Imperceptible Adversarial Attacks on Discrete-Time Dynamic Graph Models

Kartik Sharma  
Georgia Institute of Technology  
ksartik@gatech.edu

Rakshit Trivedi  
Harvard University  
rstrivedi@seas.harvard.edu

Rohit Sridhar  
Georgia Institute of Technology  
orohit@gatech.edu

Srijan Kumar  
Georgia Institute of Technology  
srijan@gatech.edu

Abstract

Real-world graphs such as social networks, communication networks, and rating networks are constantly evolving over time. Many architectures have been developed to learn effective node representations using both graph structure and its dynamics. While the robustness of static graph models is well-studied, the vulnerability of the dynamic graph models to adversarial attacks is underexplored. In this work, we design a novel attack on discrete-time dynamic graph models where we desire to perturb the input graph sequence in a manner that preserves the structural evolution of the graph. To this end, we motivate a novel Time-Aware Perturbation (TAP) constraint, which ensures that perturbations introduced at each time step are restricted to only a small fraction of the number of changes in the graph since the previous time step. We present a theoretically-grounded Projected Gradient Descent approach for dynamic graphs to find the effective perturbations under the TAP constraint. Experiments on dynamic link prediction show that our approach is up to 4x more effective than the baseline methods for attacking these models under the novel constraint. TD-PGD also outperforms the existing baselines on the node classification task by up to 6x and is 2x more efficient in running time than the best-performing baseline.

1 Introduction

Graph Neural Networks (GNNs) have been shown to be vulnerable to adversarial perturbations [2, 10, 15, 23, 39, 45]. This has raised major concerns against their use in important industrial applications such as friend/product recommendation [33, 36, 43] and fraud detection [14, 44]. However, these advancements in designing attack and defense mechanisms have predominantly focused on GNN models for static, non-evolving graphs. In reality, the graph structure evolves with time as new interactions happen and new connections are formed [18, 21]. GNN models that incorporate the temporal information are shown to outperform their static counterparts in modeling dynamic networks on tasks such as predicting link existence in the future [7, 13, 16, 20, 34].

However, the vulnerability of dynamic graph models to adversarial perturbations is less studied. The design of adversarial attacks for dynamic graphs is challenging for two reasons — (1) Attacks must simultaneously optimize both the edge(s) to perturb and when to perturb them, and more importantly, (2) Attacks must preserve the original graph evolution after perturbation in order to be less detectable. In particular, attacks that disturb original graph evolution are not desired since they can be detected as anomalies by defense mechanisms, e.g. graph anomaly detection methods [1]. Therefore, it is crucial
to formulate adversarial attacks over snapshots such that they do not significantly alter the original change in the graph structure.

In this work, we introduce a novel **Time-Aware Perturbation (TAP)** constraint to formulate imperceptible attacks on discrete-time dynamic graphs. This constraint asserts that the number of modifications added at a timestep should only be a small fraction of the actual number of changes with respect to the preceding timestep. We theoretically show that perturbations made under TAP constraint preserves the rate of change both in the structural and the embedding spaces. To find effective attacks under this proposed constraint, we consider a targeted, white-box, and evasion setting. As noted in Table 1, prior works are unable to conduct attacks on dynamic graphs under this setting. Thus, we present a theoretically-grounded Projected Gradient Descent approach for dynamic graphs (TD-PGD) following the TAP constraint. Experiments on 4 datasets and 3 dynamic graph models (victim models) show that the proposed TD-PGD is up to 4x more effective than the baseline attacks on dynamic link prediction and node classification tasks. We further propose a new metric — Embedding Variability (EV), as a measure to assess the relative variability in the perturbed embedding space. Our contributions can be summarized as follows:

1. We introduce a novel **Time-Aware Perturbation (TAP)** constraint to make less detectable perturbations in discrete-time dynamic graphs.
2. We show theoretically that the proposed TAP constraint preserves the original rate of the structural change and embedding change in the dynamic graphs.
3. We present a theoretically-grounded PGD-based white-box attack to find effective imperceptible attacks on dynamic graphs under the novel TAP constraint.
4. We show that TD-PGD outperforms the baselines across 4 different datasets and 3 victim models on both dynamic link prediction and node classification tasks.

### 2 Related Work

**Representation Learning for Dynamic Graphs.** GNNs have been combined with sequential modeling architectures [16] to model dynamic graphs. For instance, discrete-time graphs have been modeled by using GNNs and RNNs together in a pipeline [27,28] or an embedded manner [7,30]. Attention-based models have also been proposed to jointly encode the graph structure and its dynamics [34]. For continuous-time graphs, both RNN [20,24,37,38] and attention-based models [31,40] have been proposed such that embeddings are updated in real time, upon an occurrence of a new event such as the addition/deletion of an edge.

**Adversarial attacks on graphs.** Static GNNs are known to be vulnerable to adversarial attacks in different settings [15]. White-box attacks are studied assuming complete knowledge of the underlying model [39,42]. Limiting the model knowledge, gray-box [45] and black-box attacks [10] have also been proposed. In comparison, the literature on adversarial attacks for dynamic graphs is scarce. Time-aware Gradient Attack (TGA) [9] is a white-box evasion attack which greedily selects the perturbations across time under a budget constraint. In addition, attacks to poison training data [8] and black-box attacks using RL approaches [11] have also been proposed.

**Imperceptible perturbations.** The most common strategy to formulate imperceptible attacks on graphs is to bound the total number of perturbations. Various strategies, in addition to the budget constraint, have been developed to make the attacks imperceptible for static graphs. These include rewiring the perturbed edges [25,26], only perturbing the low degree nodes [29], and preserving the

<table>
<thead>
<tr>
<th>Method</th>
<th>Dynamic</th>
<th>White-box</th>
<th>Evasion</th>
<th>Targeted</th>
<th>Evolution-aware</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGD [42]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>IG-JSMA [42]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Fan et al. [11]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dyn-Backdoor [8]</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGA [9]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>TD-PGD (proposed)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 1: Comparison of our attack with existing works on graph adversarial attacks.
degree/feature distribution statistics [45]. In the case of dynamic graphs, perturbations must preserve the temporal flow to be imperceptible. Traditional anomaly detection algorithms flag an instance to be anomalous if distance between consecutive snapshots crosses a threshold [11]. In particular, Graph Edit Distance and Hamming distance between adjacency matrices have been used to monitor communication networks [5, 35]. Neural approaches have looked at the consecutive change in the embedding space to detect anomalies without feature extraction [6, 13].

3 Methodology

**Problem** Let \( G_1, G_2, \ldots, G_T \) be the original graph snapshots and \( G'_1, G'_2, \ldots, G'_T \) be the corresponding perturbed snapshots. Note that \( G_i = (X_i, A_i) \) where \( X_i, A_i \) are the node features and the adjacency matrix for snapshot \( i \), respectively. Also, let \( M \) be a victim dynamic graph model that we want to attack and let \( f_M \) be a function that generates the corresponding node embeddings of \( G_i \) given \( G_{1:i-1} \). Let \( y_{task} \) be the actual labels for a given task (for dynamic link prediction, these correspond to binary labels representing link existence in the future snapshot).

Then, the objective of the attacker is to introduce structural perturbations \( S_t = A'_t - A_t \) at each timestep \( t < T \) such that the model inference at timestep \( T \) for the target entities \( E_{tg} \) deteriorates. More formally, the attacker solves the following optimization problem:

\[
\max_{A'_1, A'_2, \ldots, A'_{T-1}} L_{task}(\hat{y}_{task}(f_M(A'_{1:T-1})), y_{task}, E_{tg}) \text{ such that } C(A'_{1:T-1}) \text{ holds}
\]

for some constraint function \( C \) on the perturbed adjacency matrices \( A'_t \) for each time \( t \). Here, \( \hat{y}_{task} \) denotes the predicted labels for the given task and \( L_{task} \) is a task-specific loss, for example, a binary cross entropy (CE) loss for link prediction.

The constraint function \( C \) is designed to ensure imperceptibility of the adversarial perturbations. In the literature, a budget constraint has been widely used to enforce imperceptibility in graphs [10] and computer vision [12]. However, this constraint only bounds the total amount of perturbations that can be introduced by an attacker. When the input is dynamic, as in the case of dynamic graphs, the perturbations should be constrained in the context of how the input evolves. However, since the budget constraint completely ignores the graph dynamics, it could lead to a drastic change in the evolution trend of the graph, violating the imperceptibility requirement. For instance, with the budget constraint, all the perturbations can be made at a single time step, leading to an anomalous spike, which would be easily detected as a possible attack by graph anomaly detection methods for dynamic graphs [11, 5, 35]. Thus, a constraint is desired which would ensure that the perturbations do not disrupt the evolving trend of the dynamic graphs.

3.1 Time-Aware Perturbation (TAP) Constraint

The simplest measure to study evolution is to consider the change in the input between consecutive time steps. Thus, for a discrete-time input \( \{x_t\} \), this corresponds to considering the discrete-time differential norm at time \( t \), given by \( dx_t = \|x_t - x_{t-1}\| \). Then, we propose

**Proposition 1** The number of perturbations introduced to input \( x \) at time step \( t \) must not be more than a fraction \( \epsilon \) times the differential at \( t \), i.e. \( TAP(\epsilon) := \|x'_t - x_t\| \leq \epsilon dx_t \forall t \).

For the case of dynamic graphs, when the graph structure evolves (for example, in social networks and transaction networks [30, 34]), this constraint becomes \( \|A'_t - A_t\| \leq \epsilon dA_t \). Alternatively, a dynamic graph may also involve a temporally-evolving signal at each node [22, 29, 32], in which case, this constraint becomes \( \|X'_t - X_t\|_1 \leq \epsilon dX_t \).

In this work, we focus on dynamic graph structures such that the constraint \( C \) (in Equation [1]) for the perturbations is a TAP constraint. The optimization problem for the attacker, thus, becomes

\[
\max_{A'_1, A'_2, \ldots, A'_{T-1}} L_{task}(\hat{y}_{task}(f_M(A'_{1:T-1})), y_{task}, E_{tg}) \text{ such that } \forall t \in (1, T): \frac{\|A'_t - A_t\|}{\|A_t - A_{t-1}\|} \leq \epsilon \text{ and } \|A'_t - A_t\| \leq \epsilon_1,
\]
where $\epsilon, \epsilon_1$ are given parameters for this optimization. We use $\|\cdot\|$ to denote the 1-norm of the matrix flattened into a vector, unless otherwise mentioned.

Implications. We show that TAP constraint has the following implications on the perturbations:

1. **Perturbations made under TAP constraint preserves the average rate of structural change.**

   **Theorem 1** Let $\bar{dA} = \frac{1}{T} \sum_1^T dA_t$, $dA_T^\top = \frac{1}{T} \sum_1^T dA^\top_t$. Then,
   \[
   \|\bar{dA}\| \leq \alpha \|\bar{A}\| + \beta,
   \]
   for some constants $\alpha, \beta \in \mathbb{R}_{\geq 0}$.

   **Proof.** By the definition of the perturbation matrix $S_t$, $dA_t^\top = \|A_t - A_{t-1}\| = \|(A_t + S_t) - (A_{t-1} + S_{t-1})\|$. Then, using triangle inequality, we get $dA_t^\top \leq \|A_t + S_t\| + \|A_{t-1} + S_{t-1}\|$. Again using triangle inequality, $dA_t^\top \leq \|A_t\| + \|S_t\| + \|A_{t-1}\| + \|S_{t-1}\|$. Now, since $\|S_t\| \leq \epsilon dA_t$ and $\|A_t\|$ is a constant, we get $dA_t^\top \leq \epsilon dA_t + \epsilon dA_{t-1} + C$, for some constant $C$.
   \[
   \|\bar{dA}\| = \frac{1}{T} \sum_1^T dA_t^\top \leq \frac{1}{T} \sum_1^T (\epsilon dA_t + \epsilon dA_{t-1} + C) \leq 2\epsilon \frac{1}{T} \sum_1^T dA_t + C.
   \]
   Hence, we get that $\|\bar{dA}\| \leq \epsilon \|\bar{A}\| + \beta$, for some constants $\alpha, \beta \geq 0$. \blacksquare

2. **Perturbations made under TAP constraint preserves the rate of embedding change.**

   **Theorem 2** Let $dZ_t = \|Z_t - Z_{t-1}\|$. Then, for some constants $\gamma, \delta \in \mathbb{R}_{\geq 0}$,
   \[
   dZ^\top \leq \gamma dZ + \delta.
   \]

   **Proof.** Note that $Z_t = f(A_t, A_{t-1}, \ldots, A_1)$. Thus, we consider a stacked vector of flattened matrices $q_{\leq t} = (q_{t_1}, q_{t_2}, \ldots, q_{t_0}, 0, 0, \ldots, 0)$, where $q_{t_i}$ is the flattened vector of $A_i$. and we prepend $(t - \tau)$ 0s to make all vectors $q_{\leq t}$ of fixed dimension $t$. Then, by Cauchy’s Mean Value Theorem in several variables, we have $Z_t - Z_{t-1} \leq \nabla f \cdot (q_{\leq t} - q_{\leq t-1})$, which gives us $\|Z_t - Z_{t-1}\| \leq \|\nabla f\| \|q_{\leq t} - q_{\leq t-1}\|$ by Cauchy-Schwarz inequality. We note that $\|q_{\leq t} - q_{\leq t-1}\|_1 = \sum_1^t ||q_{t_1} - q_{t_1-1} - q_{t_2} - \ldots - q_{t_{i-1}}|| = \sum_1^t ||q_{t_1} - q_{t_1-1}|| = \sum_1^t ||q_{t_1} - q_{t_1-1}|| = T \bar{dA}$. Thus, we have $\|Z_t - Z_{t-1}\|_F \leq C \bar{dA}$ for some constant $C \geq 0$. Using Theorem 1, we get $dZ' \leq C \bar{dA}^\top \leq \bar{dA}^\top + B$, for $A = C \bar{dA} + B = C \bar{dA}$. By mean-value theorem, we also have $dZ_t \geq \|\nabla f_{t-1}\| \|q_{\leq t} - q_{\leq t-1}\| \geq C \bar{dA}$. Hence, $dZ^\top \leq \gamma dZ + \delta$ for some constants $\gamma, \delta \geq 0$. \blacksquare

3.2 **Attack methods under TAP constraint**

While the TAP constraint allows us to limit the effect of the perturbations on the graph’s evolution, it is not clear how one can efficiently find perturbations that maximize a loss function under this constraint. To this end, we present two algorithms to solve the optimization problem of Equation 2.

**Greedy.** A greedy strategy can be adopted to find effective perturbations under our TAP constraint. In this approach, perturbations are selected in a greedy manner based on their gradient values with respect to the downstream loss. However, this does not scale well as one needs to find gradient values corresponding to all the perturbations, which would be $O(\mathcal{V})$, where $\mathcal{V}$ denotes the set of nodes. Thus, inspired by [9], we find the perturbations in two steps — first, we find the top-gradient perturbation at each time step and then, select the one that reduces the prediction probability the most. In particular, we greedily select the perturbations with the lowest probability such that TAP($\epsilon$) is not violated for any time-step. We defer the full algorithm to Appendix A.

**Projected Gradient Descent for Dynamic Graphs (TD-PGD).** Since the constrained optimization in Equation 2 has a general continuous objective, a greedy approach is only sub-optimal (even for a simpler convex objective) with no theoretical guarantees. A more standard approach to do optimization under a convex constraint is to use projected gradient descent (PGD) [3][4]. Since our problem is in discrete-space, we first relax it into continuous space, find the solution using PGD and then, randomly round it to obtain a valid solution for the discrete problem. In particular, we relax the perturbation matrix $S_t$ into a continuous vector $s_t$ and show that a closed-form projection operator exists for the TAP($\epsilon$) constraint. Algorithm 1 demonstrates the steps involved in this approach (TD-PGD), following the result of Theorem 3.

**Theorem 3** Suppose $S$ denotes the feasible perturbation space for the constraints $\|A_t - A_t\| \leq \epsilon$ for all $1 < t < T$ and $\|A_t - A_{t-1}\| \leq \epsilon_1$. Then, one can project a vector $a_t$ onto $S$ using
Algorithm 1 Projected Gradient Descent for dynamic graphs

Require: TAP variables $\epsilon_i$ (from Theorem 3), Initial perturbation vector $s^{(0)}$, Loss function $\mathcal{L}_{\text{task}}$, actual labels $\hat{y}_{\text{task}}$, Target entities $E_{tg}$, Time steps $T$, Initial learning rate $\eta_0$, Iterations $N$

Ensure: Perturbation vector $s^{(i)}$ preserves TAP($\epsilon$) at every time step $t$

1: for $i = 1$ to $N$ do
2:    Gradient descent: $a^{(i)} = s^{(i-1)} - \eta t \nabla \mathcal{L}_{\text{task}}(\{G_t \oplus s^{(i-1)}; \forall t\}, \hat{y}_{\text{task}}, E_{tg})$;
3:    Projection: For all $t \in [1, T - 1]$: $s^{(i)}_{t} = \Pi_{S}(a^{(i)}_{t})$ according to Equation 5
4: $S_{t} \sim \text{Bernoulli}(s^{(N)}_{t})$ such that $S_{t} \leq \epsilon_{i}$ for all $t$

"the following projection operator:

$$\Pi_{S}(a_{t}) = \begin{cases} P_{[0,1]}(a_{t} - \mu_{t}) & \text{if } \exists \mu_{t} > 0 : 1^{T}P_{[0,1]}(a_{t} - \mu_{t}) = \epsilon_{t} \\ P_{[0,1]}(a_{t}) & \text{if } 1^{T}P_{[0,1]}(a_{t}) \leq \epsilon_{t} \end{cases} \quad (5)$$

where $\epsilon_{t} = \epsilon \cdot d \mathcal{A}_{t} = \epsilon \| \mathcal{A}_{t} - \mathcal{A}_{t-1} \| _{f}$ for $t > 1$, and $P_{[0,1]}(x) = x$ if $x \in [0,1]$. If $x < 0$, and 1 if $x > 1$.

Proof. Please see Appendix B for the proof. ■

Following [42], we use the bisection method [3] to solve the equation $1^{T}P_{[0,1]}(a_{t} - \mu_{t}) = \epsilon_{t}$ in $\mu_{t}$ for $\mu_{t} \in [\min(a_{t} - 1), \max(a_{t})]$. This converges in the logarithmic rate, i.e. it takes $O(\log_{2}(\max(a_{t}) - \min(a_{t} - 1)/\epsilon))$ time for $\epsilon$-error tolerance.

4 Experimental Setup

Datasets. We use 3 datasets for dynamic link prediction — Radoslaw[4], UCI[5] and Reddit[6]. Radoslaw and UCI are email communication networks, where two users are connected if they have an email communication at a given time. Reddit is a hyperlink network with directed connections between subreddits if there is a hyperlink from one to the other [19]. For node classification, we use one publicly-available dataset, DBLP-5 [41]. This is a co-author network with node attributes as word2vec representations of the author’s papers. There are 5 node labels representing the different fields that the authors belong to. Please refer to Appendix C for more details regarding the datasets.

Attack Methods. We consider 4 attack methods to find adversarial perturbations in our setting.

1. TD-PGD: Algorithm 1 shows the proposed approach of projected gradient descent with a valid projection operator for the TAP constraint.
2. TGA($\epsilon$): Greedily selects the perturbation with the highest gradient value of the loss.
3. DEGREE: Flips the edges (adds or deletes if already there) attached with the highest degree nodes in the graph at each time step while making at most $\epsilon d \mathcal{A}_{t}$ perturbations.
4. RANDOM: Randomly flip (add or delete) at most $\epsilon d \mathcal{A}_{t}$ edges at each time step $t$.

Victim Models. We test the performance of the above on 3 discrete-time dynamic graph models.

1. GC-LSTM [7] embeds GCN into an LSTM to encode the sequence of graphs.
2. EVOLVEGCN [30] uses a recurrent model (RNN-LSTM) to evolve the weights of a GCN. We use the EVOLVEGCN-O version for our experiments.
3. DySAT [34] is an attention-based architecture utilizing joint structural and temporal self-attention.

Metrics. We use the relative drop, as defined below, to evaluate the efficacy of the attack methods.

$$\text{Rel. Drop} \% = \frac{\text{Perturbed performance} - \text{Original performance}}{\text{Original performance}} \times 100, \quad (6)$$

where performance is evaluated using ROC-AUC for dynamic link prediction and using Accuracy for node classification.

References:


In order to evaluate the detectability of the attack methods, we propose a novel metric Embedding Variability (EV) to compare the consecutive embedding difference for the perturbed graph and that for the original graph. Consecutive embedding difference has been used to identify anomalies in the data [13]. Here, we measure how the range of this difference changes due to the perturbation. In particular, we consider

\[ EV(Z, Z') := \left| 1 - \frac{\max_\tau dZ'_\tau - \min_\tau dZ'_\tau}{\max_\tau dZ_\tau - \min_\tau dZ_\tau} \right| \]  

This measures the relative variability of the consecutive change in the embedding space. For the attacks to be less detectable, this metric should be close to 0. Note that we do not directly optimize for this metric in our constraint.

5 Results

We compare the performance of different attack methods for the 3 victim models on link prediction and node classification tasks. For all the experiments, we vary \( \epsilon \) from 0 to 1 and fix \( \epsilon_1 = \min_{t>1} \epsilon_t := \epsilon dA_t \).

5.1 Dynamic Link Prediction

In this section, we show the attack performance on the task of dynamic link prediction [30, 34]. The task here is to predict whether a link \((u, v)\) will appear or not at the future timestep. The objective of the attacker, thus, is to introduce perturbations in the past time steps to make the model mispredict link’s existence in future. We test the victim models on the final snapshot for a set of target links. We consider 3 different sets of 100 positive and 100 negative random targets and show the mean relative ROC-AUC drop with error bars.

Figure 1 shows the performance of different attack methods on this task across different datasets and models. TD-PGD outperforms the other baselines in all cases, except in GC-LSTM model trained on Reddit. Moreover, TD-PGD is able to drop the AUC by up to 4 times the baselines and lead to \( \sim 100\% \) drop in the AUC, completely flipping the prediction. We also note that TD-PGD often has a continuously decreasing slope and its performance saturates much later than the other baselines. The second best baseline is often TGA(\( \epsilon \)) but in many cases, it is only as good as random. One can also note that EVOLVEGCN shows a larger drop than the other 2 models across all datasets. This may pertain to the lower model complexity of EVOLVEGCN compared to others.

Detectability of the attack methods: Here, we use the EV metric, as defined in Section 4, to assess how detectable the attack methods are under our constraint. Table 2 compares the attack performance of the best method TD-PGD at \( \epsilon = 0.5 \) and the corresponding variability in the embeddings, as given by Equation 7. One can note that the median value of EV is close to zero in all cases, exceeding 0.2 only in a couple of cases. In particular, we find that one can drop the performance by over 95% while changing the range of consecutive embedding distance by a factor of \( \sim 1 \pm 0.1 \) in over 50%

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Rel. Drop</th>
<th>EV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radoslaw</td>
<td>DYSAT</td>
<td>47.03 (5.42)</td>
<td>0.00 (0.00, 0.04)</td>
</tr>
<tr>
<td></td>
<td>EVOLVEGCN</td>
<td>91.61 (3.58)</td>
<td>0.25 (0.03, 0.79)</td>
</tr>
<tr>
<td></td>
<td>GC-LSTM</td>
<td>52.63 (5.09)</td>
<td>0.07 (0.01, 0.19)</td>
</tr>
<tr>
<td>UCI</td>
<td>DYSAT</td>
<td>4.02 (2.08)</td>
<td>0.06 (0.01, 0.39)</td>
</tr>
<tr>
<td></td>
<td>EVOLVEGCN</td>
<td>96.21 (0.17)</td>
<td>0.11 (0.01, 0.62)</td>
</tr>
<tr>
<td></td>
<td>GC-LSTM</td>
<td>16.12 (0.75)</td>
<td>0.20 (0.03, 0.85)</td>
</tr>
<tr>
<td>Reddit</td>
<td>DYSAT</td>
<td>23.24 (4.18)</td>
<td>0.02 (0.00, 0.34)</td>
</tr>
<tr>
<td></td>
<td>EVOLVEGCN</td>
<td>79.31 (3.13)</td>
<td>0.13 (0.02, 1.23)</td>
</tr>
<tr>
<td></td>
<td>GC-LSTM</td>
<td>15.59 (1.28)</td>
<td>0.13 (0.02, 0.84)</td>
</tr>
</tbody>
</table>

Table 2: Comparison of attack performance and detectability (refer Section 5) for TD-PGD at \( \epsilon = 0.5 \). Median values are noted with 10 and 90 percentile range values in the parentheses.
Figure 1: Attack performance on dynamic link prediction task across datasets and models.

5.2 Node Classification

In this section, we compare the attack performance on the task of semi-supervised node classification [30]. In this task, the objective is to predict node labels of a set of nodes while knowing the labels of the remaining nodes. In our experiments, we consider a 60−20−20 split on the data. The objective of the attacker is to introduce perturbations at each snapshot under the TAP constraint such that the CE loss for the target node is maximized.

Figure 2 shows the effect of structural perturbations on node classification task by different attack strategies for the 3 models. Due to sparsity of edges in DBLP-5, randomly chosen targets only allow for a very small number of perturbations under the TAP constraint. Therefore, we consider the performance on 50 top-degree nodes for each class. Results show that TD-PGD outperforms the
baselines in all models except DySAT, where all attacks perform almost equally. In particular, TD-PGD causes a 30% drop in 
\textsc{EvolveGCN} while the baselines only lead to a drop of 5%.

### 5.3 Running Time

Figure 3 compares the running time per target for different attack methods on the largest dataset, Reddit. The times are averaged over 200 targets from 3 different seeds and error bars note the standard deviation. \textsc{TGA}(\epsilon) is the most expensive method in terms of time and scales almost linearly with \(\epsilon\) (capped at 300 s). TD-PGD takes around half the time than \textsc{TGA}(\epsilon) and remains almost constant with increase in the parameter \(\epsilon\).

### 6 Conclusion

Our work has shown that dynamic graph models can be attacked in an imperceptible manner. We introduce a novel Time-Aware Perturbation (TAP) constraint to devise imperceptible perturbations in discrete-time dynamic graphs, preserving the graph evolution within a factor. We hope that our work serves as a first step towards opening exciting research avenues on studying attacks and defense mechanisms for both discrete and continuous-time dynamic graphs. Some limitations of our current exposition can be noted. First, the proposed method TD-PGD is not memory-efficient as it stores a perturbation vector corresponding to each perturbation. Second, randomly rounding the solution to discrete space may lead to suboptimal perturbations. Future work can study more effective and efficient methods to attack dynamic graphs under TAP constraint.
Acknowledgements

This research/material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00112290102 (subcontract No. PO70745), NSF ITE-2137724, NSF ITE-2230692, Microsoft AI for Health, IDEaS at Georgia Institute of Technology, and Adobe Inc. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the position or policy of DARPA, DoD, NSF, and SRI International and no official endorsement should be inferred. We also thank the members of CLAWS research group and the anonymous reviewers for their constructive feedback.

References


Appendix

A Greedy approach

Algorithm 2 Greedy algorithm

Require: TAP variables $\epsilon_t$ (from Equation 5), Initial perturbation vector $s^{(0)}$, Loss function $L_{task}$, Probability function $p_M$ predicting link existence probability, actual labels $y_{\text{task}}$, Target entities $E_{tg}$, Time steps $T$.

1: For all $t$: $G_t^i \leftarrow G_t$.
2: while True do
3: for $t = 1$ to $T$ do
4: $s_t \leftarrow \phi$
5: for $n_{tg}$ in $E_{tg}$ do
6: $\text{grads}[v] \leftarrow \partial L_{task}(G_{t-1}^{[0,T]})/\partial(n_{tg}, v)$ for $v$ in $V$.
7: Pick the first $v$ in the descending order of $\text{grads}$ such that $(n_{tg}, v) \notin S$.
8: $s_t$.append($(n_{tg}, v)$).
9: $\text{probs}$.append($p_M(G_t^i \oplus s_t)$).
10: Pick $s_t$ in the descending order of $\text{probs}$ such that $\|S_t \oplus s_t\| \leq \epsilon_t$.
11: if no $\tau$ found then
12: break
13: $G_t^i \leftarrow G_t^i \oplus s_t$.
14: $S_t$.append($s_t$).

B Proof of Theorem 3

Suppose $S$ denotes the feasible perturbation space for the constraints $\|A_t^i - A_t\|/\|A_t - A_{t-1}\| \leq \epsilon$ for all $1 < t < T$ and $\|A_t^i - A_1\| \leq \epsilon_1$. Then, one can project a vector $a_t$ onto $S$ using the following projection operator:

$$\Pi_S(a_t) = \begin{cases} 
P_{[0,1]}(a_t - \mu_t) & \text{if } \exists \mu_t > 0: \ 1^T P_{[0,1]}(a_t - \mu_t) = \epsilon_t \\
\frac{P_{[0,1]}(a_t)}{1^T P_{[0,1]}(a_t)} & \text{if } 1^T P_{[0,1]}(a_t) \leq \epsilon_t 
\end{cases}$$

(8)

where $\epsilon_t = \epsilon d A_t = \epsilon \|A_t - A_{t-1}\|$ for $t > 1$, and $P_{[0,1]}(x) = x$ if $x \in [0,1]$, 0 if $x < 0$, and 1 if $x > 1$.

Proof. Let the perturbation vector be $s = [s_1, s_2, \cdots, s_{T-1}]^T$. Then, for all $t \in [2, T-1]$, since $s_t = A_t - A_{t-1}$, our constraint becomes $\|s_t\| \leq \epsilon d A_t \leq \epsilon \|A_t - A_{t-1}\| =: \epsilon_t$, which reduces to $1^T s_t \leq \epsilon_t$. For $t = 1$, we have $\|A_t^i(V_t) - A_1(V_1)\| \leq \epsilon_1$, which also becomes $1^T s_1 \leq \epsilon_1$. Hence, we have the constraint $1^T s_t \leq \epsilon_t$ for all $t$.

By definition, then, projection operator must be $\Pi_S(a) = \text{arg min}_{s \in S} \frac{1}{2} \|s - a\|^2$, where $S = \{s \in [0,1]^{|t|}: 1^T s_t \leq \epsilon_t \ \forall t\}$. This reduces to the following optimization problem:

$$\Pi_S(a) = \text{arg min}_{s \in S} \frac{1}{2} \sum_{t \in [1, T]} \|s_t - a_t\|^2 + I_{[0,1]}(s_t),$$

(9)

such that $\forall t \in [1, T]: 1^T s_t \leq \epsilon_t$

where $I_{[0,1]}(x) = 0$ if $x \in [0,1]$ and $\infty$ otherwise.

This can be solved by the Lagrangian method. We note that the Lagrangian function of the above optimization problem is $L(s, a, \mu) = \sum_{t \in [1, T]} \left(\frac{1}{2} \|s_t - a_t\|^2 + I_{[0,1]}(s_t)\right) + \mu_t (1^T s_t - \epsilon_t)$.

$$\partial L/\partial s_t = 0 \implies s_t = a_t - \mu_t.$$ However, if $s_{t,i} < 0$ or $s_{t,i} > 1$ for any $i$, then $L = \infty$. Thus, the minimizer to the above function is $s_t = P_{[0,1]}(a_t - \mu_t)$, where $P_{[0,1]}(x) = x$ if $x \in [0,1]$, 0 if $x < 0$ and 1 if $x > 1$. In addition, the solution must satisfy the following KKT conditions $\forall t$:
µₜ(1ᵀsₜ - εₜ) = 0, \quad \mu_t \geq 0, \quad (3) 1ᵀsₜ ≤ εₜ.

If µₜ > 0, then we must have 1ᵀP[0,1](aₜ - µₜ) = εₜ. Otherwise if µₜ = 0, then 1ᵀP[0,1](aₜ) ≤ εₜ.

C Data statistics and pre-processing

Table 3 shows the statistics of the datasets used in this paper. DBLP-5 is the node classification dataset with 5 labels while the others are dynamic link prediction datasets with varying sizes. The datasets span over differing periods of time. We split each of them into finite number of timesteps to keep a large enough number of time steps while maintaining a realistic period of splitting. Radoslaw is split using a 3-week period in 13 snapshots while the 13 snapshots in UCI denote a 2-week period. The Reddit dataset is spanned over 3 years, thus we use a 2-month split to obtain the 20 snapshots. We use the publicly available pre-processed data for DBLP-5.

For datasets with no node features, i.e., Radoslaw, UCI, and Reddit, we use uniformly random features with dimension 10. The pre-processed DBLP-5 has 100 node features for each node.

<table>
<thead>
<tr>
<th># Nodes</th>
<th># Edges</th>
<th># Time-steps</th>
<th># Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radoslaw</td>
<td>167</td>
<td>22K</td>
<td>13</td>
</tr>
<tr>
<td>UCI</td>
<td>1.9K</td>
<td>24K</td>
<td>13</td>
</tr>
<tr>
<td>Reddit</td>
<td>35K</td>
<td>715K</td>
<td>20</td>
</tr>
<tr>
<td>DBLP-5</td>
<td>6.6K</td>
<td>43K</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 3: Description of the datasets.

D Additional details on experimental setup

D.1 Setup

We consider a targeted setting with single targets, that are selected using either a random sampling or a degree-biased sampling. Each target is attacked one-by-one and the total performance of an attacker is measured using either an ROC-AUC or an Accuracy over the set of sampled targets.

D.2 Hyperparameters

For TD-PGD optimization, we used ADAM optimizer [17] with the initial learning rate of 10. The initial perturbation vector s(0) was initialized with all ones, thus, giving each perturbation an equal chance at the start. The algorithm was run for 50 iterations. We use Binary Cross Entropy loss for training dynamic link prediction and Weighted Cross Entropy for node classification. Further, we stop the greedy search if the time taken exceeds 300 s, which is at least 3 times that of TD-PGD.

D.3 Victim Model training

Table 4 shows the performance on the test set of the victim models on different datasets. The performance is evaluated using ROC-AUC for the dynamic link prediction and using Accuracy for the node classification tasks. The test set for dynamic link prediction task denotes the edges and non-edges in the final snapshot, while for node classification, we use a 20% held-out set of nodes as the test set for noting the performance and attacking.

D.4 Implementation

We use the TorchGeometric-Temporal[3] implementation of EvolveGCN and GC-LSTM to train these models. For DySAT, we use the pytorch implementation[4]. We adapt the official code of

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Perf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radoslaw</td>
<td>DYSAT</td>
<td>0.743</td>
</tr>
<tr>
<td></td>
<td>EVOLVEGCN</td>
<td>0.742</td>
</tr>
<tr>
<td></td>
<td>GC-LSTM</td>
<td>0.813</td>
</tr>
<tr>
<td>UCI</td>
<td>DYSAT</td>
<td>0.952</td>
</tr>
<tr>
<td></td>
<td>EVOLVEGCN</td>
<td>0.873</td>
</tr>
<tr>
<td></td>
<td>GC-LSTM</td>
<td>0.968</td>
</tr>
<tr>
<td>Reddit</td>
<td>DYSAT</td>
<td>0.947</td>
</tr>
<tr>
<td></td>
<td>EVOLVEGCN</td>
<td>0.939</td>
</tr>
<tr>
<td></td>
<td>GC-LSTM</td>
<td>0.941</td>
</tr>
<tr>
<td>DBLP-5</td>
<td>DYSAT</td>
<td>0.699</td>
</tr>
<tr>
<td></td>
<td>EVOLVEGCN</td>
<td>0.687</td>
</tr>
<tr>
<td></td>
<td>GC-LSTM</td>
<td>0.695</td>
</tr>
</tbody>
</table>

Table 4: Performance (Perf.) on test set for different datasets and models. For Radoslaw, UCI, Reddit, we use ROC-AUC as the performance metric and for DBLP-5, Accuracy is used.

TGA\(^5\) to implement the greedy approach. For TD-PGD implementation, we adapt the DeepRobust\(^6\) implementation for dynamic graphs under TAP constraint.

### D.5 Potential negative societal impacts.

Our work has demonstrated the vulnerability of dynamic graph models under a practical TAP constraint. This means that these models are not suitable for deployment in environments where adversarial attacks under a TAP constraint are realistic. If these models are deployed in such a setting, adversarial attacks proposed in this work can be used by an adversary to hamper the predictions. However, it must be noted that none of the models used in this work are known to be deployed in the real-world. Secondly, TAP constraint with a large \(\epsilon\) may not be realistic in many applications. Thus, our study has an overall positive impact on the society as it allows practitioners to test the robustness of their models under a practical constraint before deploying in vulnerable environments. Although our attacks can be adopted by adversaries for negative use, it is important to put it out in the community so that the model designers are made aware of these attacks that are specifically designed to evade detection.

---

\(^5\)https://github.com/jianz94/tga

\(^6\)https://github.com/wenqifan03/RobustTorch