Sparse Vicious Attacks on Graph Neural Networks

Giovanni Trappolini, Valentino Maiorca, Silvio Severino, Emanuele Rodolà, Fabrizio Silvestri, and Gabriele Tolomei

Abstract—In this study, we introduce SAVAGE, a novel framework for sparse vicious adversarial link prediction attacks on graph neural networks (GNNs). While GNNs have been successful in link prediction tasks, they are susceptible to adversarial attacks where malicious nodes attempt to manipulate recommendations for a target victim. SAVAGE optimizes the attacker's goal to maximize attack effectiveness while minimizing the required malicious resources. Unlike existing methods with static resource-based upper bounds, SAVAGE employs a sparsity-enforcing mechanism to reduce the number of malicious nodes needed for the attack. Extensive experiments on real-world and synthetic datasets demonstrate the optimal tradeoff achieved by SAVAGE between a high attack success rate and the number of malicious nodes utilized. Furthermore, we demonstrate that SAVAGE can successfully target non-GNN-based link prediction systems, even those unknown at the time of the attack. This showcases the transferability of SAVAGE-generated attacks to other black-box methods for link prediction, highlighting its applicability across different real-world scenarios.

Impact Statement—Social networks have become integral to our daily lives, revolutionizing human interactions and emerging as powerful tools for online businesses and digital marketing. Their scalability to accommodate millions or billions of active users relies heavily on link prediction systems that create new connections. However, these systems are highly susceptible to attacks from adversarial agents seeking to manipulate reputations or undermine competitors. Existing attacks often exhaust the adversary's resources to achieve a high success rate. In contrast, our proposed framework and method enable effective attacks while minimizing the malicious resources required. This convenience exposes new threats and vulnerabilities that deserve careful investigation to make social networks more secure.

Index Terms—Adversarial attacks, graph neural network (GNN)-based recommender systems, link prediction.

I. INTRODUCTION

In REAL-LIFE social networks, connections among individuals typically form naturally within small and isolated communities. However, modern digital social networks with millions or billions of users require the development of automatic link prediction systems to suggest new connections between people. These machine-based systems, for instance, recommend follower–followee relationships between social network users, thereby allowing the rapid growth of new contacts. Generally speaking, link prediction methods estimate the probability that an edge between two initially disconnected nodes in a graph should exist [1], [2]. Therefore, they apply to any graph-structured data, particularly social network graphs. Several approaches to link prediction have been proposed in the literature [3]. Recently, methods based on graph neural networks (GNNs) have achieved the state-of-the-art performance [4]. When applied to social network graphs, the list of predicted links for a given user ranked by their estimated probability represents a set of recommended new contacts, which the user may eventually decide to follow. This step is crucial, especially for some authoritative, highly followed social network users, i.e., so-called influencers. Indeed, their reputation (and therefore social market value) may be affected by whom they choose to follow. Specifically, when the target of recommendations is an influencer, it is paramount to be accurate at suggesting the “right” set of candidate users to follow. As a matter of fact, being followed by an authoritative user can be seen as a sign of endorsement and ultimately could bring several advantages to the followee (e.g., higher revenue from advertising campaigns run on the social media platform due to increased popularity). Hence, malicious users are strongly interested in manipulating link prediction systems to artificially inflate their reputation on social media platforms, thus perpetrating their own (harmful) goals through adversarial attacks. For example, consider Sammy, a social network user interested in increasing their reputation. Ideally, they would have other influential users follow them to reach this goal. However, Sammy has no control over other users, making such a tactic unfeasible. Still, Sammy could adjust their existing connections (i.e., follow/unfollow users) with the hope that their name will show up in the list of recommended contacts for Terry, i.e., the user profile of a target influencer. More generally, Sammy would control a subset of nodes (in addition to themselves) and can modify their neighborhood to achieve their goal. Sammy may also be able to inject new nodes in the network and create connections between such “artificial” users and other existing nodes. All these operations (e.g., adding/deleting an edge between two nodes) have a cost that Sammy wants to account for and minimize. Although the body of work on adversarial attacks on link prediction is extensive [5], most of the literature does not entirely cover the scenario described in the abovementioned example. Indeed, most work assumes the attacker can perturb the original network topology...
by adding or deleting connections only between existing nodes to favor the malicious goal [6]. However, this setting is often unfeasible because it requires the direct control of real users, with prohibitive coordination costs. To overcome this limitation, vicious node adversarial attacks have been proposed [7]. Under this setting, the attacker can create new users ex novo and, by controlling these nodes, engineer the network to mount the attack. This approach is much more feasible than directly controlling actual nodes; indeed, the Internet is full of services selling fake accounts/comments/likes, and have been greatly improved with the recent developments in large language models and their augmented versions [8], [9], [10]. However, existing methods for conducting harmful node attacks neglect to take into account the cost associated with introducing these malicious nodes as part of the attacker’s objective [7]. These methods often assume that there is a constraint on the attacker’s allocated budget, which is typically exhausted in order to maximize the success rate of the attacks. Put simply, if an attacker is permitted to inject a maximum of $N$ malicious nodes into the network, they will be incentivized to employ all of them to increase the chances of a successful attack. We argue that this approach is impractical.

Hence, in this article we introduce sparse vicious attacks on graph networks (SA V AGE). This novel framework provides an original perspective on vicious node adversarial attacks on GNN-based link prediction methods for directed graphs. SAV AGE operates in a white-box setting and promotes a spare use of the malicious resources required by the attacker, i.e., the number of vicious nodes and their aggressiveness. Moreover, SAV AGE is general enough as it allows to frame existing vicious node attacks into it. To help the reader understand our method, we showcase it with a toy example in Fig. 1. Overall, the main contributions of this article are as follows.

1) We are the first to introduce a sparse comprehensive framework to mount adversarial attacks on GNN-link prediction systems: SAV AGE. SAV AGE balances attack effectiveness (AE) with the number of required malicious resources by enforcing sparsity in controlled nodes without imposing constraints on the attacker’s budget.

2) We show the feasibility and effectiveness of SAV AGE in attacking several GNN-based link prediction models trained on real-world and synthetic datasets via extensive experiments.

3) We demonstrate that adversarial attacks generated with SAV AGE can be successfully transferred to harm other link prediction systems (not necessarily GNN-based) under the more challenging black-box setting.

4) We release the source code and data used for this work.\footnote{[Online]. Available: https://github.com/GiovanniTRA/SA V AGE}

II. RELATED WORK

A. Link Prediction

Link prediction is one of the most investigated problems in modern graph analysis. For a comprehensive survey on this subject, we refer the reader to [1], [2], [11]. One class of simple yet effective approaches for link prediction is called heuristic methods. These assume that the existence of a link between two nodes depends on their “similarity.” In practice, they use predefined heuristics to compute node similarity scores as the link probability [3], [12]. Existing heuristics can be categorized based on the maximum hop of neighbors needed to calculate the node similarity score. For example, common neighbors [13], Jaccard [12], and preferential attachment [14] are first-order heuristics since they only involve the one-hop neighbors of two target nodes. Adamic–Adar [15] and resource allocation [16] are second-order heuristics, as they consider up to the two-hop neighborhood of the target nodes. Some high-order heuristics require knowing the entire network. Examples include Katz index [17], rooted PageRank [18], and SimRank [19]. Although

\footnote{[Online]. Available: https://ag.ny.gov/press-release/2021/attorney-general-james-issues-report-detailing-millions-fake-comments-revealing}

Fig. 1. Picture on the left (a) shows a simple directed graph, where the red node indicates the source ($s$) and the blue node the target ($t$) of the link prediction attack. In the middle, (b) a generic attack is depicted: the injection of vicious nodes (in purple) and new connections induce the link prediction system to suggest the connection between $t$ and $s$. Typically, these attacks exhaust the allocated attacker’s budget. On the other hand, SAV AGE (c) enforces sparsity on the amount of malicious resources used, deactivating unnecessary vicious nodes (grayed-out), while keeping the attack successful. (a) Original directed graph. (b) Generic attack to mispredict the existence of the link between $t$ and $s$. (c) Attack generated by SAV AGE is still successful but uses a sparser number of malicious resources.
working well in practice, heuristic methods have strong “hand-coded” assumptions on when links may exist. To overcome this limitation, the link prediction problem has been formulated as a standard binary classification task and solved using well-known supervised learning techniques [20]. With the advancements of deep learning and, specifically, GNNs, several approaches have recently proposed effective GNN-based link prediction methods [21], [22]. Roughly speaking, GNNs allow us to learn suitable representations (i.e., embeddings) of graph nodes by aggregating information derived from the local neighborhood of each individual node. These node embeddings are, in turn, used as input of a downstream link prediction function, thus making the whole model’s architecture end-to-end trainable. This work focuses on a specific adversarial attack on GNN-based link prediction.

### B. Adversarial Attacks to Link Prediction

Although very powerful, studies have shown that machine learning models may be vulnerable to so-called adversarial attacks, i.e., carefully crafted malicious examples designed to fool the predictive models. Typically, these adversarial inputs are generated by introducing minor—yet thoughtfully selected—perturbations to regular inputs. These attacks have been widely proven successful in many critical domains, such as image recognition [23] and malware detection [24]. Moreover, adversarial perturbation techniques exhibit similar similarities with powerful explanation methods, such as counterfactual examples [25], [26], [27], [28], [29], [30], [31]. However, a few works have explored how effective such adversarial attacks may be for link prediction algorithms, especially those based on GNNs, which require the attacker to perturb the input graph. Among these studies, it is worth mentioning [6], [32], [33], [34]. We refer the reader to [35] for a comprehensive survey. The abovementioned methods assume that the attacker can control a subset of existing nodes. As stated in Section I, though, this is a pretty unrealistic premise, as it would be prohibitively expensive. To address this issue, Wang et al. [7] introduced the capability for the attacker to generate new (i.e., fake/vicious) nodes to mount the attack more efficiently. Furthermore, Wang et al. [36] proposed a linear approximation of the previous method to make it more scalable. Dai et al. [37] formulated a universal perturbation that can target multiple nodes and still remain effective. In both classic and vicious settings, all the methods presented so far introduce an unnoticeability constraint of the graph perturbation. This constraint is usually of two kinds: either the attacker is given a fixed budget to spend [6], [38], or some rules are imposed to control the difference between the original and the perturbed graph, such as in [33]. Either way, this is treated as an upper bound on the amount of “malicious resources” used to implement the attack and does not promote frugality. In fact, the adversary typically saturates the constraint to ensure the attack is successful. On the other hand, the method we propose in this work enforces the sparsity of the malicious resources used by the attacker (see Fig. 1).

### III. BACKGROUND AND NOTATION

We consider a directed graph $G = (V, E)$ with $n$ nodes $V$, and $m$ edges $E$. The structure of $G$ is encoded by its adjacency matrix $A \in \{0, 1\}^{n \times n}$, where $A_{i,j} = 1$ iff $(i,j) \in E$. Notice that, in general, $A_{i,j} \neq A_{j,i}$, i.e., the adjacency matrix is not symmetric. A feature matrix $X \in \mathbb{R}^{n \times k}$ can also be used to associate features to nodes of $G$. We assume that a GNN $g$ learns a hidden representation of nodes in the graph (i.e., a node embedding). Such representation is, in turn, used for our downstream task of interest (i.e., link prediction). The embedding of each node is learned through $g$ by iteratively updating the node’s features based on the neighbors’ features. Formally, let $h_u^0$ denote the embedding of node $u \in V$ at the $0$th layer of $g$. Let $\mathcal{N}(u) = \{v \in V \mid (u,v) \in E\}$ be the one-hop neighborhood of $u \in V$, i.e., the set of all nodes that are adjacent to $u$. More generally, we can define the $l$-hop neighborhood of $u$ ($l \in \mathbb{Z}^+, l > 1$) as the set of nodes that are at most $l$ hops away from $u$, using the following recurrence relation:

$$\mathcal{N}(u, l) = \mathcal{N}(u, l-1) \cup \{v \in \mathcal{N}(u)\}.$$  

Hence, let us consider the subgraph $G_u^l$ of $G$ induced by $u$ and its $l$-hop neighborhood $\mathcal{N}(u)$ as relevant for the computation of $h_u^l$. Specifically, we consider $A_u^l$ and $X_u^l$ as the adjacency matrix, and the feature matrix of nodes in the subgraph $G_u^l$, respectively, and $h_u^l$ is computed by $g$ using the following equation:

$$h_u^l = g(A_u^l, X_u^l; \theta_g) = \phi(h_u^{l-1}, \psi(\{h_v^{l-1}\mid v \in \mathcal{N}(u)\}))$$  

where $\phi$ and $\psi$ are arbitrary differentiable functions (i.e., neural networks); $\psi$ is a permutation-invariant operator that aggregates the information from the $l$-hop neighborhood of $u$; $\phi$ updates the node embedding of $u$ by combining information from the previous layer; $\theta_g$ are the trainable parameters of $g$. The number of hidden layers in $g$ determines the set of neighbors included while learning each node’s embedding. Unless otherwise needed, we will consider $l$ fixed and omit the corresponding superscript. Given the graph $G^{obs} = (V, E^{obs})$, where $E^{obs} \subseteq E$ is a subset of the true links observed, the link prediction problem generally resorts to estimating the probability that an edge exists between two nodes $u$ and $v$, i.e., $\mathbb{P}(u,v) \in E \setminus E^{obs}$. Let $h_w = g(A_u, X_w; \theta_g)$ be the embedding of the generic node $w \in V$. Thus, we assume that the predicted link probability between two nodes $u$ and $v$ is approximated with a function $f$, defined as follows:

$$f(h_u, h_v; \theta_f) = f(g(A_u, X_u; \theta_g), g(A_v, X_v; \theta_g); \theta_f).$$  

Notice that $f$ can itself be another neural network (e.g., an MLP) that takes as input two node embeddings learned by the GNN $g$ and outputs their link prediction. A combined embedding $h_{u,v}$ of the two input nodes can be obtained, for example, via Hadamard (elementwise) product, i.e., $h_{u,v} = h_u \odot h_v$. The probability of a link existing between $u$ and $v$ can be computed as $f(h_u, h_v; \theta_f) = \sigma(\theta_f^T h_{u,v})$, where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid activation function at the last output layer of $f$. Eventually,
the link \((u, v)\) is created if the probability output by \(f\) is above a fixed threshold \(\tau\) (e.g., \(\tau = 0.5\)). In the following, we denote as \(\theta\) the overall set of end-to-end trainable parameters of \(f\) and \(g\), i.e., \(\theta = [\theta_f, \theta_g]\).

IV. PROBLEM FORMULATION

A. Attack Model

We consider a snapshot of a network, modeled as a directed graph \(G = (V, E)\), and a trained GNN-based link prediction model \(f\). In our attack model, we assume the existence of an attacker node \((s \in V)\) that can perturb the original graph \(G\) into \(\tilde{G} = (V', E')\). The perturbation represents the capability of the attacker to add a set of new vicious nodes \(V^{\text{new}}\) to \(G\), such that \(V' = V \cup V^{\text{new}}\). Furthermore, we suppose the attacker controls a subset of the original nodes \(V^{s} \subseteq V\) (trivially, \(s \in V^{s}\)) and is capable of adding/removing edges starting from the set of governed nodes \(V^{s} \cup V^{\text{new}}\). Notice that this assumption contrasts most mentioned methods in the literature, which instead assume a fixed set of existing nodes \(V\). Let \(E^{+} = (V^{s} \cup V^{\text{new}}) \times V'\) and \(E^{-} = (V^{s} \cup V^{\text{new}}) \times V'\) be the set of edges added and removed by the attacker, respectively. Eventually, \(\tilde{G} = (V', E')\) denotes the final perturbed graph, where \(E' = E \cup E^{+} \setminus E^{-}\). Thus, the goal of the attacker \(s\) is to transform \(G\) into \(\tilde{G}\), using the capabilities described previously, to ultimately induce the link prediction model \(f\) in recommending \(s\) to a target victim node \(t \in V\setminus (V^{s} \cup V^{\text{new}})\), i.e., to suggest the creation of the directed edge \((t, s)\) between the victim and the attacker. We also assume the attacker has full knowledge of \(f\), i.e., we consider a white-box threat model, where the architecture and the entire set of parameters of \(f (\theta = [\theta_f, \theta_g])\) are known and fixed [39], [40]. Later in Section VI-D, we show that this requirement does not limit the adversary’s power, as our proposed attack can be successfully transferred to disrupt other black-box link prediction models.

B. Attack Framework

The attacker \(s\) aims to maximize the chance that the directed edge between the target victim \(t\) and itself, i.e., \((t, s)\), will be among those predicted by \(f\) while minimizing the “malicious effort” needed. More formally, suppose \(f\) is defined as in (2). We will omit the parameters \(\theta_f\) and \(\theta_g\) from the equations in the following to ease the notation. Also, let \(A, A', \tilde{A}\) be the adjacency matrices associated with the original graph \(G = (V, E)\), the intermediate graph \(G' = (V', E)\), and the final perturbed graph \(\tilde{G} = (V', E')\), respectively. Thus, the problem for the attacker \(s\) is to find the optimal perturbation \(\tilde{h}_{s} \neq h_s\) such that \(f(h_s, \tilde{h}_{s}) \neq f(h_s, h_s)\) by solving the objective as follows:

\[
\tilde{h}_{s} = \arg \max_{\tilde{h}_{s}} \left\{ f(h_s, \tilde{h}_{s}) - f(h_s, h_s) - d(\tilde{h}_{s}, h_s) \right\} \quad (3)
\]

where \(f(h_s, \tilde{h}_{s}) - f(h_s, h_s)\) measures the difference between the original and the adversarial link prediction, and \(d\) captures the magnitude of the malicious effort used by the attacker for transforming \(h_s\) to \(\tilde{h}_{s}\). The generic perturbation \(\tilde{h}_{s} = g(\tilde{A}_{s}, X_{s})\) indicates the attacker node embedding output by a trained GNN \(g\), after the one-hop neighborhood of the nodes controlled by the attacker \(V^{s} \cup V^{\text{new}}\) has been modified according to its capabilities (i.e., by adding/removing edges as specified by \(E^{+}\) and \(E^{-}\), respectively). Specifically, we assume there exists a function \(\pi\) that works as follows:

\[
\tilde{h}_{s} = g(\pi(A'), X_{s}) = g(\tilde{A}_{s}, X_{s}) \quad (4)
\]

where \(A'\) is the full adjacency matrix associated with the intermediate graph \(G'\), and \(\tilde{A}_{s} = \pi(A')\). Notice that the perturbation applied by \(\pi\) reflects the attacker’s capabilities described in our attack model, i.e., it concerns only the topological structure of the neighborhood of the nodes controlled by the attacker. In contrast, the node feature matrices are left unaltered. We acknowledge that the ability to modify also the node feature matrices is an interesting direction to explore. We plan to investigate this as future work since our framework is general enough to cover that scenario. Moreover, the perturbation induced by \(\pi\) on \(h_s\) must not affect the embedding of the target victim node \(h_t\). Indeed, according to (1), the hidden representation of a node \(u\) produced by a given \(l\)-layer GNN \((h_u)\) is influenced by all the other nodes \(v\) that are in the \(l\)-hop neighborhood of \(u\), i.e., \(N^{l}(u)\). Thus, to avoid that \(h_t\) is affected by \(\pi\), we assume \(t \notin N^{l}(s)\) and, even strongly, \(s \notin N^{l}(t)\); if the opposite was true, the attack might be trivial. To measure the malicious effort of the attacker, the function \(d\) captures the difference between the original and the perturbed attacker node embeddings. Such a difference must consider: i) the number of additional vicious nodes \(V^{\text{new}}\) injected by the attacker to generate the intermediate augmented graph \(G'\); ii) the distance between the original adjacency matrix \(A_s\) and the perturbed \(\tilde{A}_{s}\) obtained from structural modifications (i.e., edge additions/deletions) of \(G'\).

V. METHOD

In this section, we describe the method used by the attacker to solve the optimization problem defined in (3), which we refer to as SAVAGE. We consider the function \(\pi\) parametrized by a perturbation matrix \(P\), i.e., \(\pi(\cdot; P)\), and thus rewrite (4) as follows:

\[
\tilde{h}_{s;P} = g(\pi(A'; P), X_{s}) = g((P + A'), X_{s}) = g(\tilde{A}_{s}, X_{s}) \quad (5)
\]

namely \(\tilde{A}_{s} = \pi(A'; P) = P + A'\). The perturbation matrix \(P \in \{-1, 0, 1\}^{n \times n}\) is a discrete squared matrix, where \(n = |V|\) is the total number of nodes of the augmented graph \(G' = (V', E)\). This includes the set of additional vicious nodes \(V^{\text{new}}\) injected by the attacker into the original graph \(G = (V, E)\) (i.e., \(V' = V \cup V^{\text{new}}\)), yet keeping the same set of original edges \(E\), thus is represented by the adjacency matrix \(A'\). For each \(i \in V^{s} \cup V^{\text{new}}, j \in V'\)

\[
P_{i,j} = \begin{cases} +1 & \text{if } (i, j) \in E^{+} : \text{add the edge between } i \text{ and } j \\ -1 & \text{if } (i, j) \in E^{-} : \text{remove the edge between } i \text{ and } j \\ 0 & \text{otherwise.} \end{cases}
\]

Hence, for a given perturbation matrix \(P\), a graph \(G'\) with \(V^{\text{new}}\) vicious nodes, a pair of source (attacker) and target (victim) nodes \(s, t \in V\), and a fixed GNN-based link prediction model

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
\( f \), such that \((t,s) \notin \mathcal{E} \land f(h_t, h_s) < \tau \), we can compute the following (instance-level) loss function:

\[
\mathcal{L}(P) = \ell_{\text{adv}}[f(h_t, \tilde{h}_s; P)] + \beta \ell_{\text{dist}}(h_s, \tilde{h}_s; P) + \gamma \ell_{\text{new}}(u_{\text{new}}),
\]

(6)

The first component \( \ell_{\text{adv}} \) penalizes the model if the adversarial prediction goal is not satisfied and is computed as

\[
\ell_{\text{adv}}[f(h_t, \tilde{h}_s; P)] = -\log[f(h_t, \tilde{h}_s; P)]
\]

(7)

which corresponds to a standard binary cross-entropy, where the class label to predict is always 1, i.e., we want to enforce the prediction of an adversarial edge between \( t \) and \( s \). So far, existing methods have considered only this first component \( \ell_{\text{adv}} \) and employed a fixed budget parameter \( \Delta \) as an upper bound constraint. We claim and prove experimentally that this approach is wasteful and inefficient in terms of resources required by the attacker to mount the attack successfully. Thus, we incorporate in the loss function two additional factors to limit the malicious effort, as described in the following.

The second component \( \ell_{\text{dist}} \) is an arbitrary distance function that discourages \( \tilde{h}_s; P \) from being too far away from \( h_s \), namely \( \tilde{A}_s \), must be close to the original \( A_s \). For example, \( \ell_{\text{dist}}(h_s, \tilde{h}_s; P) = ||A_s - \tilde{A}_s||_p \), where \( || \cdot ||_p \) is the \( L_p \)-norm. The third component \( \ell_{\text{new}} \) controls the number of vicious nodes injected into the original graph network. In practice, \( u_{\text{new}} \) is a \( |\mathcal{V}| \)-dimensional binary vector (i.e., a mask) indicating the vicious nodes injected by the attacker, i.e., \( u_{\text{new}}[u] = 1 \) iff node \( u \in \mathcal{V}_{\text{new}} \) has been added to the graph network, 0 otherwise. For example, \( \ell_{\text{new}}(u_{\text{new}}) = ||u_{\text{new}}||_p \)

In this work, we set \( p = 1 \) both for \( \ell_{\text{dist}} \) and \( \ell_{\text{new}} \). Eventually, solving the objective defined in (3) is framed into the following optimization task:

\[
\tilde{h}_s; P = \arg\min_P \mathcal{L}(P).
\]

(8)

It is worth noticing that, in this original formulation, the optimization objective defined in (8) is inherently discrete. To make it smooth, inspired by prior work [27, 41], we first consider an intermediate, real-valued perturbation matrix \( \tilde{P} \) with entries in \((-1, 1)\), obtained by applying a hyperbolic tangent transformation (tanh). Intuitively, this matrix indicates the degree of confidence of the attacker to add or remove an edge from the adjacency matrix \( A' \). Thus, we can replace \( P \) with \( \tilde{P} \) in (8) and solve the objective via standard gradient-based optimization methods, such as stochastic gradient descent or similar. Eventually, to obtain the discrete matrix \( P \), we simply threshold the entries of \( \tilde{P} \) as follows:

\[
P_{t,j} = \begin{cases} 
+1 & \text{if } \tilde{P}_{t,j} \geq t^+ \\
-1 & \text{if } \tilde{P}_{t,j} \leq t^- \\
0 & \text{otherwise.} 
\end{cases}
\]

A straightforward choice for the abovementioned thresholds is: \( t^+ = 0.5 \) and \( t^- = -0.5 \). Notice, though, that \( P + A' \) would lead to a discrete matrix \( \tilde{A}_s \) whose entries are in the set \{\(-1, 0, 1, 2\)\}, instead of \{\(0, 1\)\} as required. Indeed, removing an edge that does not exist from \( \mathcal{E} \) will lead to an entry equal to \(-1\); on the other hand, adding an edge that already exists in \( \mathcal{E} \) will result in an entry equal to 2. We, therefore, obtain the correct final perturbed matrix \( \tilde{A}_s \) by applying a clamp\([0,1]\) function elementwise, i.e., \( \tilde{A}_s[i,j] = \text{clamp}_{[0,1]}(P_{t,j} + A'_{t,j}) \forall (i,j) \), where \( \text{clamp}_{[0,1]}(x) = \max(0, \min(x, 1)) \).

VI. EXPERIMENTS

We evaluate the effectiveness of SAVAGE on several challenges. First, we assess the power of our attack model by reporting some key performance metrics. Second, we analyze the impact of the critical components of our model through ablation studies. Finally, we demonstrate the ability of SAVAGE to generate attacks that transfer toward other black-box link prediction systems.

A. Experimental Setting

a) Datasets: We validate our method both on real-world and synthetic datasets. Moreover, as our attack model dictates, we focus only on data representing directed graphs. Our real-world datasets are mainly of two kinds: citation-based and social networks. The former category includes Cora [42, 43], Arxiv [44], and Citation2 [44], whereas the latter contains Twitter [45], Wikipedia-Vote [46, 47], and GPlus [45]. It is worth noticing that we test our results on two datasets from the open graph benchmark (OGB) project [48], namely Arxiv and Citation2. The OGB community is devoted to defining a new set of standard graph benchmark that are large in scale, robust, and easily reproducible. These datasets are meant as an evolution over previous standard benchmarks, which presented several issues, such as licensing and sensibility to splits, among others. While we advocate for using OGB as it favors fairer research practices, we still include Cora due to its popularity, stressing the importance of robust datasets in graph learning. Furthermore, we test our method on a synthetic dataset, partially taken from [49], that we call synthetic. To test our method extensively, it is worth noting that we choose datasets featuring several degrees of cardinality and density.

b) GNN Models: To evaluate the effectiveness of SAVAGE, we first train a GNN-based link prediction system. The model comprises two stacked convolutional layers of 128 and 64 hidden units, respectively, and a two-layer MLP with a sigmoid activation function at the last output layer. In our setup, we minimize the prediction error with a binary cross-entropy loss using Adam optimizer with \( 3 \times 10^{-3} \) learning rate. Finally, we compute the accuracy and the area under the ROC (AUROC) curve of the learned GNN-based link predictor. We set \( \tau = 0.6 \) as the classification threshold—i.e., the probability value above which a link is created—as this maximizes the AUROC score. On average, the best model achieves 85% accuracy and 0.86 AUROC score on all datasets considered.

c) Methods: We test our method against different competitors and under several settings. As we open a new research direction, finding relevant competitors to compare our method against is not trivial. Nevertheless, we collect suitable baselines and adapt existing techniques to our setting to have methods we can compare SAVAGE against, thus strengthening the experimental evidence of this work. First, we consider a simple random
baseline (RAND). This can add/remove connections in the graph and activate/deactivate vicious nodes with a probability \( p \); the larger \( p \), the stronger the corresponding attacker. We denote by RAND-L and RAND-H the baseline with the lowest and the highest probability of addition/activation, i.e., \( p_L = 0.25 \) and \( p_H = 0.75 \). While we cannot perform a direct comparison with other methods, as they are not naturally designed for our setting, we adapt the greedy technique proposed in [7] and [6] to work in our framework, and we call it adapted iterative gradient attack (AIGA). We also conduct a comparison between our method and optimized graph structure perturbation (OGSP) [33], a technique that greedily considers loss differentials to generate adversarial perturbations for link prediction systems. It is important to note that these competing methods can be categorized into two groups: those that can only restrict the number of resources used but cannot enforce sparsity like we do, as doing so would lead to gradient collapse and task failure, and those that experience performance degradation when attempting to enforce sparsity, as demonstrated in our experiments. Finally, we consider four instances of our method: i) SAVAGE; ii) SAVAGE-N; iii) SAVAGE-I; and iv) SAVAGE-NI. SAVAGE refers to the plain full framework, randomly initialized and defined in (6); SAVAGE-N is the same as SAVAGE yet without the penalty losses (i.e., \( \ell_{\text{dist}} = \ell_{\text{new}} = 0 \)); SAVAGE-I is the same as SAVAGE yet initialized with the output from AIGA; and SAVAGE-NI is the same as SAVAGE-N yet initialized with the output from AIGA. Since SAVAGE tries to solve a highly nonconvex problem, the parameters’ initialization is crucial. Distinguishing between random and AIGA-based initialization serves two purposes. First, it shows that SAVAGE can be used just as a sparsifying framework, compatible with other techniques; second, it demonstrates the impact of initialization on the optimization problem. For all the models considered, performance is reported according to their best hyperparameter setting.

d) Evaluation metrics: Given a graph and a GNN-based link prediction model trained on it, we uniformly sample a source and a target node from the graph, providing that the two nodes are disconnected and, even further, they are not part of each other’s two-hop neighborhood. Moreover, according to the link prediction model, no edge should exist between the two sampled nodes. Then, we attack the graph using all the methods described previously (i.e., the four variants of our SAVAGE along with its competitors) and run the link prediction on the resulting perturbed graph again. We repeat this procedure for 20 different node pairs for each dataset used. We present our results according to the following metrics. The average prediction under attack (AP) is the estimated probability output by the link prediction model after the attack. The attack rate (AR) measures the fraction of successful attacks, i.e., counts the fraction of link predictions that a specific attack brings above the creation threshold \( \tau \) (0.6). AP and AR capture two different aspects of the attack power. The former indicates the average strength of each attack (i.e., the new link probability between the source and target node, which initially was below the success threshold \( \tau \)). The latter measures only how frequently the new link probability is large enough for the attack to succeed, regardless of its actual value. We also report statistics on the perturbation induced by SAVAGE on the original graph topology: the number of vicious nodes injected (AN) and the shift in the node’s degree distribution before and after the attack as measured with the Kullback–Leibler (KL) divergence. Finally, we report on what we name AE; this metric is defined as the ratio of the attack power to the number of vicious nodes employed. We use AE as the leading metric that summarizes the effectiveness of the attack, namely a tradeoff between the power of the attack and the amount of resources used.

B. Attack Power

We measure the power of the attacks generated by all the methods examined on the real-world and synthetic datasets; the results are shown in Table I. We may observe that SAVAGE can mount very effective attacks across all datasets, with the highest AP and AR among all the considered methods when considering all SAVAGE variants. In terms of AR, we notice a positive correlation between the performance gap and the density of the dataset. Furthermore, SAVAGE significantly reduces the number of malicious resources used (AN) w.r.t. to AIGA and OGSP, ranging from a minimum of 20% up to 80% decrease. Also, the KL score illustrates the limited impact on graph connectivity, especially when compared with the baselines (see Section VI-E for additional discussion). We see that the AIGA initialization can further mitigate this impact at the cost of a slightly worse AR in sparser graphs. This may be caused by the initialization that imposes the solution to get stuck around the starting local minimum. In all the settings and datasets we examined, we observed that our SAVAGE variant, which incorporates our penalty losses, consistently outperforms the SAVAGE-N version, which lacks these crafted penalties, in terms of AE. We consider AE to be a crucial metric for evaluating the effectiveness of our approach in tackling the sparse problem addressed in this study. In other words, this analysis shows that SAVAGE: i) generates more successful attacks than competitors and ii) does not rely on the introduction of random noise since random baselines are not able to disrupt the prediction even when using a comparable number of resources.

C. Ablation Study

We perform an extensive ablation study to support the design choices made for the main components of our framework. A key characteristic of SAVAGE is its ability to enforce the sparsity of malicious resources by adjusting the optimization goal of the attacker. We analyze several components of the model, such as the initialization of the perturbation matrix and the effect of the penalty losses, to study their behavior and how this impacts the sparsification effort. We provide a graphic visualization of our findings on SAVAGE inner working. In Fig. 2, we inspect the gradients of our model with our introduced sparsity losses \( \ell_{\text{dist}} \) and \( \ell_{\text{new}} \) (left) against the model not featuring them (right). It is highly evident that most vicious nodes on the left are deactivated (encoded with blue) and do not contribute as additional resources to mount the attack. This is not true for gradients on the right, which have a much higher diffusion, with
TABLE I
WE REPORT SEVERAL METRICS REGARDING THE EFFECTIVENESS AND EFFICACY OF SAVAGE

| Method    | AR ↑ | AN ↓ | AP ↑ | KL ↓ | AR ↑ | AN ↓ | AP ↑ | KL ↓ | AE ↑ | AR ↑ | AN ↓ | AP ↑ | KL ↓ | AE ↑ |
|-----------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| RAND-L    | 0.00 | 23.15 | 0.04 | 5.00 | 0.00 | 0.00 | 13.70 | 0.04 | 1.00 | 0.00 | 0.00 | 12.95 | 0.03 | 6.00 | 0.38 |
| RAND-H    | 0.10 | 74.70 | 0.11 | 10.00 | 0.13 | 0.00 | 37.35 | 0.04 | 10.00 | 0.00 | 38.85 | 0.03 | 10.00 | 0.00 |
| AIGA      | 0.60 | 100.00 | 0.61 | 0.00 | 0.60 | 0.20 | 50.00 | 0.29 | 0.00 | 0.40 | 0.90 | 50.00 | 0.93 | 0.03 | 1.80 |
| OGSP      | 0.30 | 50.00 | 0.38 | 0.00 | 0.60 | 0.30 | 50.00 | 0.29 | 0.00 | 0.60 | 0.50 | 50.00 | 0.55 | 0.01 | 1.00 |
| SAVAGE-N  | 0.95 | 95.30 | 0.93 | 4.00 | 0.93 | 0.93 | 46.70 | 0.90 | 1.00 | 1.99 | 0.85 | 37.70 | 0.89 | 4.00 | 2.25 |
| SAVAGE-I  | 1.00 | 95.55 | 0.99 | 4.00 | 0.97 | 0.97 | 50.00 | 0.95 | 0.03 | 1.94 | 1.00 | 47.10 | 0.99 | 3.00 | 2.12 |
| SAVAGE    | 0.75 | 57.95 | 0.73 | 3.00 | 1.29 | 0.80 | 35.70 | 0.79 | 0.03 | 2.24 | 0.85 | 13.10 | 0.87 | 1.00 | 6.18 |
| SAVAGE-I  | 0.75 | 61.05 | 0.70 | 3.00 | 1.22 | 0.97 | 38.70 | 0.95 | 0.03 | 2.50 | 1.00 | 7.05 | 0.98 | 0.30 | 14.18 |

Fig. 2. Visualizations of gradients for the model with $\ell_{\text{dist}}$ and $\ell_{\text{new}}$ (left) and without (right). Each cell describes the activation strength of a vicious node. In blue are deactivated nodes. We (visually) show that our losses successfully induce a sparse usage of malicious resources.

First, we manage to mount very effective link prediction attacks. Second, thanks to our penalized framework, we are able to greatly reduce the amount of vicious nodes required. KL is reported in base $10^{-3}$. The bold values indicate the best performance.

We report several metrics regarding the effectiveness and efficacy of SAVAGE. Its topology and leaving it for future work. Still, we conduct a comprehensive analysis, showing that SAVAGE is robust under different assumptions for creating the vicious nodes’ features.

a) Initialization: As hinted several times earlier, we solve a highly nonconvex optimization problem. For this reason, the initialization of the perturbation matrix $P$ can be considered pivotal, as it reflects the prior information we inject into the model and the model’s results. We run an ablation experiment to show how different initializations lead to different results. In particular, we test four possible initializations for the perturbation matrix: Random, all zeros + $\epsilon$, all ones $-$ $\epsilon$, and all negative ones + $\epsilon$; where $\epsilon$ is a small random positive number ($\epsilon < 0.3$). We test these options by running experiments for 20 different pairs randomly sampled from the synthetic dataset. Results in Table II show that random initialization exhibits the best tradeoff between the attack power and the number of resources used.

b) Penalty losses: SAVAGE relies on two penalty losses ($\ell_{\text{dist}}$ and $\ell_{\text{new}}$) to effectively sparsify the resources used to carry out the attack. Two parameters control the importance of these losses: $\beta$ and $\gamma$, respectively. We, thus, perform a sensitivity

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
analysis on these parameters to quantify the impact of both penalty losses on the model performance. In particular, we sample 20 pairs at random from the synthetic dataset and run our method on them, \( \ell_{\text{dist}}(\beta) \). As stated previously, \( \beta \) regulates the effect of \( \ell_{\text{dist}} \), namely the penalty discouraging \( \bar{h}_{s,P} \) from being too far off from \( h_s \). In Fig. 3, we report our findings. On the horizontal axis, we indicate the value of \( \beta \), while on the vertical axis, the amount of resources used and the AE for the top graph and the bottom graph, respectively. As we can see, increasing \( \beta \) corresponds to a reduction in the malicious resources used by SAVAGE to mount the attack. Furthermore, an increase in \( \beta \) also constitutes an increase in terms of AE up to a saturation point, after which it decreases rapidly. We found a smooth relationship between the increase in \( \beta \) and the reduction in the use of resources. This is desirable for the attacker, as it enables them to find a suitable tradeoff between budget and effectiveness rather easily. As opposed to a scenario in which a change in \( \beta \) would cause wild swings in terms of resources used, making the task of finding a suitable tradeoff unpractical. While not shown here for visualization purposes, the relationships remain constant outside the range considered.

As explained earlier, \( \gamma \) regulates the effect of \( \ell_{\text{new}} \), namely the penalty directly discouraging the proliferation of vicious nodes injected into the original graph network. We report results in Fig. 4. We indicate the value of \( \gamma \) on the horizontal axis, with the amount of resources used on the vertical axis (in log scale) and the AE, respectively, for the top and bottom graphs. Once again, even though we face an intrinsic discrete problem, we show how our formulation features a nice, well-behaved decrease in resources used, along with the increase in the penalty loss, with an almost-monotonic behavior. Similarly, we show that a proper value for \( \gamma \) helps in reaching an optimal AE.

![Fig. 3. Amount of malicious resources used as the penalty loss \( \beta \) changes (left). It can be noticed that the number of resources monotonically decreases as the value of \( \beta \) increases. AE metric as the penalty loss \( \beta \) changes (right). It can be noticed the attack effectiveness increases as the value of \( \beta \) increases, up to a saturation point.](image)

![Fig. 4. Amount of malicious resources used as the penalty loss \( \gamma \) changes (left). It can be noticed that the number of resources monotonically decreases as the value of \( \gamma \) increases. AE metric as the penalty loss \( \gamma \) changes (right). It can be noticed the attack effectiveness increases as the value of \( \gamma \) increases, up to a saturation point.](image)

**Table II**

| Method      | AR | AN | AP | KL |
|-------------|----|----|----|----|
| Random      | 0.90 | 14.00 | 0.89 | 0.01 |
| All zeros   | 0.75 | 11.50 | 0.74 | 0.01 |
| All ones    | 0.45 | 7.20  | 0.50 | 0.01 |
| All negative ones | 0.55 | 22.05 | 0.58 | 0.00 |

Random initialization offers the best trade-off between the resources used and the power of the attack.

**Table III**

| Method      | AR | AN | AP | KL |
|-------------|----|----|----|----|
| Existent    | 0.90 | 13.90 | 0.90 | 0.01 |
| Random      | 1.00 | 16.70 | 0.98 | 0.01 |
| Ones        | 0.75 | 42.50 | 0.75 | 0.01 |
| Zeros       | 0.95 | 14.10 | 0.91 | 0.01 |
| Mean        | 0.91 | 14.10 | 0.91 | 0.01 |
| Median      | 0.95 | 14.00 | 0.91 | 0.01 |
| While not optimizing for this, SAVAGE is robust under different settings and maintains strong performance. |

**c)** Feature matrix: SAVAGE does not focus on creating the features for the vicious nodes injected into the original graphs, focusing instead on their topology. In particular, we do not optimize for it, leaving it as future work. However, we claim that SAVAGE is robust for creating such a matrix under different assumptions. For this experiment, once again, we sample 20 pairs randomly from the synthetic dataset and run our method on them. This time, however, we consider six different settings for creating the feature matrix. The first setting, used for the main experiment, is to randomly sample features from existing nodes and add a slight noise. Three settings consist of initializing the feature matrix \( X \) with zeros, ones, and at random, respectively; we also apply some noise \( \epsilon \) here. Finally, we take the mean and the median of the feature matrix in two cases, still adding noise to it. Results are reported in Table III. These results show that SAVAGE maintains strong performance across different initialization strategies. Specifically, we may notice how the random, zeros, and median methods produce the strongest attack power, while ones results in the lowest attack power yet requires the highest number of malicious resources.

**d)** Performance across different upper bounds of malicious resources: Conventional methods employing fixed upper bounds on the number of malicious resources often utilize their entire budget to launch attacks. In contrast, SAVAGE adopts a sparsity-enforcing mechanism, integrated with purposefully designed losses, to intelligently minimize the required number of malicious nodes for the attack while outperforming competing methods. To further support this claim, we conducted additional experiments. In particular, we evaluated our main competitor, AIGA, under varying settings of upper resource limits (i.e.,
available malicious nodes). We analyzed the achieved AR and the number of resources utilized, and the results are illustrated in Fig. 5. For each dataset examined in this study, we depict the competitor’s AR with a blue line on the y-axis, representing the AR achieved, and the x-axis denotes the number of malicious resources used. We overlay our system’s performance on the same graph. The green dotted line represents the AR achieved by our proposed system, SAVAGE, while the red line shows the number of resources used to attain that result. This analysis yields two main conclusions. First, the AR of AIGA is consistently below that of SAVAGE when using the same number of malicious nodes. Second, even more remarkably, AIGA’s AR remains lower than SAVAGE’s, even when AIGA has access to a larger pool of malicious resources to exploit. Interestingly, in some datasets, AIGA’s performance decreases as the number of malicious resources increases, which might appear counter-intuitive. This phenomenon, particularly evident in the Cora dataset, suggests that AIGA tends to favor individual malicious nodes over cooperative strategies. These results provide further evidence supporting the claim of SAVAGE’s superiority in efficiently launching successful attacks, strategically leveraging available malicious nodes.

**D. Attack Transferability**

SAVAGE defines a white-box attack model, i.e., it assumes full knowledge of the target GNN-based link prediction model. This may be unrealistic in practice. First, the internals of a GNN-based link prediction system (e.g., the model’s architecture, parameters, or gradients) are rarely disclosed, especially in a commercial scenario. Second, other link prediction models, not necessarily GNN-based, may exist. Still, we claim that the abovementioned two considerations do not limit the feasibility of our method. In fact, we show that the adversarial perturbations generated by SAVAGE on a white-box GNN-based surrogate link prediction model can be successfully transferred to fool other black-box target link prediction systems. Specifically, to demonstrate this capability, we perform an experiment similar to [33] and consider the target link prediction heuristics reported in Table IV. The higher the value of these heuristics, the higher the chance of a link between two nodes. For a given graph, a source and a target node initially disconnected, and a white-box GNN-based surrogate link predictor, we generate the corresponding graph perturbation with SAVAGE to increase the probability of a connection between the victim and the source. The scores in Table IV are expressed in terms of the lift of the heuristic values as computed in the original, unperturbed graph, i.e., before and after the attack, averaged across 20 different pairs of sampled source and target nodes. We notice that all the heuristic values increase, indicating that SAVAGE generalizes and can mount attacks even on other, not necessarily GNN-based, link prediction systems. Moreover, these results highlight that SAVAGE implements attacks that genuinely affect the core principles underlying the link prediction phenomenon rather than the nuances of a specific model. Indeed, each heuristic

| Heuristic                   | Lift score (attack/non-attack) |
|-----------------------------|--------------------------------|
| Common neighbors [13]       | 70 x                          |
| Jaccard [12]                | 93 x                          |
| Preference attachment [14]  | 3.4 x                         |
| Adar-Adar [50]              | 61 x                          |
| Resource allocation [16]    | 2.9 x                         |
| Katz index [17]             | 3e5 x                         |
| PageRank [18]               | 1.2 x                         |
| SimRank [19]                | 20 x                          |

Given the large increase in heuristic values after the attack, we claim that our attacks successfully transfer to different black-box methods.

---

*Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.*
E. Limitations

Experiments have demonstrated that SAVAGE can achieve a high attack success rate while using a sparse number of new, malicious nodes. However, SAVAGE causes a shift in the node’s degree distribution between the original and the attacked graph larger than other methods, thus indicating a more substantial perturbation of the initial graph links. We argue that, in general, from the attacker’s perspective, the cost of adding a vicious node to the network is the most relevant and that should receive the highest priority. Moreover, on average, the KL-divergence between the two node distributions is smaller than 10^{-3} with SAVAGE. Still, we acknowledge this as a limitation of our current method, at least compared to the other methods considered in this work. We plan to incorporate a new factor (based on the abovementioned KL-divergence) in the attacker’s objective, limiting the impact of SAVAGE on the original graph connections.

VII. CONCLUSION

In this work, we presented SAVAGE, a novel framework for generating adversarial attacks on GNN-based link prediction systems. SAVAGE defines a white-box attack model where an adversary aims to appear in the list of recommended users to follow for a target victim node. The attacker controls a subset of original nodes and some additional vicious nodes that can inject into the network. Each of these malicious nodes can, in turn, create or destroy direct edges with others until eventually altering the link prediction between the target victim node and the attacker node. SAVAGE formulated the problem that the attacker must solve as an optimization task, which trades off between the attack’s success and the sparsity of malicious resources required. Experiments conducted on real-world and synthetic datasets showed the effectiveness and efficiency of our approach in attacking several GNN-based link prediction models. Moreover, we showed that SAVAGE attacks could be successfully transferred to disrupt other link prediction systems (not necessarily GNN-based) under the black-box setting. In future work, we plan to extend our attack model with the ability to perturb node features. Furthermore, our method must operate on dense adjacency matrices, somewhat limiting its scalability. Thus, we will develop a variant of SAVAGE that can work with sparse adjacency matrices. The design of possible defense mechanisms to combat the attacks generated by SAVAGE is another interesting direction to explore. Finally, we will also consider different downstream tasks as the target of the attacks (e.g., node classification).

REFERENCES

[1] V. Martinez, F. Berzal, and J.-C. Cubero, “A survey of link prediction in complex networks,” ACM Comput. Surv., vol. 49, no. 4, pp. 1–33, 2016, doi: 10.1145/3012704.

[2] A. Kumar, S. S. Singh, K. Singh, and B. Biswas, “Link prediction techniques, applications, and performance: A survey,” Phys. A: Stat. Mech. Appl., vol. 553, 2020, Art. no. 124289.

[3] L. Lü and T. Zhou, “Link prediction in complex networks: A survey,” Phys. A: Stat. Mech. Appl., vol. 390, no. 6, pp. 1150–1170, 2011.

[4] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. Int. Conf. Learn. Representations, 2017. [Online]. Available: https://openreview.net/forum?id=SU4ayYgl

[5] L. Sun et al., “Adversarial attack and defense on graph data: A survey,” IEEE Trans. Knowl. Data Eng., vol. 35, no. 8, pp. 7693–7711, 2023, doi: 10.1109/TKDE.2022.3201243.

[6] J. Chen, X. Lin, Z. Shi, and Y. Liu, “Link prediction adversarial attack via iterative gradient attack,” IEEE Trans. Computat. Social. Syst., vol. 7, no. 4, pp. 1081–1094, Aug. 2020.

[7] X. Wang, M. Cheng, J. Eaton, C.-J. Hsieh, and F. Wu, “Attack graph convolutional networks by adding fake nodes,” 2018, arXiv:1810.10751.

[8] T. B. Brown et al., “Language models are few-shot learners,” in Proc. 34th Int. Conf. Adv. Neural Inf. Process. Syst., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/1457cd0bdfcb496741bfb8ac142ff6a-Abstract.html

[9] G. Mialon et al., “Augmented language models: A survey,” 2023, arXiv:2302.07842.

[10] G. Trappolini, A. Santilli, E. Rodolà, A. Y. Halevy, and F. Silvestri, “Multimodal neural databases,” in Proc. 46th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2023, pp. 2619–2628, doi: 10.1145/3539618.3539190.

[11] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, “A comprehensive survey on graph neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 1, pp. 4–24, Jan. 2021.

[12] D. Liben-Nowell and J. Kleinberg, “The link prediction problem for social networks,” in Proc. 25th Int. Conf. Inf. Knowl. Manage., 2003, pp. 556–559, doi: 10.1145/956683.956972.

[13] M. E. Newman, “Clustering and preferential attachment in growing networks,” Phys. Rev. E, vol. 64, no. 2, 2001, Art. no. 026102.

[14] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” Science, vol. 286, no. 5439, pp. 509–512, 1999.

[15] L. A. Adamic and E. Adar, “Friends and neighbors on the web,” Social Netw., vol. 25, no. 3, pp. 211–230, 2003, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378873303000091

[16] T. Zhou, L. Lü, and Y.-C. Zhang, “Predicting missing links via local information,” Eur. Phys. J. B, vol. 71, no. 4, pp. 623–630, 2009.

[17] L. Katz, “A new status index derived from sociometric analysis,” Psychometrika, vol. 18, no. 1, pp. 39–43, 1953.

[18] S. Brin and L. Page, “The anatomy of a large-scale hypertextual web search engine,” Comput. Netw. ISDN Syst., vol. 30, no. 1, pp. 107–117, 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016975529800110X

[19] G. Jeh and J. Widom, “SimRank: A measure of structural-context similarity,” in Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2002, pp. 538–543, doi: 10.1145/775047.775126.

[20] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki, “Link prediction using supervised learning,” in Proc. SDM: Workshop Link Anal., Counter-Terrorism, Secur., 2006, vol. 30, pp. 798–805.

[21] M. Zhang and Y. Chen, “Link prediction based on graph neural networks,” in Proc. Int. Conf. Adv. Neural Inf. Process. Syst., 2018, pp. 5171–5181. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/53f0d7c537d99b3824f0f99d62ea2428-Abstract.html

[22] B. Li, Y. Xia, S. Xu, L. Wu, and T. Qin, “Distance-enhanced graph neural network for link prediction,” in Proc. Int. Conf. Mach. Learn.: Workshop Comput. Biol., 2021.

[23] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in Proc. Int. Conf. Learn. Representations, 2015, arXiv:1412.6572.

[24] K. Grosse, N. Papenot, P. Manoharan, M. Backes, and P. McDaniel, “Adversarial examples for malware detection,” in Proc. Eur. Symp. Res. Comput. Secur., 2017, pp. 77–93.

[25] G. Tolomei, F. Silvestri, A. Haines, and M. Lalmas, “Interpretable predictions of tree-based ensembles via actionable feature tweaking,” in Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2017, pp. 465–474.

[26] G. Tolomei and F. Silvestri, “Generating actionable interpretations from ensembles of decision trees,” IEEE Trans. Knowl. Data Eng., vol. 33, no. 4, pp. 1540–1553, Apr. 2021.

[27] A. Lucic, M. A. Ter Hoeve, G. Tolomei, M. De Rijke, and F. Silvestri, “CF-GNNEXplainer: Counterfactual explanations for graph neural networks,” in Proc. Int. Conf. Artif. Intell. Statist., 2022, pp. 4499–4511.

[28] Z. Chen, F. Silvestri, J. Wang, H. Zhu, H. Ahn, and G. Tolomei, “ReLAX: Reinforcement learning agent explainer for arbitrary predictive models,” in Proc. 31st ACM Int. Conf. Inf. Knowl. Manage., 2022, pp. 252–261.

[29] Z. Chen, F. Silvestri, G. Tolomei, J. Wang, H. Zhu, and H. Ahn, “Explain the explainer: Interpreting model-agnostic counterfactual explanations of a deep reinforcement learning agent,” IEEE Trans. Artif. Intell., 2022, to be published, doi: 10.1109/TAI.2022.3223892.
[30] Z. Chen et al., “GREASE: Generate factual and counterfactual explanations for GNN-based recommendations,” 2022, arXiv:2208.04222.

[31] Z. Chen, F. Silvestri, J. Wang, Y. Zhang, and G. Tolomei, “The dark side of explanations: Poisoning recommender systems with counterfactual examples,” in Proc. 46th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2023, pp. 2426–2430.

[32] D. Zügner, A. Akbarnejad, and S. Günnemann, “Adversarial attacks on neural networks for graph data,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Discov. Data Mining, 2018, pp. 2847–2856, doi: 10.1145/3219819.3220078.

[33] W. Lin, S. Ji, and B. Li, “Adversarial attacks on link prediction algorithms based on graph neural networks,” in Proc. 15th ACM Asia Conf. Comput. Commun. Secur., 2020, pp. 370–380.

[34] J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan, “Fast gradient attack on network embedding,” 2018, arXiv:1809.02797.

[35] E. Dai et al., “A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability,” 2022, arXiv:2204.08570.

[36] J. Wang, M. Luo, F. Suya, J. Li, Z. Yang, and Q. Zheng, “Scalable attack on graph data by injecting vicious nodes,” Data Mining Knowl. Discov., vol. 34, no. 5, pp. 1363–1389, 2020.

[37] J. Dai, W. Zhu, and X. Luo, “A targeted universal attack on graph convolutional network by using fake nodes,” Neural Process. Lett., vol. 54, pp. 3321–3337, 2022.

[38] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar, “Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach,” in Proc. Web Conf., 2020, pp. 673–683.

[39] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, “A taxonomy and survey of attacks against machine learning,” Comput. Sci. Rev., vol. 34, 2019, Art. no. 100199 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013718303289.

[40] K. Ren, T. Zheng, Z. Qin, and X. Liu, “Adversarial attacks and defenses in deep learning,” Eng., vol. 6, no. 3, pp. 346–360, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S209509911930503X.

[41] S. Srinivas, A. Subramanya, and R. V. Babu, “Training sparse neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, 2017, pp. 455–462.

[42] J. Kunegis, “KONECT: The Koblenz network collection,” in Proc. Int. Conf. World Wide Web, 2013, pp. 1343–1350.

[43] L. Šubelj and M. Bajec, “Model of complex networks based on citation dynamics,” in Proc. Int. Conf. World Wide Web, 2013, pp. 527–530.

[44] Z. Chen et al., “GREASE: Generate factual and counterfactual explanations for GNN-based recommendations,” 2022, arXiv:2208.04222.

[45] J. J. McAuley and J. Leskovec, “Learning to discover social circles in ego networks,” in Proc. Int. Conf. Adv. Neural Inf. Process. Syst., 2012, pp. 548–556. [Online]. Available: https://proceedings.neurips.cc/paper/2012/hash/7a6414fd86ec325f99f160b08096b3e6d-Abstract.html.

[46] J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg, “Signed networks in social media,” in Proc. SIGCHI Conf. Hum. Factors Comput. Syst., 2010, pp. 1361–1370, doi: 10.1145/1753326.1753532.

[47] J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg, “Predicting positive and negative links in online social networks,” in Proc. Int. Conf. World Wide Web, 2010, pp. 641–650, doi: 10.1145/1772609.1772756.

[48] W. Hu et al., “Open graph benchmark: Datasets for machine learning on graphs,” in Proc. Int. Conf. Adv. Neural Inf. Process. Syst., 2020, pp. 22118–22133. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/fb60d411ac5c5b72bce7d43527cfc844d0-Abstract.html.

[49] M. Fey and J. E. Lenssen, “Fast graph representation learning with PyTorch geometric,” 2019, arXiv:1903.02428.

[50] L. A. Adamic and E. Adar, “Friends and neighbors on the web,” Soc. Netw., vol. 25, no. 3, pp. 211–230, 2003.

Giovanni Trappolini received the Ph.D. degree in machine learning from the Sapienza University of Rome, Rome, Italy, in 2022. He is a Postdoctoral Researcher with the Department of Computer Engineering, Sapienza University of Rome, Rome, Italy. His research focuses on the intersection of geometric and multimodal deep learning, with a particular focus on graph learning.

Dr. Trappolini is a Member of the RSTLess Research Group led by Professor Fabrizio Silvestri, and a recipient of the title of Sapienza’s honor student.

Valentino Maiorca is currently working toward the Ph.D. degree in computer science with the Sapienza University of Rome, Rome, Italy.

He has a background in natural language processing, information retrieval, and geometric deep learning. His research focuses on representation learning, with a strong focus on investigating the geometrical properties of the latent spaces of artificial neural networks.

Silvio Severino received the M.Sc. degree in computer science from the Sapienza University of Rome, Rome, Italy, in 2021, where he is currently working toward the Ph.D. degree in computer science.

His research interests include natural language processing and geometric deep learning, with a particular emphasis on machine translation, multilingualism, and generative models.

Dr. Severino is a Member of the Gladia research group, led by Professor Emanuele Rodolà.

Emanuele Rodolà received the Ph.D. degree in computer science from the Ca’ Foscari University of Venice, Venice, Italy, in 2012.

He is a Full Professor of computer science with the Sapienza University of Rome, Rome, Italy, an ELLIS Fellow, and the Director of the Ph.D. in computer science. He leads the GLADIA group of Geometry, Learning and Applied AI, funded by an ERC Starting Grant and a Google Research Award. He is one of the pioneers of geometric deep learning and has coauthored some of the most influential works in the field. His research focuses on exploiting different representations to establish analogies in the geometry and semantics of different objects.

Fabrizio Silvestri received the Ph.D. degree in computer science from the University of Pisa, Pisa, Italy, in 2004.

He is a Full Professor with DIAG, Sapienza University of Rome, Rome, Italy. He has authored more than 150 papers in international journals and conference proceedings. It holds nine industrial patents. At Facebook AI, he has directed research groups to develop artificial intelligence techniques to combat malicious actors who use the Facebook platform for malicious purposes (hate speech, misinformation, terrorism, and so on). His research interests include artificial intelligence, machine learning applied to web search problems and natural language processing.

Gabriele Tolomei received the M.Sc. degree from the University of Pisa, Pisa, Italy, in 2005, and the Ph.D. degree from the Ca’ Foscari University of Venice, Venice, Italy, in 2011, both in computer science.

He is an Associate Professor with the Department of Computer Science, Sapienza University of Rome, Rome, Italy. He was a Research Scientist with Yahoo Labs, London, U.K., and an Assistant Professor with the Department of Mathematics, University of Padua, Padua, Italy. He has authored more than 40 papers in peer-reviewed international conferences and journals, and he is the inventor of four U.S. patents. His research interests include (human-) explainable, robust, and collaborative machine learning.