial vulnerability as analyzed in Section 6.

Adversarial Attacks on Graphs. For static GNNs, NETTACK [17] performs targeted attacks by selecting perturbations guided by a linearized surrogate model. FGA [18] uses first-order gradients of the attack loss to rank candidate edges, while IG-JSMA [19] applies integrated gradients for more accurate attribution. Metattack [20] treats global poisoning as a meta-learning problem, optimizing perturbations through the model's training procedure. RL-S2V [28] formulates attack as a Markov decision process solved via reinforcement learning. PGD topology attack [21] relaxes the discrete perturbation matrix to continuous space and applies projected gradient descent. For dynamic graphs, TGA [22] extends gradient-based selection by computing gradients across temporal snapshots for greedy edge selection, and TD-PGD [23] generalizes PGD topology attack to the temporal setting with constraints preserving temporal dynamics. Neither method considers temporal position when weighting perturbations, leaving potential improvement unexplored.

Practical Attack Scenarios. Adversarial attacks on dynamic GNNs are practically relevant in settings where adversaries can manipulate interactions over time. In fraud detection, adversaries may alter transaction patterns across time periods to evade GNN-based detectors [7]. In social networks, malicious actors may create or remove connections to manipulate recommendation systems. While white-box evaluation assumes strong attacker knowledge, such analysis establishes upper bounds on vulnerability that inform defense design and deployment decisions [15].

4. Methodology

4.1. Temporal Importance Weight

In discrete-time dynamic graphs, GNN models process graph snapshots sequentially, updating node representations at each timestep based on both current graph structure and previous hidden states. This sequential processing creates an asymmetry: perturbations introduced at earlier snapshots influence more subsequent computations than those at later snapshots. We design a weighting mechanism that exploits this temporal asymmetry.

Temporal Position Weight. We define $\alpha(t)$ to prioritize perturbations at earlier snapshots:

$$\alpha(t) = \left(\frac{T-t}{T}\right) \times \exp(-\lambda t) \tag{2}$$

where T is the sequence length and λ controls the decay rate. The linear term $(T-t)/T$ captures *propagation opportunity*: a perturbation at timestep t can influence $(T-t)$ subsequent snapshots before reaching the prediction target. The exponential term $\exp(-\lambda t)$ models *propagation attenuation*: as information flows through sequential layers, its influence diminishes due to normalization operations and non-linear activations. Together, these terms estimate the effective impact of perturbations based on their temporal position.

We note that this prioritization is motivated by the sequential processing of recurrent architectures: perturbations at earlier timesteps pass through more recurrent updates before reaching the prediction target at T, creating greater opportunity for cumulative influence on learned representations.

Edge Persistence Factor. Beyond temporal position, we consider structural stability. Edges that persist across multiple snapshots are more deeply encoded in learned representations, as the model repeatedly observes and reinforces these patterns. We quantify persistence using Jaccard similarity between consecutive edge sets:

$$\beta(t) = \frac{1}{K}\sum_{k=1}^{K} J(E_t, E_{t+k}), \quad J(E_a, E_b) = \frac{|E_a \cap E_b|}{|E_a \cup E_b|} \tag{3}$$

where K is the lookahead window. Higher $\beta(t)$ indicates that edges at snapshot t tend to persist, suggesting perturbations at this snapshot target more stable, and thus more impactful, structural patterns.

Combined Weight. The temporal importance weight integrates both factors:

$$W(t) = \alpha(t) \times \beta(t) \tag{4}$$

Normalization. We apply $W(t)$ as a learning rate multiplier during gradient-based perturbation optimization. To preserve overall optimization dynamics, we normalize by the mean:

$$\tilde{W}(t) = \frac{W(t)}{\bar{W}}, \quad \text{where } \bar{W} = \frac{1}{T}\sum_{\tau=1}^{T} W(\tau) \tag{5}$$

This ensures the average multiplier equals 1: snapshots with above-average importance receive amplified updates ($\tilde{W}(t) > 1$), while below-average snapshots receive dampened updates ($\tilde{W}(t) < 1$). The total learning capacity is preserved while redistributing emphasis toward temporally significant snapshots.

4.2. TIDGA Attack Algorithm

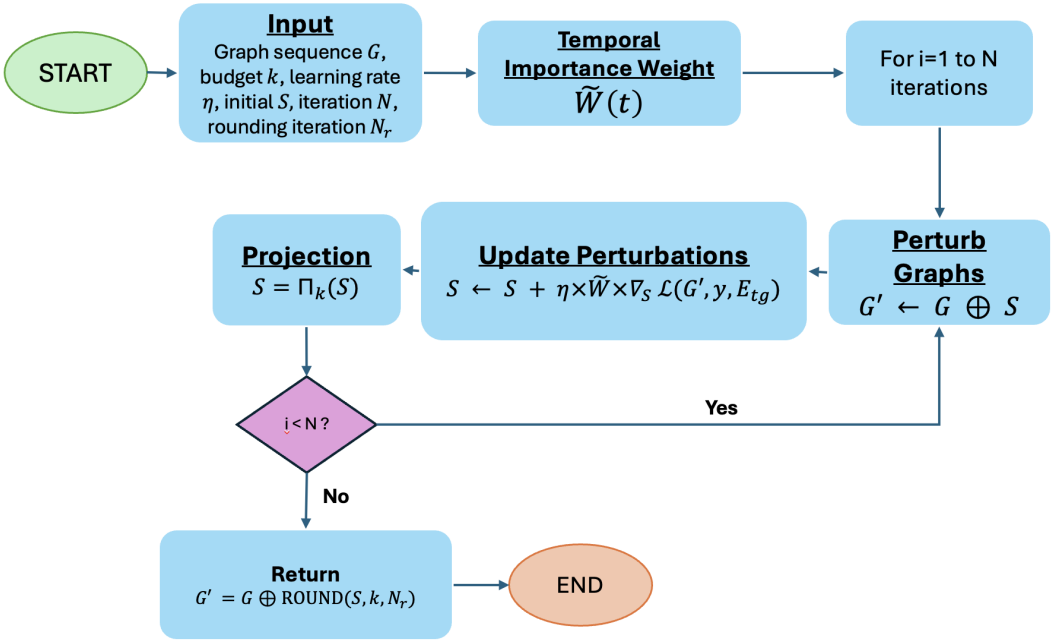

Figure 1: Overview of the TIDGA attack framework

TIDGA optimizes the perturbation matrix S initialized to ones. At each iteration, gradients are weighted by temporal importance:

$$S_t \leftarrow S_t + \eta \times \tilde{W}(t) \times \nabla_{S_t}\mathcal{L} \tag{6}$$

where η is the learning rate. The normalized weight $\tilde{W}(t)$ functions as a per-snapshot learning rate multiplier, making the effective learning rate $\eta \times \tilde{W}(t)$ for each snapshot. For earlier snapshots where $\tilde{W}(t) > 1$, gradient updates are amplified, causing perturbation scores to increase more rapidly.

After each update, the perturbation matrix is projected onto the feasible set. When $\sum P_{[0,1]}(S) > k$, a threshold μ is found via bisection search such that $\sum P_{[0,1]}(S - \mu) = k$, where $P_{[0,1]}(x) = \max(0, \min(1, x))$. After optimization converges, randomized rounding with multiple samples converts continuous values to discrete perturbations, selecting the sample maximizing attack loss.

Algorithm 1 TIDGA

Require: Input: Graph sequence G, budget k, learning rate η, Initial S_0, iteration N, rounding iteration N_r

Ensure: Output: Perturbed graph G'

$\tilde{W} \leftarrow \text{TemporalImportanceWeight}(G)$
for $i = 1$ to N **do**
  $G' \leftarrow G \oplus S$
  $S \leftarrow S + \eta \times \tilde{W} \times \nabla_S \mathcal{L}(G', y, E_{tg})$
  $S \leftarrow \Pi(S)$
end for
return $G' = G \oplus \text{Discretization}(S, k, N_r)$

5. Experimental Evaluation

Datasets. We evaluate on three datasets with varying characteristics, following the preprocessing protocol of Sharma et al. [23]. Table 1 summarizes the dataset statistics. (1) **Radoslaw**—a dense email communication network from a manufacturing company with 167 nodes, 22K edges across 13 snapshots (3-week split), exhibiting high edge density and strong temporal persistence; (2) **UCI**—a medium-scale online message network with 1.9K nodes, 24K edges across 13 snapshots (2-week split) (3) **Reddit**—a large sparse subreddit hyperlink network with 35K nodes, 715K edges across 20 snapshots (4-week split). These datasets span diverse graph properties to comprehensively evaluate temporal weighting effectiveness.

Table 1
Dataset statistics. $|V|$ denotes number of nodes, $|E|$ denotes total edges.

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	Snapshots	Split
Radoslaw	167	22K	13	3 weeks
UCI	1.9K	24K	13	2 weeks
Reddit	35K	715K	20	4 weeks

Victim Models. Two discrete-time dynamic GNN architectures serve as targets: **EvolveGCN-O** [24], which uses an LSTM to evolve GCN weight matrices over time ($W_t = \text{LSTM}(W_{t-1})$) independent of node embeddings, and **GC-LSTM** [25], which embeds graph convolution within LSTM gates to process hidden and cell states, allowing graph structure to directly influence recurrent state updates.

Training Protocol. Both victim models are trained on clean graph snapshots from timesteps 1 to $T-1$ for link prediction at timestep T, following the configuration of Sharma et al. [23]: embedding dimension 32, Adam optimizer with learning rate 0.0001, and 500 training epochs with early stopping patience of 10. The attack operates in a poisoning setting where perturbations are injected into training snapshots before model training. For target edge selection, UCI and Radoslaw use all edges as attack targets, while Reddit randomly samples 500 positive and 500 negative edges due to computational constraints.

Task and Evaluation Protocol. We target link prediction at the final timestep T, where the model predicts edge existence based on learned node embeddings. Attack effectiveness is measured by relative drop:

$$\text{Rel. Drop} = \frac{\text{AUC}_{\text{attacked}} - \text{AUC}_{\text{clean}}}{\text{AUC}_{\text{clean}}} \times 100\% \tag{7}$$

where larger negative values indicate more effective attacks.

Baselines. TIDGA is compared against four methods: (1) **Random**—uniform random perturbation selection serving as lower bound; (2) **Degree**—targeting edges connected to high-degree nodes based on structural importance heuristic; (3) **TGA-Greedy** [22]—greedy selection by gradient magnitude

without temporal consideration; (4) **TD-PGD** [23]—projected gradient descent with temporal dynamics constraints, without temporal importance consideration.

Configuration. Attack budgets $k \in \{5, 10, 20, 50\}$ are tested with 50 optimization iterations using Adam optimizer [29] with initial learning rate $\eta = 10$, temporal decay $\lambda = 0.1$, and lookahead window $K = 3$. Results are averaged over 4 random seeds with standard deviation reported. Statistical significance is assessed with $\alpha = 0.05$ using paired t-tests when normality assumptions hold, and Wilcoxon signed-rank tests otherwise.

Computational Overhead. TIDGA adds negligible per-iteration overhead compared to TD-PGD, as temporal weighting involves only element-wise multiplication with precomputed weights; the one-time preprocessing for Jaccard similarities is $O(T \times K \times |E|)$.

6. Results and Discussion

6.1. Main Results

Radoslaw-EvolveGCN-O. TIDGA achieves the best performance among all attack methods, including Random, Degree heuristic, TGA, and TD-PGD. PGD-based methods (TIDGA and TD-PGD) substantially outperform other approaches—at budget=50, TIDGA achieves -60.10% relative drop compared to TGA's -19.29% and Random's -9.62%. Among PGD-based methods, TIDGA outperforms TD-PGD with margins of 1.17% (budget=5) to 6.10% (budget=50), with statistical significance confirmed (Wilcoxon signed-rank, p-value $= 3.05$e-5). The improvement magnitude increases with budget size, suggesting temporal weighting becomes more effective when more perturbations are available for strategic allocation. This configuration combines favorable conditions: dense graph structure ensuring substantial gradient flow, short sequence (13 snapshots) where gradient vanishing remains manageable, and EvolveGCN-O's parameter-evolving architecture.

Radoslaw-GCLSTM. TIDGA's effectiveness diminishes on GC-LSTM architecture. While PGD-based methods still outperform other approaches (TIDGA: -15.79% vs TGA: -2.11% at budget=50), TIDGA shows marginally lower effectiveness than TD-PGD with differences of 0.34–1.38%. GC-LSTM integrates graph convolution within LSTM gates operating on embeddings rather than evolving weights. Consequently, perturbations at early snapshots do not accumulate influence through the same propagation mechanism, reducing temporal weighting benefit.

UCI Dataset. On EvolveGCN-O, TIDGA and TD-PGD achieve comparable performance, both substantially outperforming other baselines (PGD-based: $> 60\%$ drop vs TGA: $< 53\%$). Non-monotonic behavior occurs at budget=20, where both methods underperform compared to adjacent budgets, indicating convergence to local optima due to UCI's intermediate density creating a complex loss landscape. On GC-LSTM, TIDGA shows reduced effectiveness compared to TD-PGD, consistent with the pattern observed on Radoslaw.

Reddit Dataset. On EvolveGCN-O, TIDGA underperforms TD-PGD by 0.61–1.78%, a difference confirmed as statistically significant (Wilcoxon signed-rank, $p < 0.001$). On GC-LSTM, TIDGA shows substantially reduced effectiveness across all budgets, with the gap widening at higher budgets (-3.23% vs -6.48% at budget=50). This represents the least favorable configuration for TIDGA.

6.2. Analysis of Effectiveness Conditions

Figure 3 reveals the mechanism underlying TIDGA's performance variation. On Radoslaw, actual perturbation distribution (Early: 49.2%, Late: 22.5%) reasonably aligns with intended allocation (Early: 65.4%, Late: 7.3%). On Reddit, while temporal weighting intends 69.4% at early timesteps and 6.1% at late timesteps, actual distribution shows the opposite: 17.3% early and 70.2% late, with 51% concentrated at $t = 18$.

This discrepancy arises because $\tilde{W}(t)$ functions as a learning rate multiplier rather than strict enforcement. On Reddit, gradient-based optimization identifies late timesteps as more effective perturbation targets. TIDGA's temporal weighting works against this by dampening updates at late timesteps while

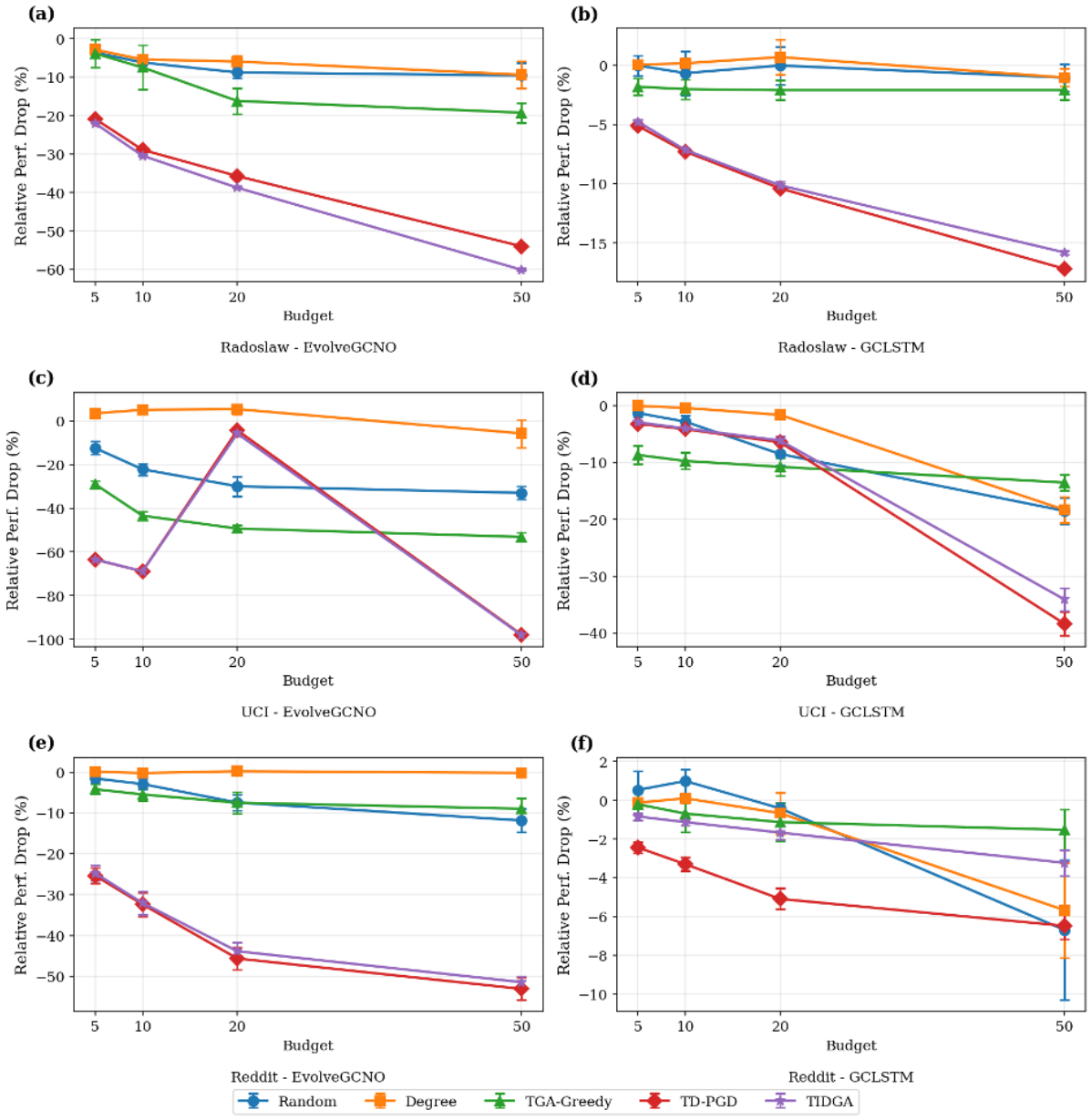

Figure 2: Attack performance across datasets and models

boosting early timesteps where gradients indicate lower attack effectiveness. Since actual perturbation placement is determined by gradient magnitudes after weighting, TIDGA's intervention becomes counterproductive when the optimization landscape favors late timesteps. On Radoslaw, gradients favor early timesteps, so temporal weighting successfully reinforces this direction.

Architectural Vulnerability Analysis. The consistent performance gap between EvolveGCN-O and GC-LSTM reveals fundamental architectural differences. EvolveGCN-O evolves weight matrices through LSTM recurrence ($W_t^{(l)} = \text{LSTM}(H_{t-1}, W_{t-1}^{(l)})$), creating a cascading effect where early perturbations accumulate influence through the weight evolution chain. In contrast, GC-LSTM operates recurrence on node embeddings ($h_t = \text{LSTM}(\text{GCN}(X_t, A_t), h_{t-1})$) with fixed parameters across timesteps, limiting temporal accumulation of perturbation effects and explaining why temporal weighting provides minimal benefit regardless of dataset.

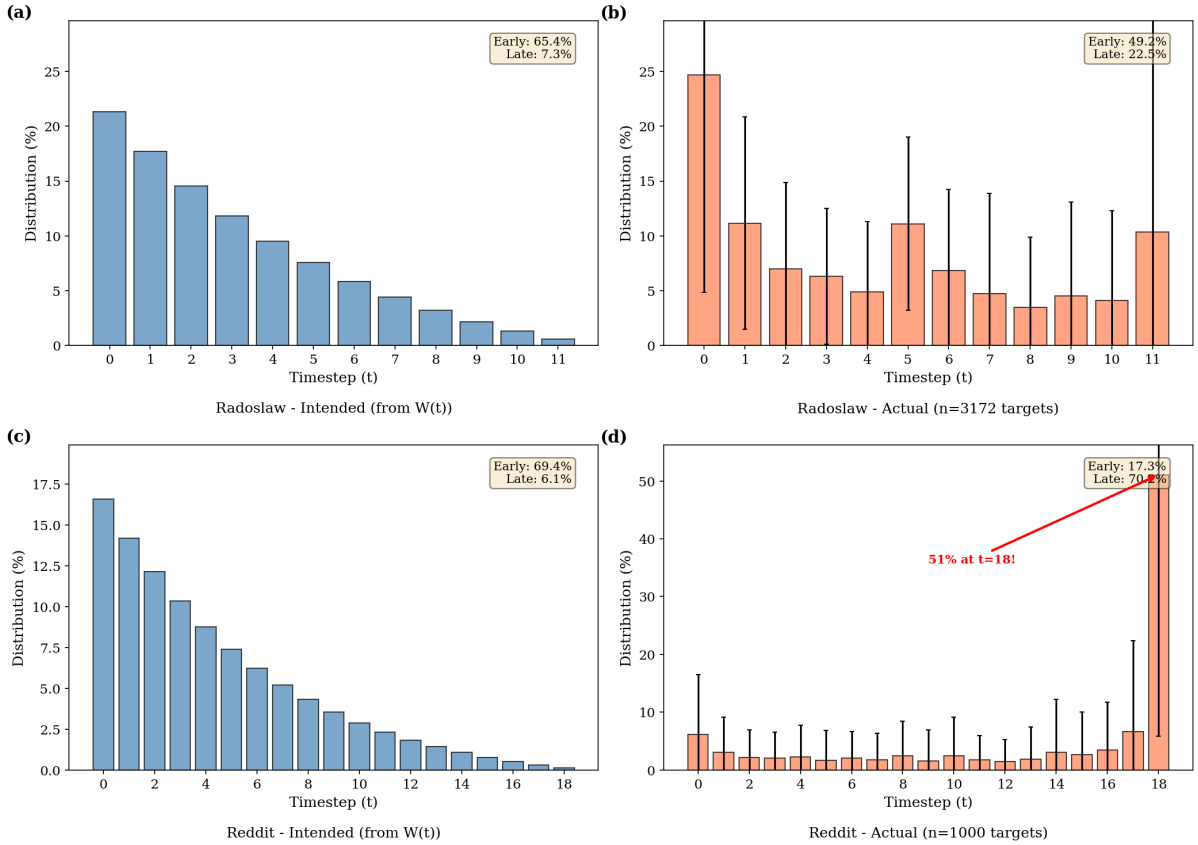

Figure 3: Perturbation Distribution Analysis: Intended Allocation from $W(t)$ versus Actual Placement from Optimization

6.3. Ablation Study

We conduct ablation on Radoslaw with EvolveGCN-O, comparing TIDGA-α (temporal position only) and TIDGA-β (edge persistence only). At budget=50, TIDGA-α achieves -60.56% versus -54.45% for TIDGA-β, confirming that temporal position weight contributes more substantially. Figure 4 summarizes the results.

6.4. Summary of Findings

TIDGA improves attack effectiveness when two conditions are satisfied: (1) the victim model employs recurrent architecture that evolves model parameters across snapshots (EvolveGCN-O), and (2) gradient magnitudes remain distributed across multiple snapshots rather than concentrated at specific timesteps. When both conditions are met, TIDGA achieves statistically significant improvement over TD-PGD. When either condition is violated, TD-PGD maintains equal or superior performance. Across all configurations, both PGD-based methods substantially outperform heuristic baselines by 20–50 percentage points.

7. Conclusion

This work proposed TIDGA, a temporal importance-aware adversarial attack framework for discrete-time dynamic graph neural networks. By incorporating temporal position weight $\alpha(t)$ and edge persistence factor $\beta(t)$ as learning rate multipliers during gradient-based optimization, TIDGA guides perturbation allocation toward temporally strategic positions.

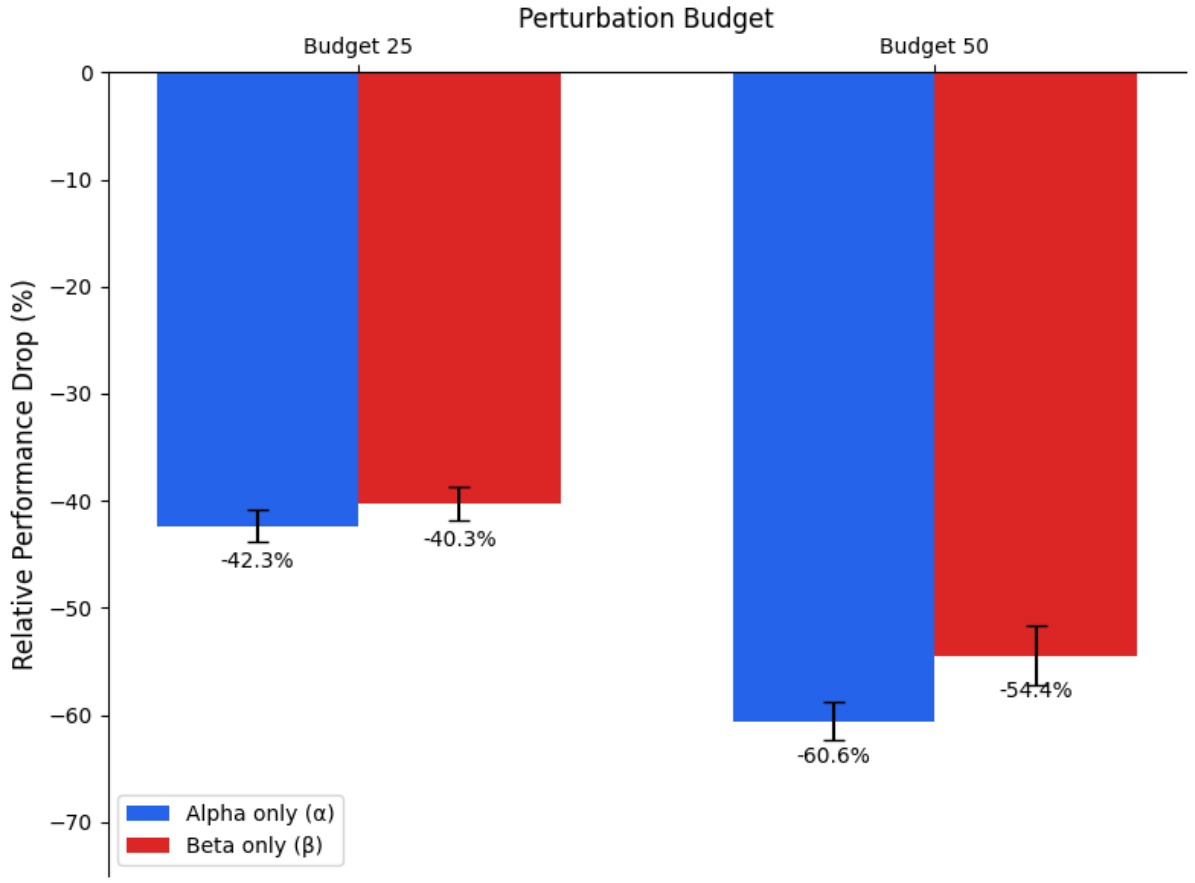

Figure 4: Ablation study comparing the contribution of temporal position weight (α) and edge persistence factor (β) on Radoslaw dataset with EvolveGCN-O victim model

Experimental results demonstrate that TIDGA achieves the best performance among all attack methods on EvolveGCN-O with dense graphs (Radoslaw), outperforming TD-PGD by margins reaching 6.10% at higher budgets. TIDGA is effective when gradient magnitudes are balanced across snapshots, allowing temporal weighting to meaningfully redistribute optimization emphasis. However, TD-PGD outperforms TIDGA on GC-LSTM, where the architecture limits temporal propagation and gradient magnitudes concentrate at later snapshots, diminishing the effect of weighting at earlier timesteps. Ablation studies reveal that temporal position weight $\alpha(t)$ contributes more significantly than edge persistence factor $\beta(t)$, indicating that perturbation timing is more critical than structural stability for attack effectiveness. The key insight is that $\tilde{W}(t)$ operates as a learning rate multiplier rather than budget enforcement—effective only when gradients at early timesteps remain substantial enough to amplify.

These findings reveal that dynamic GNNs with recurrent weight propagation are particularly vulnerable to temporally-aware attacks under favorable conditions. Our evaluation is limited to small-to-medium graphs and link prediction; future work includes scaling to larger graphs, extending to tasks such as node classification, and exploring adaptive weighting strategies that learn temporal importance from gradient distributions.

Acknowledgments

This paper is based on results obtained from the project, "Research and Development Project of the Enhanced infrastructures for Post-5G Information and Communication Systems" (JPNP20017), commis-

sioned by the New Energy and Industrial Technology Development Organization (NEDO), JST CREST Grant Number JPMJCR22M2, and JSPS KAKENHI Grant Number JP23K24949.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2017. URL: https://arxiv.org/abs/1609.02907. arXiv:1609.02907.

[2] W. L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, 2018. URL: https://arxiv.org/abs/1706.02216. arXiv:1706.02216.

[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, 2018. URL: https://arxiv.org/abs/1710.10903. arXiv:1710.10903.

[4] A. Sankar, Y. Liu, J. Yu, N. Shah, Graph neural networks for friend ranking in large-scale social platforms, in: Proceedings of the Web Conference 2021, WWW '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 2535–2546. doi:10.1145/3442381.3450120.

[5] X. Tang, Y. Liu, N. Shah, X. Shi, P. Mitra, S. Wang, Knowing your fate: Friendship, action and temporal explanations for user engagement prediction on social apps, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 2269–2279. doi:10.1145/3394486.3403276.

[6] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 974–983. doi:10.1145/3219819.3219890.

[7] S. Xiang, M. Zhu, D. Cheng, E. Li, R. Zhao, Y. Ouyang, L. Chen, Y. Zheng, Semi-supervised credit card fraud detection via attribute-driven graph representation, Proceedings of the AAAI Conference on Artificial Intelligence 37 (2023) 14557–14565. URL: https://ojs.aaai.org/index.php/AAAI/article/view/26702. doi:10.1609/aaai.v37i12.26702.

[8] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, P. S. Yu, Enhancing graph neural network-based fraud detectors against camouflaged fraudsters, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, CIKM '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 315–324. URL: https://doi.org/10.1145/3340531.3411903. doi:10.1145/3340531.3411903.

[9] Z. Liu, Y. Dou, P. S. Yu, Y. Deng, H. Peng, Alleviating the inconsistency problem of applying graph neural network to fraud detection, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20, ACM, 2020, p. 1569–1572. URL: http://dx.doi.org/10.1145/3397271.3401253. doi:10.1145/3397271.3401253.

[10] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, Q. He, Pick and choose: A gnn-based imbalanced learning approach for fraud detection, in: Proceedings of the Web Conference 2021, WWW '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 3168–3177. URL: https://doi.org/10.1145/3442381.3449989. doi:10.1145/3442381.3449989.

[11] T. Zhao, B. Ni, W. Yu, Z. Guo, N. Shah, M. Jiang, Action sequence augmentation for early graph-based anomaly detection, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 2668–2678. URL: https://doi.org/10.1145/3459637.3482313. doi:10.1145/3459637.3482313.

[12] G. Kossinets, D. J. Watts, Empirical analysis of an evolving social network, Science 311 (2006) 88–90. doi:10.1126/science.1116869.

[13] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph evolution: Densification and shrinking diameters, ACM Trans. Knowl. Discov. Data 1 (2007) 2–es. doi:10.1145/1217299.1217301.

[14] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, P. Poupart, Representation learning for dynamic graphs: A survey, Journal of Machine Learning Research 21 (2020) 1–73. URL: http://jmlr.org/papers/v21/19-447.html.

[15] S. Günnemann, Graph Neural Networks: Adversarial Robustness, Springer Nature Singapore, Singapore, 2022, pp. 149–176. URL: https://doi.org/10.1007/978-981-16-6054-2_8. doi:10.1007/978-981-16-6054-2_8.

[16] W. Jin, Y. Li, H. Xu, Y. Wang, S. Ji, C. Aggarwal, J. Tang, Adversarial attacks and defenses on graphs, SIGKDD Explor. Newsl. 22 (2021) 19–34. doi:10.1145/3447556.3447566.

[17] D. Zügner, A. Akbarnejad, S. Günnemann, Adversarial attacks on neural networks for graph data, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18, ACM, 2018, p. 2847–2856. doi:10.1145/3219819.3220078.

[18] J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, Q. Xuan, Fast gradient attack on network embedding, 2018. URL: https://arxiv.org/abs/1809.02797. arXiv:1809.02797.

[19] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, L. Zhu, Adversarial examples for graph data: Deep insights into attack and defense, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 4816–4823. doi:10.24963/ijcai.2019/669.

[20] D. Zügner, S. Günnemann, Adversarial attacks on graph neural networks via meta learning, 2019. URL: https://arxiv.org/abs/1902.08412. arXiv:1902.08412.

[21] K. Xu, H. Chen, S. Liu, P.-Y. Chen, T.-W. Weng, M. Hong, X. Lin, Topology attack and defense for graph neural networks: An optimization perspective, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 3961–3967. doi:10.24963/ijcai.2019/550.

[22] J. Chen, J. Zhang, Z. Chen, M. Du, Q. Xuan, Time-aware gradient attack on dynamic network link prediction, IEEE Transactions on Knowledge and Data Engineering 35 (2023) 2091–2102. doi:10.1109/TKDE.2021.3110580.

[23] K. Sharma, R. Trivedi, R. Sridhar, S. Kumar, Temporal dynamics-aware adversarial attacks on discrete-time dynamic graph models, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 2023–2035. doi:10.1145/3580305.3599517.

[24] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, C. Leiserson, Evolvegcn: Evolving graph convolutional networks for dynamic graphs, Proceedings of the AAAI Conference on Artificial Intelligence 34 (2020) 5363–5370. doi:10.1609/aaai.v34i04.5984.

[25] J. Chen, X. Wang, X. Xu, Gc-lstm: graph convolution embedded lstm for dynamic network link prediction, Applied Intelligence 52 (2021) 7513–7528. doi:10.1007/s10489-021-02518-9.

[26] A. Sankar, Y. Wu, L. Gou, W. Zhang, H. Yang, Dysat: Deep neural representation learning on dynamic graphs via self-attention networks, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 519–527.

[27] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, K. Achan, Inductive representation learning on temporal graphs, 2020. URL: https://arxiv.org/abs/2002.07962. arXiv:2002.07962.

[28] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, L. Song, Adversarial attack on graph structured data, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, volume 80 of *Proceedings of Machine Learning Research*, PMLR, 2018, pp. 1115–1124. URL: https://proceedings.mlr.press/v80/dai18b.html.

[29] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017. URL: https://arxiv.org/abs/1412.6980. arXiv:1412.6980.