LoReTTA: A Low Resource Framework to Poison Continuous Time Dynamic Graphs

Himanshu Pal*, **Venkata Sai Pranav Bachina***, **Ankit Gangwal**, **Charu Sharma**

International Institute of Information Technology, Hyderabad

Abstract

Temporal Graph Neural Networks (TGNNs) are increasingly used in high-stakes domains, such as financial forecasting, recommendation systems, and fraud detection. However, their susceptibility to poisoning attacks poses a critical security risk. We introduce **LoReTTA** (**Lo**w **Re**source **T**wo-phase **T**emporal **A**ttack), a novel adversarial framework on Continuous-Time Dynamic Graphs, which degrades TGNN performance by an average of **29.47%** across 4 widely benchmark datasets and 4 State-of-the-Art (SotA) models.

LoReTTA operates through a two-stage approach: (1) sparsify the graph by removing high-impact edges using any of the 16 tested temporal importance metrics, (2) strategically replace removed edges with adversarial negatives via LoReTTA's novel degree-preserving negative sampling algorithm. Our plug-and-play design eliminates the need for expensive surrogate models while adhering to realistic unnoticeability constraints. LoReTTA degrades performance by upto **42.0%** on MOOC, **31.5%** on Wikipedia, **28.8%** on UCI, and **15.6%** on Enron. LoReTTA outperforms 11 attack baselines, remains undetectable to 4 leading anomaly detection systems, and is robust to 4 SotA adversarial defense training methods, establishing its effectiveness, unnoticeability, and robustness.

Code — https://github.com/ansh997/LoReTTA

Extended version — https://arxiv.org/abs/2511.07379

Introduction

Temporal Graph Neural Networks (TGNNs) are increasingly deployed in real-world systems such as social media analysis (Deng, Rangwala, and Ning 2019; Fan et al. 2019), transportation modeling (Jiang and Luo 2022; Yu, Yin, and Zhu 2017), recommendation engines (Gao et al. 2022; Wu et al. 2022), outbreak tracking (Cencetti et al. 2021; So et al. 2020), and knowledge graph reasoning (Cai et al. 2022). These applications often rely on modeling evolving relationships over time using dynamic graphs, making the robustness of TGNNs a crucial concern.

Dynamic graphs can be broadly categorized as Discrete-Time Dynamic Graphs (DTDGs) and CTDGs. DTDGs represent dynamic graphs as a sequence of static snapshots, while CTDGs model them as a continuous stream of timestamped interactions. CTDGs are considered more expressive for modeling real-world systems, as they capture fine-grained temporal dependencies that DTDGs may miss (Zheng, Wei, and Liu 2023; Ennadir et al. 2024).

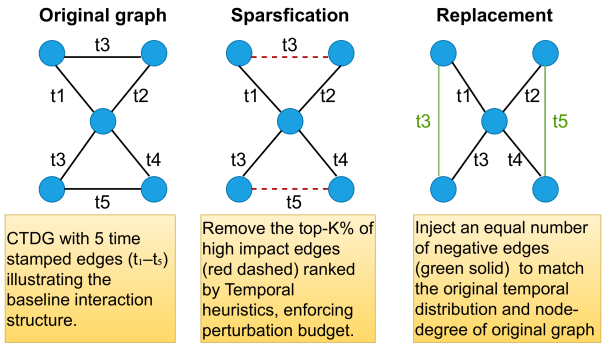


Figure 1: Overview of the LoReTTA attack framework.

Recent work (Lee, Lee, and Shin 2024) has demonstrated that TGNNs are susceptible to adversarial poisoning attacks, i.e small but carefully crafted perturbations to the training graph that can significantly degrade performance. This threat becomes especially critical in high-stakes domains such as fraud detection or traffic forecasting, where reliability is paramount.

While adversarial attacks have been extensively studied for static graphs (Zügner, Akbarnejad, and Günnemann 2018; Bojchevski and Günnemann 2019; Dai et al. 2018; Ma, Ding, and Mei 2020a; Zügner and Günnemann 2024), their temporal counterparts (CTDGs) are less explored. The challenges are non-trivial: CTDGs evolve over time, requiring the attacker to perturb edges with precise temporal coordination. TGNNs further complicate this by maintaining memory states that propagate historical information and may dilute isolated attacks. Moreover, the attacker must decide *when* and *where* to insert perturbations to maximize the impact, all the while adhering to unnoticeability constraints (cf. Appendix A).

The SotA work, i.e., Spear and Shield (Lee, Lee, and Shin 2024), introduces an adversarial poisoning framework for TGNNs, but it carries significant practical limitations. Specif-

*These authors contributed equally.

ically, it relies on training a surrogate model to estimate gradients, an approach that is **computationally expensive for large-scale datasets**. Furthermore, it assumes the attacker has complete access to the entire dataset (train, validation, and test) and the ability to make arbitrary changes. In real-world scenarios, such assumptions rarely hold in the context of an attacker. Attackers often only have access to the training portion of the graph and must operate under realistic, limited knowledge and manipulation constraints.

To address these limitations, we propose a novel adversarial poisoning framework for TGNNs, that we call **LoRETTA** (**Lo**w **Re**source **T**wo-phase **T**emporal **A**ttack). LoRETTA does not rely on surrogate models, and it assumes access to only the training portion of the data. It adheres to 4 practical unnoticeability constraints (cf. §) that preserve the structural and temporal plausibility of the poisoned graph. At its core, LoRETTA computes scores for edges (using any of our 16 tested heuristics) across timestamps to quantify evolving influence, and removes edges with high temporal influence; thereby maximizing disruption to the TGNN's representation learning. These edges are then replaced with strategically selected adversarial negatives through LoRETTA's novel negative sampling algorithm designed to preserve the original graph's edge density and local structural cues.
Our main contributions are summarized below:

1. We propose LoRETTA, a novel adversarial poisoning framework that operates by removing high-influence edges and replacing them with adversarial negatives using LoRETTA's novel negative sampling algorithm.
2. We enforce 4 unnoticeability constraints (cf.§) that includes graph sparsity, temporal plausibility, node activity windows, and degree preservation to ensure attack stealth.
3. We demonstrate LoRETTA (a) outperforms 11 baselines (cf. Appendix A) with 29.47% average degradation across 4 datasets and 4 SotA TGNN models; (b) while evading 4 SotA anomaly detectors (cf. Appendix A) and (c) resisting 4 adversarial defense methods (cf. Appendix A).

Related Work

Temporal Graph Neural Networks (TGNNs)

Dynamic graphs are commonly categorized into two settings: DTDGs and CTDGs. DTDG approaches discretize a temporal graph into sequential snapshots and apply static graph learning to each frame (Pareja et al. 2020; Sankar et al. 2020). However, these methods neglect fine-grained temporal continuity, leading to less faithful modeling of real-world dynamic systems. In contrast, CTDG-based approaches, particularly TGNNs (Trivedi et al. 2019; Rossi et al. 2020; Xu et al. 2020; Ma et al. 2020), model interactions as continuous temporal streams and capture temporal dependencies directly. These models often incorporate memory networks (Ma et al. 2020; Rossi et al. 2020), sequential encoders (Cong et al. 2023), and temporal message-passing schemes, allowing more expressive modeling of evolving systems. TGNNs have shown success in modeling complex phenomena such as user-item interactions, social dynamics, and temporal knowledge graphs (Ennadir et al. 2024). In this work, we focus on CTDGs due to their expressive power and alignment with real-world interaction patterns.

Adversarial Attacks on Graphs

Adversarial attacks fall into two broad categories:
Evasion Attacks: Manipulates inputs at inference time to mislead the model after training.
Poisoning attacks: Injects perturbations *during training* to corrupt the learned representations. **LoRETTA is a poisoning attack**, aiming to degrade overall model performance by manipulating the training graph.

Poisoning static graphs vs. temporal graphs: A substantial body of literature explores poisoning attacks on static graphs (Bojchevski and Günnemann 2019; Li et al. 2020), where edge or node feature perturbations can significantly impact downstream tasks like node classification or link prediction. However, directly applying these methods to CTDGs is ineffective due to the evolving nature of temporal graphs. Specifically, past perturbations quickly lose influence as new edges arrive, and future edges are not observable at attack time, preventing attackers from crafting precise interventions. This dynamic nature limits the transferability of static attack strategies to temporal domains.

Poisoning DTDGs vs CTDGs: Chen et al. (2021); Sharma et al. (2023) extend poisoning strategies to DTDGs by targeting discrete graph snapshots. While effective in short-term settings, these methods are inherently limited in scope. Perturbations in DTDGs only influence the snapshot in which they occur and lack temporal propagation, making them ill-suited for CTDGs where edge interactions carry real-valued timestamps and temporal continuity is essential.

Existing Literature on Poisoning CTDGs: T-SPEAR (Lee, Lee, and Shin 2024) is, to our knowledge, the first dedicated poisoning attack on CTDGs. It trains a surrogate model to select edges that, when inserted, disrupt downstream predictions while adhering to 4 unnoticeability constraints (similar but weaker than ours). They also present an adversarial defense strategy against poisoning attacks called T-shield. While effective, T-SPEAR has critical limitations: (1) Assumes full access to the dataset, which is not true in real-world scenarios, as adversary often has access to just training dataset. (2) Requires significant compute to train and optimize a surrogate model. (3) Introduces edges but does not consider deleting critical ones, limiting perturbation scope. (4) The effectiveness of the attack depends on the accuracy of the surrogate model used.

Preliminaries

Dynamic graph setting: We consider a CTDG defined as $G = (V, E)$, where V is the set of n nodes, and E is the set of m timestamped edges. Each edge is a tuple (u, v, t), where $u, v \in V$ denote the interacting nodes and $t \in \mathbb{R}$ is the time of interaction. Multiple edges can exist between the same node pair at different times, capturing repeated interactions over time.

Temporal PageRank: Temporal PageRank (TPR) extends classical PageRank to temporal graphs by replacing static walks with time-respecting walks (Rozenshtein and Gionis 2016). Let $Z^T(v, u \mid t)$ denote the set of temporal walks (cf Appendix A.12) from node v to u that occur strictly before time t. The probability of a walk z is defined as:

$$\Pr'[z \in Z^T(v, u \mid t)] = \frac{c(z \mid t)}{\sum_{z' \in Z^T(v,x|t),\, x \in V,\, |z'|=|z|} c(z' \mid t)},$$

where $c(z \mid t)$ is the decay-weighted count of z.

Let $e_i = ((u_{i-1}, u_i, t_i), (u_i, u_{i+1}, t_{i+1}))$ denote consecutive edge pairs in walk z, and let $N(u_i, t_i, t_{i+1})$ denote the set of intervening interactions $\{(u_i, y, t') \mid t' \in [t_i, t_{i+1}], y \in V\}$ at node u_i between times t_i and t_{i+1}. The decay-weighted count is then:

$$c(z \mid t) = (1 - \beta) \prod_{e_i \in z} \beta^{|N(u_i, t_i, t_{i+1})|}.$$

Transitions are penalized exponentially by the number of intervening interactions to model temporal decay. The final TPR score of node u at time t is:

$$r(u, t) = \sum_{v \in V} \sum_{k=0}^{t} (1 - \alpha)\alpha^k \sum_{\substack{z \in Z_T(v,u|t) \\ |z|=k}} \Pr'[z \mid t],$$

where α is the jump probability. This formulation biases the walk toward shorter, temporally coherent paths and naturally captures influence drift in evolving graphs.

Problem Formulation

Attacker's objective. Given a clean CTDG G, the attacker seeks to degrade the performance of a Temporal Graph Neural Network (TGNN) on dynamic link prediction by injecting adversarial edges and/or removing crucial ones. The perturbed graph is denoted as:

$$\tilde{G} = (V, (E \setminus E') \cup \tilde{E}),$$

where $E' \subset E$ are the removed edges, $\tilde{E}$ are the adversarial (inserted) edges, and the total number of modifications is bounded by a fixed budget Δ. This is a **poisoning attack**, the graph is modified before training, and the TGNN is trained on the corrupted $\tilde{G}$.

Attacker's knowledge. We assume a strict black-box setting: the attacker has no knowledge of the TGNN architecture, loss function, or gradients. However, the attacker can observe the training portion of the dataset, a common assumption in graph poisoning (Lee, Lee, and Shin 2024; Ma, Ding, and Mei 2020b; Zügner, Akbarnejad, and Günnemann 2018). Unlike previous work like T-spear (Lee, Lee, and Shin 2024), we do not assume access to validation/test splits.

Unnoticeability Constraints To ensure the attack remains stealthy and realistic, we adopt and strengthen the four unnoticeability constraints introduced by T-spear:

- **(C1) Perturbation Budget:** The total number of modified edges (inserted + removed) is bounded by $\Delta = \lfloor p \cdot |E| \rfloor$, where p is the perturbation rate.
- **(C2) Temporal Feasibility:** Timestamps of inserted edges are drawn from the same distribution as original timestamps, i.e., $\tilde{t} \sim P_t(E)$.
- **(C3) Node Activity Window:** Inserted edges may only connect nodes that have been active within a temporal window W around the chosen timestamp $\tilde{t}$. That is, both endpoints of $\tilde{e} = (u, v, \tilde{t})$ must appear in $V_{i,W} = \{u_j, v_j \mid j \in [i - W + 1, i]\}$.
- **(C4) Degree Preservation:** We impose a stronger variant of node-level stealth by matching the degree distribution of each node before and after perturbation. This prevents attackers from noticeably increasing or decreasing a node's interaction footprint.

Methodology

We propose a two-stage adversarial attack framework, **Sparsification** followed by **Replacement**, designed to degrade TGNNs trained on CTDGs. In this section, we explain LoReTTA in detail.

Step 1: Sparsification

We perform the Sparsification of a CTDG in 16 different ways. We classify them broadly into broad types:

Edge Sparsification Strategy: For each timestamp t_i, we construct a static aggregated graph $G^{(i)} = (V, E^{(i)})$ where $E^{(i)} = \{(u, v, t) \in E : t \leq t_i\}$. We define a general temporal importance function $H : V \times V \times \mathbb{R} \to \mathbb{R}$ that computes the importance score for any edge (u, v, t_j) as:

$$H(u, v, t_j) = f((u, v), G^{(j)}), \tag{1}$$

where f represents any graph-theoretic heuristic applied to edge (u, v) within the context of static aggregation $G^{(j)}$. Edges are ranked by $H(u, v, t_j)$ in descending order, and the top-Δ highest scoring edges are selected for removal to achieve the desired p. We consider 4 heuristic strategies and also a Random edge removal strategy (cf. Appendix A.1) .

Timestamp Sparsification Strategy: We identify critical timestamps by measuring temporal drift in node importance through generalized distance metrics. Let $r^{(t_i)} \in \mathbb{R}^{|V|}$ denote the TPR vector at timestamp t_i. We define temporal drift at timestamp t_i as:

$$\delta^{(t_i)} = d(r^{(t_i)}, r^{(t_{i-1})}), \tag{2}$$

where $d(\cdot, \cdot)$ represents a generalized distance function measuring importance shift between consecutive timestamps. High $\delta^{(t_i)}$ values indicate volatile periods where minimal perturbations significantly impact learned temporal dynamics. We rank all timestamps by their drift values and select the top timestamps with highest $\delta^{(t_i)}$ scores. If the final timestamp exceeds Δ, we select exactly the remaining number of edges at that timestamp to meet the budget. Similar to edge sparsification, where heuristic scores can be replaced with various graph-based metrics, temporal drift computation supports multiple distance metrics (eg. ℓ_2-norms, cosine distance, Jaccard distance etc). We consider with 11 different distance metrics (cf Appendix A.1) .

Step 2: Adversarial Negative Sampling

After removing critical edges in Step 1, we insert feasible negative edges to maintain graph density and temporal dynamics. This is designed to respect unnoticeability constraints (C2–C4) without requiring access to model gradients or internals. Our negative sampling algorithm operates as follows: for each removed edge, we insert a new edge that (i) mimics natural temporal patterns, (ii) connects temporally active nodes, and (iii) preserves the degree distribution.

C2: We first sample candidate timestamps from the empirical distribution of existing edge times using kernel density estimation (KDE). This ensures inserted edges follow the same temporal rhythm as genuine interactions, satisfying temporal stealth (**C2**).

C3: Given a sampled timestamp, we construct a candidate node pool by selecting nodes active within a local time window W around that timestamp. This enforces the activity constraint (**C3**), ensuring inserted edges do not connect inactive or dormant nodes, which would appear suspicious under standard monitoring tools. We adopt the same W used by T-spear (Lee, Lee, and Shin 2024). For bipartite datasets, endpoints are sampled from disjoint node sets. Additionally, we enforce that node pairs (u, v) are new to the graph—no prior interaction exists in either direction—preventing semantic leakage and contradictions with observed history.

C4: To maintain degree consistency (**C4**), we track deletions and insertions per node and update candidate pools accordingly. New edges are selected to keep each node's in-degree and out-degree statistically unchanged, guarding against structural anomalies such as hub overloading or connectivity spikes.

Our constraint-aware algorithm iteratively samples valid edge–timestamp pairs that satisfy all stealth constraints. When no valid candidates remain due to constraint overlap or capacity limits, we trigger a recovery step that reinitializes the KDE over timestamps and samples a fresh set of candidate times. This resampling enables the algorithm to explore new feasible timestamps without relaxing any constraints, ensuring the perturbation budget is met while preserving unnoticeability.

These steps form a principled, constraint-driven negative sampling routine. Unlike learned adversarial strategies requiring surrogate models, our method generates structurally and temporally plausible perturbations without supervision, yielding effective attacks in low-resource settings. Alternative approaches like random sampling (selecting edges arbitrarily out of temporal order) or Havel-Hakimi (Kleitman and Wang 1973) are insufficient—random sampling lacks strategic impact, while Havel-Hakimi preserves degrees without addressing temporal dynamics or unnoticeability requirements. We provide the pseudocode for LoRETTA's negative sampling algorithm in Appendix .

Complexity Analysis

We analyze the time and space complexity of each component in LoRETTA.

Time complexity of Temporal PageRank *The time complexity of the TPR algorithm is $O(|E| + |V|)$, where $|E|$ is the number of edges in the input graph and $|V|$ is the number of vertices.*

Space Complexity of Temporal PageRank *Let $|V|$ be the number of distinct nodes and $|T|$ the number of distinct timestamps in the input sequence. The combined data structures of TPR and TER occupy*

$$O\big(|V|\,|T|\big) \text{ memory.}$$

Time Complexity of TimeStamp Selector *The time complexity of our method is*

$$O\big(|V|(\log|V| + d_{\max} k \log k) + |E|\log|E|\big),$$

where $|V|$ the number of vertices $|E|$ is the number of edges, d_{max} is the maximum degree of any vertex, and k is the average number of timestamps per edge.

Space complexity of Timestamp Selector *Let W be the length of the time window used when gathering candidate edges. Algorithm Timestamp Selector uses*

$$O\big(|V| \;+\; |E|\,W\big) \textit{ memory.}$$

Detailed proofs and empirical analysis are provided in Appendix , where we also demonstrate that **LoRETTA achieves up to 10× speedup over T-spear**.

Experiment Setup

We provide details about datasets and models along with metrics and baselines used in experiments to show the effectiveness of our method (see the supplementary material) .

Datasets: We perform experiments on 4 benchmark real-world datasets: Wikipedia (Kumar, Zhang, and Leskovec 2019), MOOC (Feng, Tang, and Liu 2019), UCI (Panzarasa, Opsahl, and Carley 2009), and Enron (Shetty and Adibi 2004). For all datasets, we adopt a consistent chronological split of 70%-15%-15% for training-validation-test sets, following the methodology in (Rossi et al. 2020). Detailed descriptions of these datasets are provided in Appendix.

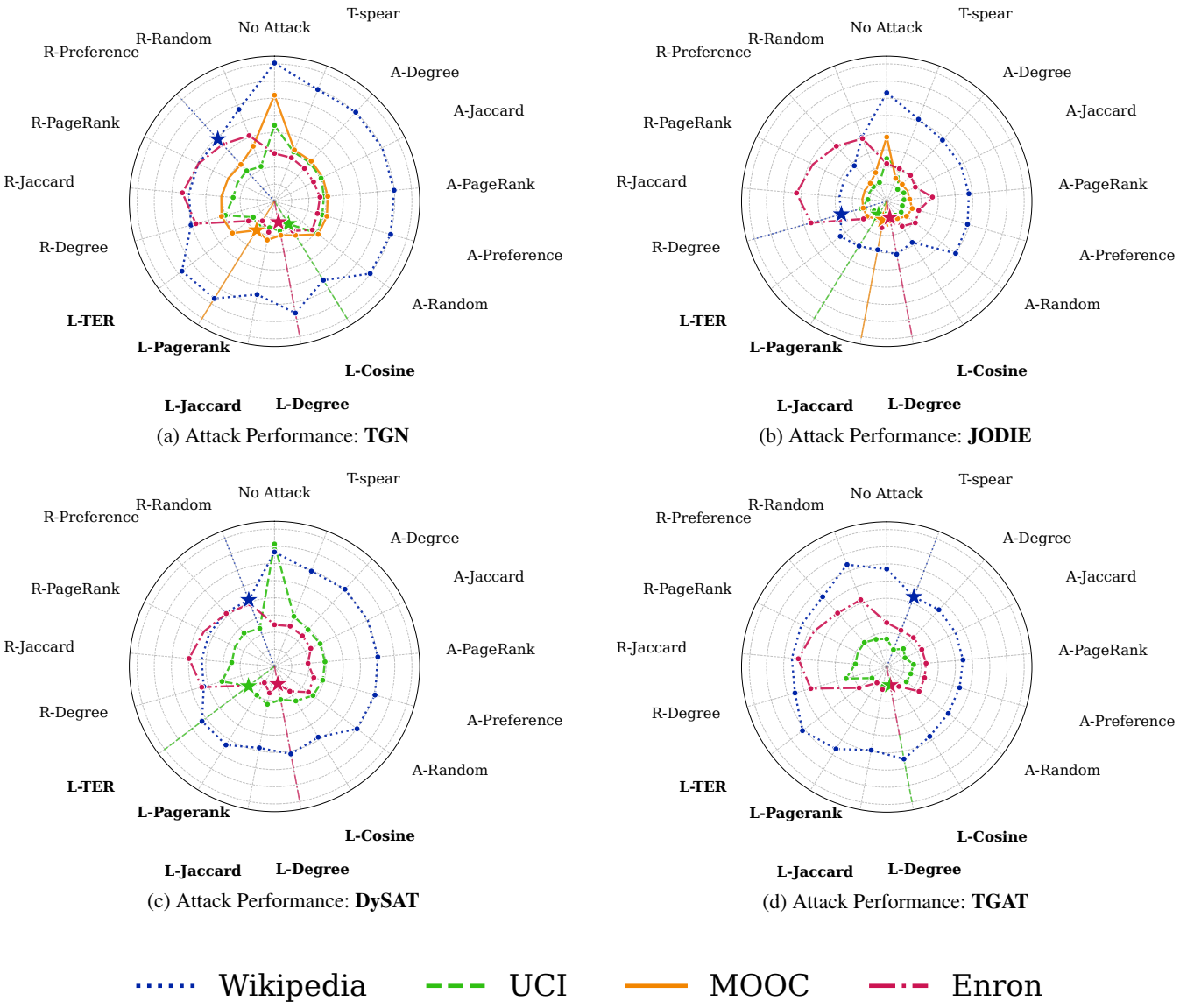
Models: To evaluate the effectiveness of LoRETTA, we conduct experiments on 4 SotA TGNNs: TGN (Rossi et al. 2020), JODIE (Kumar, Zhang, and Leskovec 2019), TGAT (Xu et al. 2020), and DySAT (Sankar et al. 2018).

Metrics: To ensure consistency and comparability, we utilize Mean Reciprocal Rank (MRR), which is a robust and a widely recognized evaluation metric for ranking tasks.

Baselines: We evaluate our method against 11 baselines: T-spear (SotA CTDG poisoning attack), 5 edge addition baselines, i.e., ADD, which were used in the T-spear paper, and 5 edge removal baselines, i.e., REM, extrapolated from static graph methods. Unlike the original T-spear work, which poisons the entire dataset, including the validation and test sets, **we poison only the training set even for T-spear to ensure a fair comparison with our method, which accounts for any differences in reported T-spear performance numbers**. Detailed explanation of baselines is in Appendix .

Discussion

In this section, we evaluate performance of LoRETTA against baselines (cf. Appendix A), highlight its robustness to SotA defenses (cf. Appendix A),



(a) Attack Performance: **TGN**

(b) Attack Performance: **JODIE**

(c) Attack Performance: **DySAT**

(d) Attack Performance: **TGAT**

Figure 2: Performance of our attack on each model across datasets. **(Prefixes: R=REM, A=ADD, L=LoReTTA.)** Lower values indicate stronger attack impact. Each concentric circle represents a 10% increase in MRR; numerical labels are omitted to ensure readability. Clean baselines are shown under No-Attack. DySAT and TGAT fail on MOOC due to Out-Of-Memory error. Naive Jaccard is omitted for bipartite graphs (MOOC and UCI). Dashed radial lines highlight the most effective attack for each dataset-model pair. LoReTTA-Degree is the best metric for both Enron and UCI on TGAT.

anomaly detection algorithms (cf. Appendix A) and present our ablation studies(cf. Appendix A)

Attack Performance

We perform a comprehensive evaluation of our proposed attack method against 11 baseline approaches (see the supplementary material) . Figure 2 presents the average performance of 5 runs of all victim models under various attack methods at a perturbation rate of 0.3. The results for MOOC on TGAT and DySAT are not included due to out-of-memory errors caused by the size of the models.

LoReTTA consistently beats all the baselines in the majority of cases, demonstrating superior attack effectiveness. The single exception occurs on the Wikipedia dataset, where REM baselines achieve comparable or slightly better performance. This anomaly stems from Wikipedia's unique structural properties that make sparsification-only attacks particularly effective. Specifically, Wikipedia exhibits highly modular semantic clusters where users contribute predominantly to domain-specific areas. Sparsification attacks exploit this modularity by removing high-signal edges within semantic clusters, directly undermining the model's ability to capture

Attack	SVD		Cosine		T-shield-F		T-shield	
	Wiki	UCI	Wiki	UCI	Wiki	UCI	Wiki	UCI
Degree	40.96(1.29)	34.04(1.78)	27.57(1.05)	8.05(0.37)	10.45(0.65)	5.78(0.62)	10.30(0.90)	5.98(0.32)
PageRank	40.27(0.88)	34.44(1.26)	27.55(1.40)	7.72(0.38)	10.98(0.74)	5.86(0.40)	11.03(0.80)	5.62(0.41)
Cosine	46.01(1.36)	34.34(1.26)	17.87(2.11)	7.63(0.90)	10.32(0.57)	6.25(0.41)	9.62(1.08)	6.10(0.79)
Jaccard	48.94(0.97)	33.70(1.69)	19.33(2.19)	7.74(0.57)	9.08(0.74)	5.69(0.52)	9.61(0.37)	5.50(0.27)
TER	47.99(1.39)	34.46(0.94)	32.11(0.97)	7.40(0.76)	11.31(1.07)	6.20(0.57)	11.38(1.05)	5.96(0.41)
Clean	80.50(0.50)	44.2(0.4)	80.50(0.50)	44.2(0.4)	80.50(0.50)	44.2(0.4)	80.50(0.50)	44.2(0.4)

Table 1: Performance (% MRR) of variants of LORETTA against 4 SotA defenses. Values shown as mean(std). Wiki refers to Wikipedia dataset. The "Clean" row establishes the baseline performance of TGN without any perturbations or defenses applied.

critical local coherence patterns. In contrast, our method's negative sampling component introduces cross-domain edges that inadvertently provide implicit regularization, encouraging broader generalization across otherwise disjoint clusters and partially counteracting the sparsification damage. However, it should be noted that REM baselines achieve this performance by adhering only to the C1 constraint, thereby disrupting CTDG structure more extensively than our approach, which respects both C1 and C2 constraints. We provide a detailed analysis of this phenomenon in Appendix.

Importantly, LORETTA demonstrates clear superiority across all datasets, proving LORETTA's general effectiveness. Furthermore, LORETTA doesn't just work for a specific heuristic. We consistently outperform many baselines for all 5 chosen sparsification strategies, showing the stability of LORETTAand its independence from specific sparsification techniques. The attack performance in tabular form is in Appendix. We also assess dominance using a one-sided exact binomial sign test across baselines; see Appendix.

Robustness against SotA Defenses

We evaluate the robustness of our proposed LORETTA against 4 SotA adversarial defense methods using TGN on Wikipedia and UCI datasets with p as 0.3, as presented in Table 1. The defenses include 2 static graph methods (SVD-based reconstruction and cosine similarity filtering) and 2 variations of Tshield (Lee, Lee, and Shin 2024) (Tshield-F and Tshield), which represent the current SotA CTDG defense methods.

Our results demonstrate that even after applying these defense mechanisms, victim models consistently fail to recover their original clean performance levels. Notably, we observe that Cosine similarity filtering, T-shield, and T-shield-F defenses cause additional performance degradation beyond that induced by LORETTA alone. This counterintuitive result occurs because these filtering-based approaches attempt to identify and remove adversarial edges from the dataset. However, when adversarial edges are incorrectly identified, true edges are inadvertently removed, resulting in further performance deterioration. We experimentally validate this hypothesis in Appendix , where we discovered that only approximately 30% of the filtered edges correspond to actual adversarial modifications. This finding not only validates the unnoticeability of LORETTA but also underscores its robustness against current defense strategies, highlighting the urgent need for more robust defense methods

Robustness against Anomaly Detection methods

When adversaries employ data poisoning techniques to degrade model performance, anomaly detection systems can be deployed to identify adversarial edges and effectively clean the dataset before training. Consequently, it is critical for poisoning attacks to remain undetectable to such systems. To demonstrate LORETTA's stealth capabilities, we evaluate its robustness against 4 SotA edge-stream anomaly detection methods: MIDAS (Bhatia et al. 2020), F-FADE (Chang et al. 2021), AnoEdge-L, and AnoEdge-G (Bhatia et al. 2023). We focus on unsupervised approaches due to their practical relevance, as supervised methods require labeled anomaly datasets that are typically unavailable in real-world deployment scenarios.

Attack	Anomaly Detection Methods							
	MIDAS		F-FADE		AnoEdge-L		AnoEdge-G	
	P	R	P	R	P	R	P	R
Wikipedia Dataset								
Degree	0.35	0.52	1.00	0.28	0.47	0.45	0.39	0.55
PageRank	0.35	0.53	1.00	0.30	0.49	0.46	0.40	0.55
Cosine	0.26	0.70	1.00	0.21	0.50	0.38	0.46	0.74
Jaccard	0.28	0.67	1.00	0.18	0.50	0.36	0.42	0.75
TER	0.37	0.73	1.00	0.15	0.46	0.31	0.29	0.73
UCI Dataset								
Degree	0.48	0.29	0.98	0.44	0.28	0.26	0.32	0.44
PageRank	0.51	0.33	0.98	0.45	0.33	0.26	0.34	0.48
Cosine	0.51	0.34	0.33	0.67	0.46	0.51	0.59	0.82
Jaccard	0.57	0.36	0.28	0.84	0.49	0.45	0.65	0.84
TER	0.63	0.62	0.26	1.00	0.52	0.37	0.44	0.67

Table 2: Anomaly–class precision (P) and recall (R) of anomaly detection methods after perturbed using LORETTA

Table 2 presents the precision and recall scores of anomaly detection methods on the adversarial edges class when LORETTA perturbs 30% of the training data for the TGN model across Wikipedia and UCI datasets. Each anomaly detection system assigns a score to every edge, where higher scores indicate greater likelihood of being anomalous. We employ Youden's thresholding method (Youden 1950) to determine the optimal threshold for anomaly classification, as it minimizes both false positives and false negatives.

The results reveal a fundamental trade-off inherent in anomaly detection systems. High precision with low recall indicates accurate identification of true anomalies but at the

Method	Wikipedia				UCI			
Attacker's Knowledge	**0.2**	**0.4**	**0.6**	**0.8**	**0.2**	**0.4**	**0.6**	**0.8**
Cosine	57.16 ± 0.77	58.90 ± 0.85	58.00 ± 0.91	58.25 ± 1.25	33.58 ± 1.58	35.84 ± 1.68	33.42 ± 1.78	34.27 ± 2.40
Random	62.61 ± 0.81	63.17 ± 1.26	62.61 ± 1.13	62.61 ± 0.47	35.06 ± 1.45	34.83 ± 2.00	34.60 ± 2.52	35.32 ± 1.34
Jaccard	58.56 ± 0.82	58.45 ± 0.79	58.60 ± 0.51	59.97 ± 1.47	35.54 ± 1.66	36.39 ± 1.68	32.69 ± 1.53	33.95 ± 1.76
Pagerank	61.18 ± 0.69	64.43 ± 0.42	64.91 ± 1.33	67.40 ± 0.89	35.01 ± 2.32	35.83 ± 1.87	32.33 ± 2.62	36.10 ± 0.65
Degree	60.78 ± 0.70	64.40 ± 0.53	65.11 ± 0.31	67.06 ± 1.01	34.64 ± 2.67	35.11 ± 1.85	32.21 ± 1.86	35.38 ± 1.45
TER	60.78 ± 0.63	63.04 ± 1.04	65.93 ± 0.51	66.78 ± 1.48	34.74 ± 2.54	33.57 ± 2.47	32.53 ± 1.83	36.36 ± 0.86

Table 3: Performance (%) of different removal strategies under varying levels of adversary's knowledge on the Wikipedia and UCI datasets. Each value is reported as mean ± standard deviation over 5 runs.

cost of allowing many perturbed edges to remain undetected. Conversely, high recall with low precision typically results in numerous false positives that erroneously classify legitimate edges as anomalies, potentially degrading model performance when removed, as observed in §. Critically, across all dataset-attack-detector combinations, no method achieves both precision and recall scores exceeding 0.7 simultaneously. This demonstrates that LoRETTA successfully evades SotA anomaly detection systems and remains practically undetectable in realistic deployment scenarios, establishing its effectiveness as a stealthy adversarial attack. We also report in Appendix the AUPRC score (area under the precision–recall curve) for each anomaly detector, i.e., a threshold-free summary that integrates performance across all decision thresholds.

Ablation Studies

In this section, we analyze the sensitivity of LoRETTA to p and the knowledge of the training dataset.

Impact of the Knowledge of Adversary: Table 3 demonstrates the effect of increasing knowledge and access to the training dataset to the adversary on TGN performance degradation using LoRETTA on Wikipedia. We discover that increasing adversarial knowledge does not always lead to more effective attacks. Heuristic strategies like pagerank and degree underperform—even degrade—with more knowledge. These methods remove high-centrality nodes assuming they're critical, but often eliminate redundant or noisy nodes, unintentionally improving performance. This suggests that vulnerabilities are concentrated in a small set of influential nodes; once these are removed, additional knowledge yields diminishing or negative returns.

Interestingly, random removal shows stable performance across knowledge levels, suggesting non-monotonicity in attack impact. The system's response is highly non-linear; only specific perturbations degrade performance. These findings indicate that attack effectiveness depends on targeting specific vulnerable components rather than simply having more information about the system. For the adversary knowledge ablation, we perturb up to 30% of the edges to study the effect of strong adversarial influence across strategies. We experimentally validate our hypothesis in Appendix.

Impact of Perturbation Rate: Figure 3 demonstrates the effect of increasing p on TGN performance degradation using LoRETTA on Wikipedia, compared against T-spear. Performance degrades steadily with increasing p before plateauing, indicating that LoRETTA's deterministic sparsification algorithm targets influential edges first. Beyond a certain threshold, removing additional edges yields diminishing returns as the algorithm begins targeting less critical connections that contribute minimally to model learning. Notably, LoRETTA consistently outperforms the SotA baseline across all p, demonstrating superior effectiveness despite operating as a surrogate-free attack.

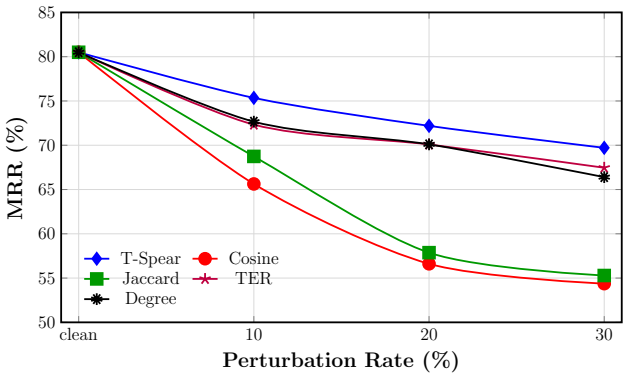


Figure 3: Effect of perturbation rate p on the performance of LoRETTA using different sparsification strategies.

Conclusion

In this paper, we propose LoRETTA, a novel attack strategy for TGNNs that exploits temporally significant edges to achieve effective perturbations. Our method consistently outperforms baselines, including T-SPEAR, across various datasets and models, highlighting the critical role of temporal dynamics in graph vulnerabilities. LoRETTA is also robust to SotA adversarial defenses and anomaly detection systems.

References

Bhatia, S.; Hooi, B.; Yoon, M.; Shin, K.; and Faloutsos, C. 2020. Midas: Microcluster-based Detector of Anomalies in Edge Streams. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 3242–3249.

Bhatia, S.; Wadhwa, M.; Kawaguchi, K.; Shah, N.; Yu, P. S.; and Hooi, B. 2023. Sketch-based Anomaly Detection in Streaming Graphs. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 93–104.

Bojchevski, A.; and Günnemann, S. 2019. Adversarial Attacks on Node Embeddings via Graph Poisoning. In *International Conference on Machine Learning*, 695–704.

Cai, B.; Xiang, Y.; Gao, L.; Zhang, H.; Li, Y.; and Li, J. 2022. Temporal Knowledge Graph Completion: A Survey. *arXiv preprint:2201.08236*.

Cencetti, G.; Santin, G.; Longa, A.; Pigani, E.; Barrat, A.; Cattuto, C.; Lehmann, S.; Salathe, M.; and Lepri, B. 2021. Digital Proximity Tracing on Empirical Contact Networks for Pandemic Control. *Nature communications*, 12(1): 1655.

Chang, Y.-Y.; Li, P.; Sosic, R.; Afifi, M.; Schweighauser, M.; and Leskovec, J. 2021. F-fade: Frequency Factorization for Anomaly Detection in Edge Streams. In *Proceedings of the 14th ACM international conference on web search and data mining*, 589–597.

Chen, J.; Zhang, J.; Chen, Z.; Du, M.; and Xuan, Q. 2021. Time-Aware Gradient Attack on Dynamic Network Link Prediction. *IEEE Transactions on Knowledge and Data Engineering*, 35: 2091–2102.

Cong, W.; Zhang, S.; Kang, J.; Yuan, B.; Wu, H.; Zhou, X.; Tong, H.; and Mahdavi, M. 2023. Do We Really Need Complicated Model Architectures for Temporal Networks? *arXiv preprint:2302.11636*.

Dai, H.; Li, H.; Tian, T.; Huang, X.; Wang, L.; Zhu, J.; and Song, L. 2018. Adversarial Attack on Graph Structured Data. In *International Conference on Machine Learning*, 1115–1124.

Deng, S.; Rangwala, H.; and Ning, Y. 2019. Learning Dynamic Context Graphs for Predicting Social Events. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1007–1016.

Ennadir, S.; Gandler, G. Z.; Cornell, F.; Cao, L.; Smirnov, O.; Wang, T.; Zólyomi, L.; Brinne, B.; and Asadi, S. 2024. Expressivity of Representation Learning on Continuous-Time Dynamic Graphs: An Information-Flow Centric Review. arXiv:2412.03783.

Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; and Yin, D. 2019. Graph Neural Networks for Social Recommendation. In *The World Wide Web Conference*, 417–426.

Feng, W.; Tang, J.; and Liu, T. X. 2019. Understanding Dropouts in Moocs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 517–524.

Gao, C.; Wang, X.; He, X.; and Li, Y. 2022. Graph Neural Networks for Recommender System. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, 1623–1625.

Jiang, W.; and Luo, J. 2022. Graph Neural Network for Traffic Forecasting: A Survey. *Expert Systems with Applications*, 207: 117921.

Kleitman, D.; and Wang, D. 1973. Algorithms for constructing graphs and digraphs with given valences and factors. *Discrete Mathematics*, 6(1): 79–88.

Kumar, S.; Zhang, X.; and Leskovec, J. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1269–1278.

Lee, D.; Lee, J.; and Shin, K. 2024. Spear and Shield: Adversarial Attacks and Defense Methods for Model-Based Link Prediction on Continuous-Time Dynamic Graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 13374–13382.

Li, J.; Zhang, H.; Han, Z.; Rong, Y.; Cheng, H.; and Huang, J. 2020. Adversarial Attack on Community Detection by Hiding Individuals. In *Proceedings of the web conference 2020*, 917–927.

Ma, J.; Ding, S.; and Mei, Q. 2020a. Towards More Practical Adversarial Attacks on Graph Neural Networks. *Advances in Neural Information Processing Systems*, 33: 4756–4766.

Ma, J.; Ding, S.; and Mei, Q. 2020b. Towards More Practical Adversarial Attacks on Graph Neural Networks. arXiv:2006.05057.

Ma, Y.; Guo, Z.; Ren, Z.; Tang, J.; and Yin, D. 2020. Streaming Graph Neural Networks. In *Proceedings of the 43rd international ACM SIGIR Conference on Research and Development in Information Retrieval*, 719–728.

Panzarasa, P.; Opsahl, T.; and Carley, K. M. 2009. Patterns and Dynamics of Users' Behavior and Interaction: Network Analysis of an Online Community. *Journal of the American Society for Information Science and Technology*, 60: 911–932.

Pareja, A.; Domeniconi, G.; Chen, J.; Ma, T.; Suzumura, T.; Kanezashi, H.; Kaler, T.; Schardl, T.; and Leiserson, C. 2020. Evolvegcn: Evolving Graph Convolutional Networks for Dynamic Graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 5363–5370.

Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; and Bronstein, M. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. *arXiv preprint:2006.10637*.

Rozenshtein, P.; and Gionis, A. 2016. Temporal PageRank. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, 674–689.

Sankar, A.; Wu, Y.; Gou, L.; Zhang, W.; and Yang, H. 2018. Dynamic Graph Representation Learning via Self-Attention Networks. *arXiv preprint:1812.09430*.

Sankar, A.; Wu, Y.; Gou, L.; Zhang, W.; and Yang, H. 2020. 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*, 519–527.

Sharma, K.; Trivedi, R.; Sridhar, R.; and Kumar, S. 2023. Temporal Dynamics-Aware Adversarial Attacks on Discrete-Time Dynamic Graph Models. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2023–2035.

Shetty, J.; and Adibi, J. 2004. The Enron Email Dataset Database Schema and Brief Statistical Report. *Information Sciences Institute Technical Report, University of Southern California*, 4: 120–128.

So, M. K.; Tiwari, A.; Chu, A. M.; Tsang, J. T.; and Chan, J. N. 2020. Visualizing Covid-19 Pandemic Risk Through Network Connectedness. *International Journal of Infectious Diseases*, 96: 558–561.

Trivedi, R.; Farajtabar, M.; Biswal, P.; and Zha, H. 2019. Dyrep: Learning Representations over Dynamic Graphs. In *International Conference on Learning Representations*.

Wu, S.; Sun, F.; Zhang, W.; Xie, X.; and Cui, B. 2022. Graph Neural Networks in Recommender Systems: A Survey. *ACM Computing Surveys*, 55(5): 1–37.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2020. Inductive Representation Learning on Temporal Graphs. *arXiv preprint:2002.07962*.

Youden, W. J. 1950. Index for Rating Diagnostic Tests. *Cancer*, 3(1): 32–35.

Yu, B.; Yin, H.; and Zhu, Z. 2017. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. *arXiv preprint:1709.04875*.

Zheng, Y.; Wei, Z.; and Liu, J. 2023. Decoupled graph neural networks for large dynamic graphs. *arXiv preprint arXiv:2305.08273*.

Zügner, D.; Akbarnejad, A.; and Günnemann, S. 2018. Adversarial Attacks on Neural Networks for Graph Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2847–2856.

Zügner, D.; Akbarnejad, A.; and Günnemann, S. 2018. 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.

Zügner, D.; and Günnemann, S. 2024. Adversarial Attacks on Graph Neural Networks via Meta Learning. arXiv:1902.08412.