SHORT-PAPER

Adversarially Attacking Graph Properties and Sparsification in Graph Learning

CHUNJIANG ZHU, Old Dominion University, Norfolk, VA, United States

BLAKE B GAINES, University of Connecticut, Storrs, CT, United States

JINGFENG DENG, The University of North Carolina at Greensboro, Greensboro, NC, United States

JINBO BI, University of Connecticut, Storrs, CT, United States

PDF Download
3746252.3760882.pdf
27 January 2026
Total Citations: 0
Total Downloads: 68

Published: 10 November 2025

Citation in BibTeX format

CIKM '25: The 34th ACM International Conference on Information and Knowledge Management
November 10 - 14, 2025
Seoul, Republic of Korea

Conference Sponsors:
SIGWEB
SIGIR

Adversarially Attacking Graph Properties and Sparsification in Graph Learning

Chunjiang Zhu*
Old Dominion University
Norfolk, VA, USA

Blake Gaines
University of Connecticut
Storrs, CT, USA

Jing Deng
UNC Greensboro
Greensboro, NC, USA

Jinbo Bi
University of Connecticut
Storrs, CT, USA

Abstract

Graph neural networks and graph transformers *explicitly or implicitly* rely on fundamental properties of the underlying graph, such as spectral properties and shortest-path distances. However, it is still not clear how these graph properties are vulnerable to adversarial attacks and what impacts this has on the downstream graph learning. Moreover, while graph sparsification has been used to improve computational cost of learning over graphs, its susceptibility to adversarial attacks has not been studied. In this paper, we study adversarial attacks on graph properties and graph sparsification and their impacts on downstream graph learning, paving the way for how to protect against these potential attacks. Our proposed methods are effective in attacking spectral properties, shortest distances, and graph sparsification as demonstrated in our experimental evaluation.

CCS Concepts

• **Theory of computation → Graph algorithms analysis**; • **Security and privacy → Software and application security**.

Keywords

Graph Machine Learning, Spectral Graph Properties, Shortest Distances, Shortest Path Interdiction

ACM Reference Format:
Chunjiang Zhu, Blake Gaines, Jing Deng, and Jinbo Bi. 2025. Adversarially Attacking Graph Properties and Sparsification in Graph Learning. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3746252.3760882

1 Introduction

Graph machine learning, such as graph neural networks (GNNs) [11, 16, 26] and graph transformers [21, 36], has received significant attention, thanks to its robust performance in various graph prediction tasks and abundant instances of graph-structured data in real-life and scientific fields. These advanced graph models *explicitly or implicitly* rely on inherent properties of the underlying graph, such as spectral properties and shortest-path distances. In network and graph data, however, false data can be easily injected by adversaries: spammers can create fake followers on online social networks, false "knowledge" triplets that are hard to verify can be added to a knowledge graph, *etc.* It is crucial to obtain a good understanding of the impact of these adversarial attacks on the graph properties and downstream graph learning, before developing effective defense strategies.

In addition, it is still open to address the tradeoff between the expressive power and the computational cost. Recently, graph sparsification [28], which can approximate a graph by a small subgraph, has been used to improve the computational overhead of graph learning [6, 12, 25, 38]. Graph learning models run faster in small graphs while generating comparable performance [25, 33] if the small graphs are a good approximation of the original graphs. However, it is still under-explored whether this newly added graph sparsification module is secure and reliable and whether it is a vulnerable point for adversarial attacks. For example, if the learned graph representation is used for downstream graph clustering, a dirty spectrally sparsified graph may not contain the key information of the original graph for spectral clustering. Considering routing algorithms for traffic management in wireless networks and machine learning based network resource allocation and optimisation in wireless networks, if a GNN model needs to work on a sparsified network, and an adversary attacks sparsification process, it can alter the GNN prediction.
Our *contributions* in this exploration are summarized as follows:

- We study adversarial attacks on spectral properties in a graph and spectral sparsification and their impacts on downstream graph learning.
- We investigate adversarial attacks on graph shortest-path distances and graph spanners through linear programming. The proposed method can simultaneously elongate the shortest paths between multiple pairs of vertices using optimal perturbations while maintaining a small runtime.
- We have conducted extensive experiments to demonstrate the effectiveness of our proposed methods in attacking graph properties and sparsification: a small budget of perturbing less than 5% of graph edges can significantly deteriorate the quality of graph clustering results; the shortest path interdiction method enlarges distances for multiple pairs of vertices using optimal perturbations, often much smaller than baselines, under comparable runtime.

Related Work. Adversarial attacks on machine learning have attracted increasing research interest [2, 13, 29]. Small, often unnoticeable, perturbations on the samples designed by the attackers can completely alter the output of the machine learning models. Like the studies on grid data or other data, carefully crafted small perturbations to graph structure and/or node features can also produce models with wrong prediction results [3, 7, 9, 10, 19, 20, 31, 32, 39, 40]. While these methods often directly target attacking a specific task,

*Corresponding author, Email: czhu@odu.edu it is unclear how to attack fundamental properties of the underlying graph, such as spectral properties and shortest distances, and what impacts this has on downstream machine learning tasks.

Our problem of attacking shortest paths is also known as the shortest path interdiction [15, 22]. Miller et al. [22] increase edge weights such that a pre-chosen path connecting two vertices becomes their shortest path in the perturbed graph. This method can be used to solve a simplified version of our problem when only a single vertex pair is considered. However, their method requires calculating the second shortest path using Yen's algorithm [35], and thus incurs cubic computational cost. In contrast, our proposed model works for multiple vertex pairs and even all pairs, using only the shortest path algorithm.

2 Attacking Spectral Properties and Sparsifiers

Spectral Properties. Spectral graph theory studies inherently combinatorial graphs from an algebraic perspective and spectral properties have been exploited explicitly or implicitly in graph machine learning, e.g., spectral clustering [23], Laplacian smoothing [25], and semi-supervised learning [5]. Define the graph *Laplacian* of a graph G as $L = D - A$ where A is the adjacency matrix of G and D is a diagonal matrix with the i^{th} diagonal entry equal to the sum over the i-th row of A. The eigenvalues λ and eigenvectors $\mathbf{u}$ of the Laplacian matrix are called Laplacian eigenvalues and eigenvectors. Our goal is to study the impacts of adversarial attacks on spectral graph properties and subsequent graph learning tasks based on these properties.

We consider undirected unweighted graphs and use edge flipping as the attacking method, specifically, adding new edges by flipping 0 to 1 in the adjacency matrix A and deleting existing edges by flipping 1 to 0 to get the graph A' after the attack. To measure the changes in the Laplacian eigenvalues and eigenvectors, we adopt the mean square error (*MSE*) between the eigenvalues and normalized mutual information (*NMI*) of the eigenvectors, respectively. Then the attacking problem under a budget of flipping at most F edges can be re-formulated as:

$$\begin{aligned}&\max_{A' \in R^{n\times n}} O(A, A') = f(\lambda, \lambda') + \alpha \cdot g(\mathbf{u}, \mathbf{u}')\\ =&\max_{A' \in R^{n\times n}} \frac{1}{n} \sum_{i=1}^{n} \left((\lambda_i - \lambda_i')^2 + \alpha \cdot \mathrm{NMI}(\mathbf{u}_i, \mathbf{u}_i')\right), \qquad (1)\\ &\text{s. t. } ||A - A'||_0 \leq 2F, (A')^T = A'.\end{aligned}$$

where α controls the relative weight. Because of the undirected nature of graphs, A' is symmetric and the zero-norm of the difference between A and A' is not larger than $2F$.

The search space in Eq. (1) is over all possible n-vertex attacked graphs. However, we can generate a set of candidates for edge flips, $C = C_a \cup C_d$, which includes candidates to add C_a and candidates to delete C_d. For adding edges, we randomly sample $|C_a|$ candidate vertex pairs with no edge. For C_d, we include all existing edges in the clean graph while avoiding generating any singleton vertices.

To avoid computing the eigenvalues from scratch and reduce complexity, we employ the fast approximation method of Laplacian eigenvalues in [3] as shown in Theorem 2.1. The approximation of eigenvectors is more complicated and time-consuming and thus will be addressed in future study. Among the edge candidates C, we greedily select the top F flips that maximize the objective in Eq. (1).

Theorem 2.1 (Thm. 4 in [3]). *Consider the normalized graph Laplacian $\mathbb{L} = D^{-1/2} L D^{-1/2}$. The eigenvalue λ_y' of $\mathbb{L}'$ obtained after a single edge flip (i, j) can be approximated by $\lambda_y' = \lambda_y + (1 - 2A_{i,j})\left((u_{yi} - u_{yj})^2 - \lambda_y(u_{yi}^2 + u_{yj}^2)\right)$.*

The computational complexity is $O(n^{\omega} + |C|)$, where $\omega \in [2, 2.373)$ (2.373 being the matrix multiplication exponent). It first performs eigendecomposition to obtain eigenvalues/vectors of the input graph in $O(n^{\omega})$ time [8]. For each candidate edge in C, it then computes the approximate eigenvalues in $O(1)$ time. Finally, top F flips are selected as the perturbations. We emphasize that our method is easily parallelizable because the evaluations of different edge flips in C are completely independent. In contrast, one could iteratively select the top edge flip and then estimate the objective based on the remaining candidates and the new attacked graph. We found that this method has similar objective values but is much more computationally expensive.

Spectral Sparsifiers. Spectral sparsifiers are an important type of graph sparsifiers that aims to approximately preserve the Laplacian eigenvalues and eigenvectors [28]. A $(1 + \epsilon)$-spectral sparsifier H requires that for all continuous vectors $\mathbf{x}$, the quadratic form of its Laplacian L_H, i.e., $\mathbf{x}^T L_H \mathbf{x}$, approximates that of the original graph within $1 \pm \epsilon$ factor. Lee and Sun [17] showed that H can be constructed with size $O(n/\epsilon^2)$ in nearly linear time, $\tilde{O}(m/\epsilon^{O(1)})$ in the number of edges m. Despite the wide use of spectral sparsification in graph learning, it is still unclear whether this additional module is vulnerable to adversarial attacks.

We can define adversarial attacks in different ways: when targeting the *spectrum preservation*, we want to see the Laplacian eigenvalues and eigenvectors of the sparsifier S for the attacked graph to be significantly different from those of the sparsifier S' for the original graph, i.e., maximize $O(S, S')$. Since spectral sparsification itself well approximates the Laplacian eigenvalues/vectors, we can simplify the problem by removing the sparsification process and turn to the same objective $O(A, A')$ in Eq. (1). In the experiment, we evaluate both objectives and verify the success of our attack method in both setups. In addition, we can target the *number of edges* in the sparsifier and strive for that the size of the sparsifier for the attacked graph is considerably larger than that of the sparsifier for the input graph (e.g., $\frac{|S'|}{|S|} > c$ for constant $c > 1$), reducing the gain due to the sparsification. Solving this would require a novel breakthrough and is currently under investigation. Finally, attacking the sparsification runtime for a large ratio of runtime with and without the attack is also an interesting direction.

3 Attacking Graph Distances and Spanners

Graph shortest paths and distances capture the graph structural information and have been used explicitly or implicitly for learning over graphs such as in graph neural networks [34, 37], spectral methods [6, 38], and a series of graph kernel methods based on shortest paths and connectivity [4, 14, 27]. Here we study how to maximally change/attack shortest distances given limited edge weight perturbations. We choose the perturbations as *increasing the weight $W(e)$* of an edge e by $d(e)$, which subsumes edge deletions (with $d(e) = \infty$), while ignoring decreasing edge weights as it could make the problem not affordable. Towards unnoticeable attacks, we define two types of budgets to constrain the changes: a *local budget* $b(e)$ indicates the maximum added weight for each edge and a *global budget* B specifies the maximum total added weights for all the edges.

A general attacking objective is that for each vertex pair u, v of multiple targeted pairs P in a graph G, their shortest distance in the attacked graph G', $dist(u, v, G')$, is no smaller than $h(dist(u, v, G))$ with $h(\cdot)$ being the distortion function. The formulated linear program would look like the following with requirement (a):

$$\begin{aligned} &\min \sum_{e \in E} d(e) \\ s.t.\ &(a)\ \forall u, v \in P, dist(u, v, G') \geq h(dist(u, v, G)) \\ &(b)\ \forall p(u, v) \in Q, \sum_{e \in p} (W(e) + d(e)) \geq h(dist(u, v, G)) \\ &(c)\ \forall p(u, v) \in \underline{Q}, \sum_{e \in p} (W(e) + d(e)) \geq h(dist(u, v, G)) \\ &\text{for } e \in E, d(e) \in [0, b(e)] \\ &\sum_{e \in E} d(e) \leq B \end{aligned} \tag{2}$$

Here requirement (a) needs to be realized. A naive method is to compute, for every pair $u, v \in P$, paths $Q_{u,v}$ between u and v with distance smaller than $h(dist(u, v, G))$ and then take union of these paths to get Q. If all the paths in Q have a distance at least $h(dist(u, v, G))$ as in constraint (b), then requirement (a) holds since the distance of a path cannot decrease after adding edge weights in the graph. This method clearly has a high computational cost to compute the path set Q, which is at least as expensive as finding top-K shortest paths, $O(Kn^3)$ [35]. To avoid the overhead due to Q, we observe that a solution of the linear program with only a subset $\underline{Q}$ of paths in Q is *optimal*, as long as requirement (a) is met.

Observation 3.1. *Let S be the solution of solving the linear program (2) with constraint (c). If requirement (a) is satisfied, S is optimal.*

This inspires our main algorithm in Alg. 1, where we only include pair-wise shortest paths in $\underline{Q}$ and generate constraints gradually.

Algorithm 1 Attacking Pair-wise Shortest Distances

```
Input: $G(V, E, W)$, $h(\cdot)$, targeted vertex pairs $P$, local budget $b(e)$, and
    global budget $B$
Output: Added weight $d(e)$ for every edge $e$
1: $d(\cdot) = 0$; $\underline{Q} = \emptyset$
2: while $true$ do
3:     for every $u, v \in P$ do
4:         Compute the shortest paths $p$ between $u$ and $v$ in $G'$ under $d(\cdot)$
5:         if $dist(p) \geq h(dist(u, v, G))$ then
6:             Continue;
7:         $\underline{Q} = \underline{Q} \cup \{p\}$
8:     if $\underline{Q}$ was not changed then
9:         Break;
10:    Solve the linear program (2) with constraint (c) to get new $d(\cdot)$
11: return $d(\cdot)$;
```

Distance between Two Sets. It is well-motivated that in transportation networks one is interested in elongating the shortest distance between two communities. The communities can be formulated as source and target vertex sets S and T. The distance between two sets of vertices U, V is defined as $dist(U, V) = \min\{dist(u, v) \mid u, v \in U \times V\}$. To attack this type of distances, we add a virtual vertex s pointing to every vertex in S with weight 0 and a virtual vertex t pointing from every vertex in T with weight 0, and then turn to Alg. 1 with $P = \{(s, t)\}$. It is easy to see $dist(s, t) = dist(S, T)$. We need to carefully avoid adding weights to the virtual edges $\{(s, u) \mid u \in S\} \cup \{(v, t) \mid v \in T\}$ though.

Graph Spanners. Graph spanners are a sparse graph structure that approximates shortest distances in a graph [24, 30]. Unfortunately, the robustness of graph spanners to adversarial perturbations remains an intriguing open question. Attacking graph spanners can target the approximation of shortest distances by setting $\forall u, v \in P, dist(u, v, spanner(G')) \geq h(dist(u, v, G))$ for a set P of targeted pairs. However, using this constraint in a linear program can be highly complex: for every tentative G', we have to compute its spanner to verify the distance inequality. A simple yet effective way is to remove $spanner()$ and turn to linear program (2). Since $dist(u, v, Spanner(G')) \geq dist(u, v, G')$, requirement (a) is still satisfied, with possibly sub-optimal attacks w.r.t. the original constraint.

Attacking the Spanner Size. Assume a fixed spanner algorithm, e.g., the greedy algorithm that iteratively picks the minimum edge into the spanner when the edge's end vertices have shortest path distance in the spanner larger than the required value [1]. Since the edge with the minimum weight will be chosen in each iteration, an interesting question is whether we can add appropriate edge weights so that significantly more edges are included in the spanner of the attacked graph.

4 Experimental Results

In this section, we analyze selected results of our experiments.

Datasets. We use five real-world datasets from various domains including *Facebook*, *Cora*, *Citeseer*, *ca-HepTH*, and *LastFM*. Collected from Standford SNAP [18], these datasets cover collaboration networks, social networks, and citation networks. The number of nodes ranges from 1K to 10K and the number of edges ranges from 3K to 88K.

Spectral Graph Properties. To evaluate the deviation of spectral properties, we report the objective values in Eq. (1). We also run spectral clustering on both attacked and clean graphs and calculate the normalized mutual information (*NMI*) between their clustering results to verify the impact on graph learning. A smaller *NMI* value indicates more dissimilar clustering and is preferred. All performance measures are averaged over five runs and reported together with their standard deviation.

For comparison, we implement three heuristic/random baselines: for a budget of F edge perturbations, *Random* randomly deletes edges; *Betweenness* removes edges of top-F largest edge betweenness centrality; and *Pagerank* computes the pagerank of each edge as the mean pagerank of its two vertices and then removes edges with top-F highest pagerank. We evaluate two types of candidates, $C_a \cup C_d$ (with $|C_a| = |C_d|$) and C_d only, and find that the latter results in one order of magnitude larger objective, thus using C_d as candidates and keeping consistent with the baselines. We vary the perturbation rate across $\{1\%, 2\%, 3\%, 4\%, 8\%, 12\%, 16\%, 20\%\}$. As shown in Fig. 1, our attack method almost always achieves the

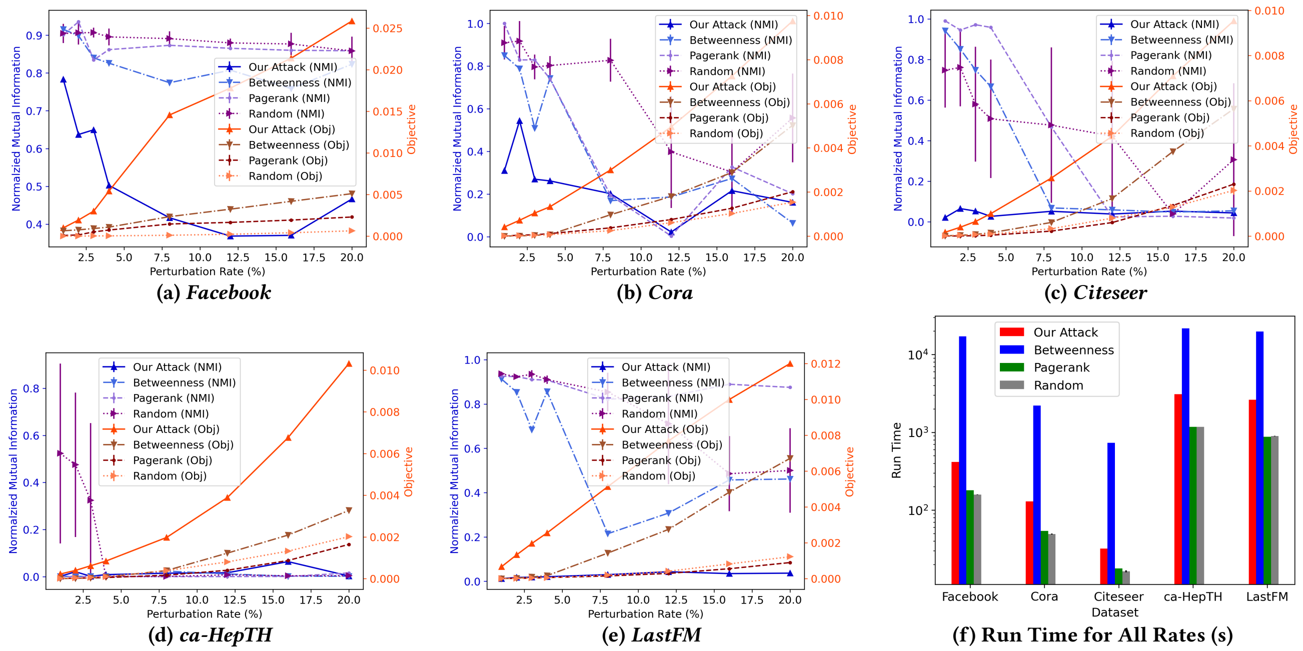
(a) *Facebook* (b) *Cora* (c) *Citeseer* (d) *ca-HepTH* (e) *LastFM* (f) Run Time for All Rates (s)

Figure 1: Performance Measures of Different Attack Methods on Spectral Clustering

poorest clustering quality in terms of the smallest *NMI* values for different perturbation rates among all tested methods. In the *Facebook* and *Cora* datasets in Fig. 1(a,b), the margins between our attack (blue solid line) and the baselines are quite significant, although our *NMI* values are not ideally close to zero. In the *Citeseer*, *ca-HepTH*, and *LastFM* datasets, our attack under small perturbation rates can already drop the *NMI* to almost zero, destroying the clustering quality, as in Fig. 1(c,d,e). *Betweenness* often performs better than *Pagerank*, since it can better characterize the importance of edges.

Fig. 1(f) shows the runtime to complete all the attacks of varied rates from 1% to 20%. Our method has larger runtimes than *Pagerank* and *Random* but much smaller runtimes than *Betweenness*, while still formulating considerably more effective attacks. It should be noted that our method's runtime can be accelerated using multi-core parallel computing, because the computations of different candidate flips can be performed in parallel.

For attacking spectral sparsifiers, we also test the inclusion of the sparsification process, that is maximizing $O(S, S')$ instead of $O(A, A')$ in Eq. (1). The results remain similar as those of maximizing $O(A, A')$. We further vary the approximation parameter ϵ of spectral sparsifiers in $\{0.3, 0.5, 0.7, 0.9\}$ in the *Facebook* and observe that a larger value of ϵ allows more errors in the approximation, leading to slightly worse clustering quality.

Shortest Distances. For each graph data, we assign all edge weights either equally as 1, or according to a uniform distribution between 1 and 2, or a Poisson distribution with $\lambda = 1$ with 1 added to each value. We then randomly select 50 sets of 20 pairs of vertices each for the multiple-pairs case and 50 pairs with 20 vertices each for the sets case. We ensure that the distance between each pair of vertices is at least 5. For every combination of graph, selected vertices, and attacking method, we run the same experiment 3 times and average the relevant metrics. Experiments in which the algorithm reaches

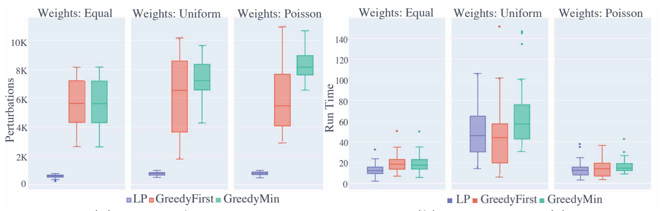
(a) Perturbations (b) Run Time (s)

Figure 2: Performance of Attack Models on Multiple Pairs of Vertices in the *Facebook* network.

200 iterations or 300 paths are automatically terminated. For all experiments, $h(x) = 5x + 0.1$, and the global budget is 1000.

We adopt two greedy baselines from [22] for the comparison. GreedyFirst considers the first edge in the path traversal order, and assigns it whatever weight is needed to make the path satisfy its corresponding constraint in the linear program. GreedyMin is similar but perturbs the edge with the smallest weight instead. For a fair comparison with these baselines, we exclude local budget constraints from the linear program. The runtime of our algorithm does not significantly differ when including local budget constraints.

Fig. 2 plots the performance of attack methods on multiple pairs of vertices in the *Facebook* graph (other results in the full paper), where each box plot shows the distributions of summed edge perturbations (a) or run time (b) for different methods. Across several different graphs, these experiments demonstrate that our method LP significantly outperforms greedy baselines. For example, in *Facebook*, it finds solutions between 7.87 and 10.39 times less costly than the runner-up, depending on how edge weights were assigned.

Acknowledgements

C Zhu was partially supported by NSF grant CCF-2349369. J Bi's research was partially sponsored by NIH grants R01-CA297855 and R01-DA051922 and NSF grant AGS-2426655.

GenAI Usage Disclosure

No GenAI tools were used in any stage of the research, nor in the writing.

References

[1] R. Ahmed, G. Bodwin, S.F. Darabi, K. Hamm, L.J.M. Javad, S. Kobourov, and R. Spence. 2020. Graph spanners: a tutorial review. *Computer Science Review* 37 (2020).

[2] B. Biggio, B. Nelson, and P. Laskov. 2012. Poisoning attacks against support vector machines. In *Proceedings of International Conference on Machine Learning (ICML)*. 1467–1474.

[3] Aleksandar Bojchevski and Stephan Günnemann. 2019. Adversarial attacks on node embeddings via graph poisoning. In *Proceedings of ICML Conference.* 695–704.

[4] K.M. Borgwardt and H.-P. Kriegel. 2005. Shortest-path kernels on graphs. In *Proceedings of ICDM Conference.* 74–81.

[5] D. Calandriello, I. Koutis, A. Lazaric, and M. Valko. 2018. Improved large-scale graph learning through ridge spectral sparsification. In *Proceedings of ICML Conference.* 688–697.

[6] J. Chen, H. Sun, D.P. Woodruff, and Q. Zhang. 2016. Communication-optimal distributed clustering. In *Proceedings of NIPS Conference.* 3720–3728.

[7] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *Proceedings of ICML Conference.* 1115–1124.

[8] James Demmel, Ioana Dumitriu, and Olga Holtz. 2007. Fast linear algebra is stable. *Numer. Math.* 108, 1 (2007), 59–91.

[9] Sofiane Ennadir, Johannes Lutzeyer, Michalis Vazirgiannis, and El Houcine Bergou. 2024. If You Want to Be Robust, Be Wary of Initialization. *Advances in Neural Information Processing Systems* 37 (2024), 23796–23823.

[10] Simon Geisler, Daniel Zügner, and Stephan Günnemann. 2020. Reliable Graph Neural Networks via Robust Aggregation. In *Proceedings of NeurIPS Conference.*

[11] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning.* PMLR, 1263–1272.

[12] Anuj Godase, Md Khaledur Rahman, and Ariful Azad. 2021. GNNfam: utilizing sparsity in protein family predictions using graph neural networks. In *Proceedings of the 12th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics.* 1–10.

[13] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of ICLR Conference.*

[14] L. Hermansson, F. D. Johansson, and O. Watanabe. 2015. Generalized shortest path kernel on graphs. In *Proceedings of International Conference on Discovery Science.* 78–85.

[15] E. Israeli and R. K. Wood. 2002. Shortest-Path Network Interdiction. *Networks* 40, 2 (2002), 97–111.

[16] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR Conference.*

[17] Y.T. Lee and H. Sun. 2017. An SDP-based algorithm for linear-sized spectral sparsification. In *Proceedings of ACM STOC Conference.* 678–687.

[18] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[19] X. Liu, S. Si, X. Zhu, Y. Li, and C. Hsieh. 2019. A unified framework for data poisoning attack to graph-based semi-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS).* 9777–9787.

[20] Nikita Malik, Rahul Gupta, and Sandeep Kumar. 2025. Hyperdefender: A robust framework for hyperbolic gnns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 19396–19404.

[21] Dominic Masters, Josef Dean, Kerstin Klaser, Zhiyi Li, Sam Maddrell-Mander, Adam Sanders, Hatem Helal, Deniz Beker, Ladislav Rampášek, and Dominique Beaini. 2022. Gps++: An optimised hybrid mpnn/transformer for molecular property prediction. *arXiv preprint arXiv:2212.02229* (2022).

[22] Benjamin A Miller, Zohair Shafi, Wheeler Ruml, Yevgeniy Vorobeychik, Tina Eliassi-Rad, and Scott Alfeld. 2021. Optimal Edge Weight Perturbations to Attack Shortest Paths. *arXiv preprint arXiv:2107.03347* (2021).

[23] A.Y. Ng, M.I. Jordan, and Y. Weiss. 2001. On spectral clustering: analysis and an algorithm. In *Proceedings of NIPS Conference.* 849–856.

[24] D. Peleg and A.A. Schaffer. 1989. Graph spanners. *Journal of Graph Theory* 13, 1 (1989), 99–116.

[25] V. Sadhanala, Y.-X. Wang, and R. J. Tibshirani. 2016. Graph sparsification approaches for Laplacian smoothing. In *Proceedings of AISTATS Conference.* 1250–1259.

[26] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. 2021. E (n) equivariant graph neural networks. In *International conference on machine learning.* PMLR, 9323–9332.

[27] N. Shervashidze, P. Schweitzer, E.J. van Leeuwen, K. Mehlhorn, and K.M. Borgwardt. 2011. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research* 12, 77 (2011), 2539–2561.

[28] D.A. Spielman and S.-H. Teng. 2004. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of STOC Conference.* 81–90.

[29] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of ICLR Conference.*

[30] M. Thorup and U. Zwick. 2005. Approximate distance oracles. *J. ACM* 52, 1 (2005), 1–24.

[31] B. Wang and N. Z. Gong. 2019. Attacking graph-based classification via manipulating the graph structure. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security.* 2023–2040.

[32] Jiahao Wu, Ning Lu, Zeiyu Dai, Kun Wang, Wenqi Fan, Shengcai Liu, Qing Li, and Ke Tang. 2024. Backdoor graph condensation. *arXiv preprint arXiv:2407.11025* (2024).

[33] Carl Yang, Yuxiao Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. 2020. Heterogeneous network representation learning: A unified framework with survey and benchmark. *IEEE Transactions on Knowledge and Data Engineering* 34, 10 (2020), 4854–4873.

[34] Y. Yang, X. Wang, M. Song, J. Yuan, and D. Tao. 2019. SPAGAN: shortest path graph attention network. In *Proceedings of IJCAI Conference.* 4099–4015.

[35] Jin Y Yen. 1971. Finding the k shortest loopless paths in a network. *management Science* 17, 11 (1971), 712–716.

[36] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? *Advances in neural information processing systems* 34 (2021), 28877–28888.

[37] J. You, R. Ying, and J. Leskovec. 2019. Position-aware graph neural networks. In *Proceedings of ICML Conference.* 7134–7143.

[38] C.J. Zhu, S. Storandt, K.-Y. Lam, S. Han, and J. Bi. 2019. Improved dynamic graph learning through fault-tolerant sparsification. In *Proceedings of ICML Conference.* 7624–7633.

[39] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of ACM SIGKDD Conference.* 2847–2856.

[40] Daniel Zügner and Stephan Günnemann. 2019. Certifiable robustness and robust training for graph convolutional networks. In *Proceedings of ACM SIGKDD Conference.* 246–256.