Graph Poisoning for Node Rank Manipulation

Seyed Mohammad Hosseini
Toronto Metropolitan University
Toronto, Ontario, Canada
m1hosseini@torontomu.ca

Radin Hamidi Rad
Mila – Quebec AI Institute
Montreal, Quebec, Canada
radin.hamidi-rad@mila.quebec

Morteza Zihayat
Toronto Metropolitan University
Toronto, Ontario, Canada
mzihayat@torontomu.ca

Ebrahim Bagheri
University of Toronto
Toronto, Ontario, Canada
ebrahim.bagheri@utoronto.ca

Abstract

Graph-based retrieval systems rely heavily on structural dependencies, making them vulnerable to adversarial manipulation. We present a black-box graph poisoning attack that degrades a target node's ranking using only edge deletions, without access to model parameters, gradients, or retraining. Prior heuristic methods treat edge influence as a static property and fail to capture how an edge's impact varies with local neighborhood structure. We address this limitation by modeling edge influence as context-dependent. Our method samples multiple ego-networks around a target node, measures empirical utility changes from edge ablations, and trains a local scorer to predict context-sensitive edge effects. At inference, predictions are aggregated across sampled subgraphs to yield stable deletion decisions. Experiments on standard benchmarks show that this approach consistently outperforms existing black-box and white-box baselines across perturbation budgets while remaining model-agnostic.

1 Introduction

Information retrieval systems increasingly rely on graph structures to encode relationships among queries, documents, users, and entities [8]. Examples include bipartite query–document graphs that capture behavioral signals [3], nearest-neighbor graphs used for candidate generation and reranking in embedding-based retrieval, and citation or knowledge graphs that propagate semantic relations [2]. In these settings, relevance is determined through path-based propagation or message passing, making ranking outcomes sensitive to even small structural modifications in the underlying graph.

Given the sensitivity of these approaches to graph structure, *graph poisoning* has received growing attention by the community. The objective is to decrease a chosen target score in ranking or to induce a misclassification for a specific node while applying only a small number of edge deletions in the graph. Existing work can be broadly classified into four directions. *White-box* approaches [4] rely on gradients or internal losses to evaluate perturbations, which ties the attack to architectures and training signals that may not necessarily be accessible to the attacker. *Gray-box* methods [10] assume partial access to the model, such as predictions or linear approximations, but still depend on some internal knowledge to guide perturbations. *Black-box* methods [9] estimate influence through repeated interactions or reinforcement learning and can be effective, yet require large query budgets and often retraining. *Heuristic structural* methods score candidates with single snapshot statistics such as degree, centrality, or spectral measures. These methods are efficient, yet they assume that edge impacts are uniform on all graph nodes, whereas the impact of the same edge can vary depending on the target node and its neighborhood, making static global scores a weaker proxy for graph poisoning attacks.

Our work in this paper adopts a *black-box* setting in which the attacker does not have access to the model architecture, parameters, or gradients, and does not require model retraining. The attack acts only on the input graph by perturbing its edges in order to introduce downstream effects. In this setting, the effect of removing an edge depends on the neighborhood that the scorer actually uses at evaluation time, and that neighborhood can vary as indexing and approximate nearest-neighbor search change which nodes appear nearby. We treat edge influence as *conditional* on local structure. For each target, we sample budgeted k-hop ego networks that represent plausible local views, ablate candidate edges inside each view, and record the resulting change in the downstream objective of node rank demotion. Using these empirical labels, we train a scoring function on features extracted within the subgraphs to predict per-edge sensitivity. At inference, we evaluate candidate edges across a small set of sampled ego networks, aggregate the predicted effects to obtain a stable estimate, and delete the highest-scoring edges within the perturbation budget. This design respects the *black-box* constraints while aligning the attack with neighborhood variability.

In theory, our proposed conditional subgraph based estimation yields two benefits relative to prior poisoning strategies, namely (1) by treating edge influence as an *expectation* over plausible local neighborhoods rather than a fixed quantity tied to a single snapshot, it matches path based propagation of information used in graph representation methods and provides lower variance and greater *stability* compared to heuristic methods that assume invariant edge importance; and (2) by supervising on empirical ablations within sampled ego networks, it captures the same directional signal that gradient driven white box methods target, namely how small structural edits change scores, yet it does so without access to model parameters or losses, which keeps the method *model agnostic*.

The main contributions of our work can be enumerated as follows:

- A *black-box* framework for graph poisoning through edge removal that targets node ranking in retrieval settings without access to model internals or retraining;
- A subgraph conditioned training procedure that learns edge sensitivity from multiple ego network realizations using labels obtained by direct ablation within each subgraph;
- An inference strategy that aggregates predictions across sampled neighborhoods to produce stable demotion estimates under neighborhood variability;

- Experimental study on standard node retrieval benchmarks that compares against heuristic and learning based baselines across perturbation budgets, showing consistent improvements without gradients or retraining.

2 Methodology

In this section, we investigate *graph structure poisoning attacks*, where an adversary is capable of *removing edges* in the original graph under a limited modification budget.

2.1 Problem Definition

Let $G = (V, E)$ be a graph with adjacency $\mathbf{A}$ and node features $\mathbf{X}$. For retrieval, query $\mathbf{q}$ is scored against a target node v_t by a similarity $s(\mathbf{q}, v_t)$; we write per-target utility as:

$$J(G; v_t) = \mathbb{E}_{\mathbf{q}\sim Q(v_t)}\left[s(\mathbf{q}, v_t)\right],$$

where $Q(v_t)$ denotes the query distribution for v_t. The utility is the (positive) margin $J(G; v_t) = m(v_t)$. An attack chooses a set of incident edges $\mathcal{B} \subseteq \mathcal{N}(v_t)$, $\mathcal{N}(v_t) \triangleq \{(v_t, u) \in E\}$ (the set of edges incident to v_t) with $|\mathcal{B}| \le B$ and deletes them, producing $G' = (V, E \setminus \mathcal{B})$. The setting is *black-box* and hence the attacker observes utilities J but has no access to model internals.

2.2 Proposed Method

To relate structural edits to representation changes, we analyze a linearized k-layer propagation model $f_{\text{GNN}}(\mathbf{A}, \mathbf{X}) = \mathbf{A}^k\mathbf{X}\mathbf{W}$. Let ":" denote the Frobenius inner product, $\mathbf{U} : \mathbf{V} = \operatorname{tr}(\mathbf{U}^\top\mathbf{V})$. Deleting a single edge induces a perturbation $\Delta\mathbf{A}$ and a first-order change in the embedding of v_t,

$$\Delta\mathbf{z}_{v_t} \approx \frac{\partial \mathbf{z}_{v_t}}{\partial \mathbf{A}} : \Delta\mathbf{A} = \sum_{l=1}^{k} \mathbf{A}^{\,l-1}\,\Delta\mathbf{A}\,\mathbf{A}^{\,k-l}\,\mathbf{X}\mathbf{W},$$

which shows how a local edit propagates along multiple hops through powers of $\mathbf{A}$. This motivates a scalar attack sensitivity that directly measures task impact. For an edge $e = (v_i, v_j)$ and query $\mathbf{q}$, we define

$$\mathcal{S}(v_t, e \mid \mathbf{q}) = \left|\frac{\partial s(\mathbf{q}, v_t)}{\partial A_{ij}}\right|, \qquad \mathcal{S}(v_t, e) = \mathbb{E}_{\mathbf{q}\sim Q(v_t)}\left[\mathcal{S}(v_t, e \mid \mathbf{q})\right],$$

Large $\mathcal{S}(v_t, e)$ indicates that removing e strongly decreases the utility. Closed-form computation of $\mathcal{S}$ is intractable for deep nonlinear models, so we learn a surrogate $f_\theta(\mathbf{h}_e) \approx \mathcal{S}(v_t, e)$ by supervised regression on context-dependent edge descriptors extracted from local subgraphs. Supervision is obtained by empirical ablation on the full graph:

$$\hat{\mathcal{S}}(v_t, e) = \left[J(G; v_t) - J(G \setminus \{e\}; v_t)\right]_+, \qquad [x]_+ = \max(x, 0),$$

We adopt a larger-is-better convention for J (similarity or positive margin), so harmful deletions yield positive labels while helpful or neutral deletions map to zero. To expose structural variation while controlling cost, we generate m localized views around v_t by a stochastic k-hop expansion that samples up to s_h neighbors at hop h without replacement and then prunes to a node budget $S_{\max}$, i.e., retain v_t and its one-hop neighbors; if over budget, sample additional nodes from the remainder with probability proportional to $\deg^\alpha$ for $\alpha \in [0, 1]$, insert shortest-path connectors as needed to keep nodes reachable from v_t, and, if still over budget, drop farthest-by-hop nodes (ties by lower degree). We then denote the resulting views by $\{G_s^{(r)}\}_{r=1}^m$. For a given view G_s, we form an edge representation $\mathbf{h}_e(G_s) \in \mathbb{R}^d$ only when $e \in E(G_s)$. The scorer $f_\theta : \mathbb{R}^d \to \mathbb{R}_{\ge 0}$ is trained to predict the *same* label $\hat{\mathcal{S}}(v_t, e)$ across *all* realizations in which the edge appears, using the mean-squared error objective

$$\min_\theta\ \mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|}\sum_{(v_t, e, G_s)\in\mathcal{D}} \Big(f_\theta\big(\mathbf{h}_e(G_s)\big) - \hat{\mathcal{S}}(v_t, e)\Big)^2,$$

$$\mathcal{D} = \bigcup_{v_t}\Big\{(v_t, e, G_s) : e \in \mathcal{N}(v_t) \cap E(G_s),\ G_s \in \{G_s^{(r)}\}\Big\}.$$

2.3 Inference

At inference time, we mirror training to stabilize decisions under neighborhood variability. Around v_t, we sample $\{G_s^{(r)}\}_{r=1}^m$, score each candidate edge in each view, and aggregate the predictions to estimate the expected effect under the subgraph distribution,

$$\widetilde{\mathcal{S}}(v_t, e) = \frac{1}{m_e}\sum_{r:\, e\in E(G_s^{(r)})} f_\theta\big(\mathbf{h}_e(G_s^{(r)})\big), \qquad m_e = \big|\{\, r : e \in E(G_s^{(r)})\,\}\big|.$$

$$\widetilde{\mathcal{S}}(v_t, e) \approx \mathbb{E}_{G_s\sim\mathcal{P}(\cdot\mid v_t)}\left[f_\theta\big(\mathbf{h}_e(G_s)\big)\right].$$

Given a budget B, we select

$$\mathcal{B}^\star(v_t) = \operatorname*{arg\,max}_{\mathcal{B}\subseteq\mathcal{N}(v_t),\ |\mathcal{B}|\le B}\ \sum_{e\in\mathcal{B}} \widetilde{\mathcal{S}}(v_t, e),$$

The aggregated score $\widetilde{\mathcal{S}}$ provides low-variance estimates of edge impact across neighborhood realizations.

2.4 Time Complexity Analysis

Let d_t denote the degree of a target node v_t, R the number of sampled ego-networks, S the maximum subgraph size, C_{oracle} the cost of one utility evaluation $J(\cdot)$, and C_f the cost of a forward pass of the scorer.
Training. For each target, we compute $J(G; v_t)$ once and $J(G\setminus\{e\}; v_t)$ for every incident edge e, giving a labeling cost of $O(d_t\, C_{\text{oracle}})$. Sampling R subgraphs of size at most S costs $O(RS)$. Each edge appears in up to R subgraphs, yielding $O(Rd_t)$ training examples and a scorer training cost of $O(Rd_t\, C_f)$. Overall training time per target is: $O\big(d_t\, C_{\text{oracle}} + RS + Rd_t\, C_f\big)$.
Inference. Attack inference mirrors training: sampling R ego-networks costs $O(RS)$, and scoring all incident edges in all views costs $O(Rd_t\, C_f)$. Selecting the top-B edges requires $O(d_t)$. Total inference time per target is $O\big(RS + Rd_t\, C_f + d_t\big)$.

The beauty of our approach is that both training and inference scale linearly with the target node's degree and with the number of sampled ego-networks, and do not depend on the global graph size beyond these local quantities. As we will see in the experiments, this is in contrast to some of the state of the art baselines [9] that fail to produce any results and reach a timeout.

3 Experiments

Research Questions. To assess the effectiveness and robustness of our proposed black-box graph poisoning method, we organize our empirical evaluation around the following research questions:

RQ1: How does our proposed method perform on the retrieval demotion task compared to existing baselines? This question examines whether our attack can degrade model utility more effectively than both heuristic and learning-based baselines, despite operating in a fully black-box setting.

RQ2: How scalable is our approach compared to other black-box methods under increasing graph size? Here, we investigate the computational efficiency of our method in comparison to other black-box techniques, particularly NAG-R, by analyzing time and space complexity on large graphs and identifying performance bottlenecks.

RQ3: How does adversarial effectiveness change with increasing perturbation budgets? This question examines the robustness of our edge selection strategy under varying budgets. We assess whether the model maintains or improves its demotion impact as the number of allowable deletions increases, and how this compares to baseline trends.

3.1 Evaluation Setup

Datasets. We evaluate our attack on three standard network benchmarks: `CoraFull` [2], `Amazon-Photo` [8], and `PubMed` [7]. These datasets vary in size, feature dimensionality, and number of classes, providing a diverse evaluation setting. Detailed statistics for each dataset are summarized in Table 1.

Implementation Details. Experiments were conducted using an NVIDIA RTX 6000 ADA GPU. To ensure fair comparisons, we use the same budget constraints and evaluation protocols across all baselines. For methods that required more than 4 days of computation to complete attack generation for the specified targets, we marked them as *OOT* (Out of Time). Our implementation of the proposed attack method, including, baseline integration, and analysis tools, is publicly available at:
https://github.com/m-hoseyny/graph-poisoning-proposed-model.

Baselines. To evaluate the effectiveness of our proposed attack, we compare it against a diverse set of baselines spanning heuristic, unsupervised, and supervised methods, covering white-box, gray-box and black-box scenarios:

- *Degree*: A simple structural heuristic that removes edges connected to low-degree nodes in the neighborhood. The intuition is that nodes with lower degree typically exert more influence, and removing them may degrade the target node's prominence.
- *PageRank* [5]: This method ranks neighbors using personalized PageRank scores and removes edges to the most influential nodes. It approximates long-range influence propagation and targets nodes with broader graph centrality.
- *Random*: A naive baseline that deletes edges uniformly at random. It serves as a control to assess whether structured deletion strategies provide meaningful improvements.
- *Link Prediction (VGAE)* [1]: Uses a variational graph autoencoder to estimate link probabilities. Edges with the lowest predicted likelihood are removed, assuming they are structurally weak and less informative.
- *VIKING* [4]: A gradient-based poisoning attack that learns edge deletions using supervised training on model internals. Though effective, it assumes white-box access to gradients and architecture.
- *NAG-R* [9]: A recent black-box method that combines Nesterov [6] Accelerated Gradient with rewiring to craft adversarial perturbations. It addresses common limitations in black-box optimization (e.g., local optima) and maintains stealth by preserving node degrees.
- *Nettack* [10]: A foundational gray-box attack designed for targeted node misclassification. It optimizes edge and feature perturbations based on surrogate gradient objectives and enforces structural constraints for stealth. Nettack is widely used to benchmark robustness in transductive node classification.

Table 1: Characteristics of datasets used in our experiments.

Dataset	# Nodes	# Edges	# Features
CoraFull [2]	19,793	126,842	8,710
Amazon-Photo [8]	7,650	238,162	745
PubMed [7]	19,717	88,648	500

Metric. We evaluate attack effectiveness using specific metric for the retrieval task. We report the *Average Rank Demotion (ARD)*, measuring how much the target node's rank degrades after the attack. This

3.2 Findings

Our experiments yield several key findings that highlight the effectiveness, efficiency, and robustness of our proposed attack strategy:

(1) As shown in Table 2, our method consistently achieves the highest performance across all three datasets (PubMed, Amazon-Photo, and CoraFull) and across both retrieval models (EPAGCL and GCA). Notably, our black-box method surpasses even gray-box attacks such as Nettack [10], which has access to gradient approximations and internal model parameters. Despite this privileged access, Nettack performs even worse than simple structural heuristics like Degree or PageRank in several cases. This suggests that surrogate modeling using linear approximations fails to capture the complex, nonlinear interactions present in modern GNNs. Meanwhile, VIKING [4] underperforms across all settings despite having full model access and using perturbation-based optimization. This suggests that its global modeling lacks the localized sensitivity necessary to identify high-impact deletions. In contrast, our approach benefits from subgraph sampling and localized edge scoring, allowing it to better capture context-dependent edge effects. Moreover, in some scenarios, especially on PubMed using GCA, we observe negative rank demotions. This indicates that edge deletions unexpectedly improve the target node's ranking. This reveals that some edges may contribute to noisy or adversarial message passing, and their removal can lead to richer or more coherent embeddings. Sparse datasets are more susceptible to such behavior, where each edge plays a disproportionately critical role in structural signal propagation. This addresses **RQ1** by demonstrating our method's superior performance in rank demotion compared to all baselines, including those with internal model access.

(2) Although NAG-R is also a black-box approach, it fails to execute within a reasonable timeframe on larger graphs like CoraFull. This is due to its quadratic time complexity $O(T \times B \times N^2)$ and memory complexity $O(N^2)$, originating from dense adjacency matrix operations. The inefficiency arises from three main bottlenecks: (i) cloning and storing dense adjacency matrices at each iteration, (ii) computing gradients over the entire graph even though perturbations

Table 2: Comparison of attack performance under different budget constraints (B=1 to 5) for EPAGCL and GCA models across three datasets. Best results are in bold

		EPAGCL					GCA				
Dataset	**Method**	**B=1**	**B=2**	**B=3**	**B=4**	**B=5**	**B=1**	**B=2**	**B=3**	**B=4**	**B=5**
Pubmed	Degree	14.63	16.70	24.33	22.75	27.58	10.61	15.14	21.24	18.18	24.95
	Page Rank	13.51	14.48	20.44	16.86	21.10	8.47	9.56	15.71	13.73	18.31
	Random	2.80	5.23	7.63	11.71	13.69	-2.60	-8.11	-9.59	-12.71	-14.39
	Viking	2.06	5.53	8.00	11.37	15.40	-2.83	-5.88	-10.21	-13.77	-13.00
	Link Prediction	4.81	-0.68	2.41	-0.48	4.83	-2.61	-19.06	-26.09	-34.23	-39.85
	Nag-R			OOT			7.83	12.14	16.17	20.58	23.02
	Nettack	1.94	3.54	6.33	9.98	14.06	-6.03	-12.87	-21.43	-28.69	-30.06
	Proposed	**23.32**	**38.79**	**53.18**	**66.28**	**76.83**	**37.78**	**57.68**	**69.99**	**77.53**	**81.12**
Amazon	Degree	3.77	3.68	6.44	6.60	8.82	2.13	3.41	5.96	5.82	7.46
	Page Rank	2.99	3.04	5.71	5.21	6.23	2.23	2.81	5.04	5.52	6.53
	Random	0.60	-0.17	0.77	1.45	0.74	0.18	0.63	1.02	0.77	0.88
	Viking	0.23	0.20	0.59	0.32	0.65	0.04	0.30	0.77	0.33	0.79
	Link Prediction	-1.21	-2.11	-3.45	-4.60	-5.66	1.29	1.37	2.10	1.99	2.47
	Nag-R	0.22	0.54	0.98	1.55	2.11	1.59	3.00	4.15	5.51	6.56
	Nettack	0.02	0.88	1.69	2.05	2.86	-0.38	-0.73	-0.41	-0.76	-0.45
	Proposed	**5.73**	**9.06**	**11.18**	**13.53**	**15.34**	**5.18**	**7.46**	**9.42**	**10.88**	**12.77**
CoraFull	Degree	4.89	4.59	7.29	8.38	10.88	2.84	4.24	8.02	10.45	13.20
	Page Rank	4.87	4.51	7.00	7.14	9.40	1.71	4.71	6.98	9.39	12.83
	Random	0.80	2.45	2.21	2.43	3.06	1.36	2.64	3.11	3.02	4.26
	Viking	0.98	0.20	1.61	4.45	2.86	0.69	0.26	2.79	1.72	4.91
	Link Prediction	1.07	1.33	2.21	2.68	3.67	0.24	1.78	3.20	3.42	5.06
	Nag-R			OOT					OOT		
	Nettack	0.68	1.37	2.85	2.78	2.61	3.74	6.47	8.72	13.92	17.97
	Proposed	**5.88**	**9.37**	**12.25**	**14.76**	**17.02**	**11.42**	**17.36**	**22.29**	**25.46**	**28.26**

affect only a local neighborhood, and (iii) performing sequential attacks per node. In contrast, our method scales linearly with the target node's degree and the number of sampled subgraphs, making it more practical for real-world deployment. Together with the time complexity analysis of our proposed method in Section 2.4, we addressed **RQ2** by demonstrating that our approach is significantly more scalable and computationally efficient than the existing black-box attack baseline.

(3) Our method continues to show strong performance as the budget increases. The effectiveness is not limited to single-edge attacks. It persists and even strengthens under larger budgets. This indicates that our scoring mechanism not only identifies impactful edges but also ensures that deletions act synergistically. In contrast, baseline methods often plateau or slightly improve at higher budgets due to redundant or suboptimal edge selections. The increasing performance gap between our approach and others as the budget grows further highlights its ability to adaptively identify structurally critical edges for demotion. This validates **RQ3** by confirming that our method maintains and improves effectiveness as the perturbation budget scales.

4 Concluding Remarks

We proposed a black-box graph poisoning attack that demotes target node rankings via edge removal without accessing model internals. Our method learns to identify high-impact removals effectively by modeling edge influence using sampled subgraphs and local ablation signals. Experiments across three benchmark datasets and two retrieval models show that our approach outperforms both heuristic and gradient-based baselines, scales efficiently to large graphs, and improves as the perturbation budget increases. These findings affirm the effectiveness, scalability, and robustness of our method in practical black-box settings.

References

[1] Seong-Jin Ahn and Myoung Ho Kim. 2021. Variational Graph Normalized AutoEncoders. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong (Eds.). ACM, 2827–2831. doi:10.1145/3459637.3482215

[2] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. arXiv:1707.03815 [stat.ML] https://arxiv.org/abs/1707.03815

[3] Yankai Chen, Yixiang Fang, Yifei Zhang, and Irwin King. 2023. Bipartite Graph Convolutional Hashing for Effective and Efficient Top-N Search in Hamming Space. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben (Eds.). ACM, 3164–3172. doi:10.1145/3543507.3583219

[4] Viresh Gupta and Tanmoy Chakraborty. 2021. VIKING: Adversarial Attack on Network Embeddings via Supervised Network Poisoning. In *Advances in Knowledge Discovery and Data Mining - 25th Pacific-Asia Conference, PAKDD 2021, Virtual Event, May 11-14, 2021, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 12714)*, Kamal Karlapalem, Hong Cheng, Naren Ramakrishnan, R. K. Agrawal, P. Krishna Reddy, Jaideep Srivastava, and Tanmoy Chakraborty (Eds.). Springer, 103–115. doi:10.1007/978-3-030-75768-7_9

[5] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 2019*. https://openreview.net/forum?id=H1gL-2A9Ym

[6] Yurii Nesterov. 1983. A method for unconstrained convex minimization problem with the rate of convergence O (1/k2). In *Dokl. Akad. Nauk. SSSR*, Vol. 269. 543.

[7] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Mag.* 29, 3 (2008), 93–106. doi:10.1609/AIMAG.V29I3.2157

[8] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Pitfalls of Graph Neural Network Evaluation. arXiv:1811.05868 [cs.LG] https://arxiv.org/abs/1811.05868

[9] Shu Zhao, Wenyu Wang, Ziwei Du, Jie Chen, and Zhen Duan. 2023. A Black-Box Adversarial Attack Method via Nesterov Accelerated Gradient and Rewiring Towards Attacking Graph Neural Networks. *IEEE Trans. Big Data* 9, 6 (2023), 1586–1597. doi:10.1109/TBDATA.2023.3296936

[10] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2019. Adversarial Attacks on Neural Networks for Graph Data. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, Sarit Kraus (Ed.). ijcai.org, 6246–6250. doi:10.24963/IJCAI.2019/872