Rebuttal Letter for reviewer C92L

Anonymous Author(s)

1 RELATION BETWEEN IMBALANCED AGGREGATION AND OVERLOOKING

We observed that existing noticeability measures show near-zero noticeability when the attack rate is low, i.e., the number of attack edges ≫ the number of real edges (***overlooking***). To overcome this problem, HideNSeek aggregates the edges scores in an imbalance-aware way, using the Area Under the Receiver Operating Characteristic Curve (AUROC). AUROC is a popular method that can measure the imbalance binary classification performance, such as anomaly detection (detailed computation is in the main paper, Appendix A).

2 EXTREME CASES

In Table 1, we report LEO's performance in detecting attack edges on the attack graph with 50% of attack rate. The AUROC scores are reported. We used the Cora dataset with five attack methods used in this paper. We find that LEO performs surprisingly well. First, LEO still outperforms Cosine Sim., a baseline that is independent of the attack rate. Second, the performance decline of LEO over increasing attack rate was small (up to 00%p).

Table 1: [exp. 2] LEO's performance in various attack rate scenario

Attack rate	Random	DICE	PGD	Structack	Metattack
5%	0.890	0.914	0.855	0.950	0.892
10%	0.894	0.899	0.842	0.941	0.823
15%	0.891	0.890	0.835	0.933	0.813
20%	0.887	0.897	0.812	0.924	0.809
50%	0.879	0.881	0.804	0.900	0.815
Baseline*	0.803	0.814	0.794	0.825	0.808

*Measured with Cosine Sim. on attacked graph with attack ratio 50%

3 ADAPTIVE ATTACK EXPERIMENTS

In this section, we explore the performance of HideNSeek and LEO with adaptive attacks.

3.1 On HideNSeek's adaptive attack detection

We tested whether HideNSeek performance w.r.t. adaptive attacks.
Adaptive attack design. We designed an adaptive, gradient-based attack. Specifically, we first trained HideNSeek on a clean graph and computed all the node pair scores generating a score matrix. Then, we replaced the adjacency matrix with the score matrix, and applied PGD attack.
Evaluation setting. We used the Cora dataset with a 10% attack rate. We ran 5 trials and report the mean noticeability (HideNSeek score; i.e. AUROC) and mean node classification accuracy after the evasion attack in Table 2.
Results interpretation. The adaptive attack was highly noticeable and did not bypass HideNSeek (AUROC score = 0.912)

Table 2: [exp. 3.1] HideNSeek's performance against adaptive attack

	Random	PGD	Adaptive	Clean
HideNSeek	0.894	0.874	0.912	-
Node classification (%)	0.803	0.689	0.797	0.814

3.2 On LEO improving GNN robustness

Furthermore, we tested whether LEO improves GNN robustness against adaptive attacks.
Evaluation setting. We use the same gradient-based attack as in Section 3.1. We used the Cora dataset with a 10% attack rate. We ran 5 trials and reported the mean node classification accuracy after the poisoning attack in Table 3.
Results interpretation. LEO effectively improves GCN robustness (0.7%p accuracy increase) even against the adaptive attacks.

Table 3: [exp. 3.2] Node classification accuracy using LEO to purify the attack graph. GCN denotes a GCN model trained on the attack graph and GCN + LEO denotes a GCN model trained on a purified graph using LEO.

	GCN	GCN + LEO
Adaptive	0.786	0.793

4 GUARD BASELINE

In this section, we explore the performance of LEO against GUARD.

4.1 Detecting attack edges

In an extension of Table 2 in the original manuscript, we compare GUARD and LEO in detecting attack edges.
Evaluation setting. We use the Cora dataset with a 10% attack rate. Here, we added GUARD as a competitor. The other settings are identical to those in Section 5.1 in the original manuscript.
Results interpretation. The results are in Table 4. For all attack methods, LEO outperforms GUARD in detecting attack edges.

Table 4: [exp. 4.1] GUARD and LEO's performance against five attack methods

	Random	DICE	PGD	Structack	Metattack
GUARD	0.767	0.730	0.737	0.861	0.760
LEO	0.894	0.899	0.874	0.941	0.894

4.2 Improving GNN robustness

In an extension of Figure 5 in the original manuscript, we compare GUARD and LEO in improving the robustness of GNNs.
Evaluation setting. We used the Cora dataset with a 10% attack rate. The other settings are identical to those in Section 5.1 in the original manuscript.
Results interpretation. The results are in Table 5. For all attack methods, LEO outperforms GUARD in improving GNN robustness.

Table 5: [exp. 4.2] Node classification accuracy using LEO and GUARD to purify the attack graph. GCN + GUARD denotes a GCN model trained on a purified graph using GUARD.

	Random	DICE	PGD	Structack	Metattack
GCN	0.791	0.782	0.685	0.773	0.446
GCN+GUARD	0.787	0.771	0.699	0.774	0.482
GCN+HideNSeek	0.793	0.782	0.710	0.784	0.639

5 RELATIONSHIP BETWEEN HIDENSEEK AND SEMANTIC SIMILARITY IN PAPER [C]

Summary Response. Both papers address the criterion of how similar the attacked graph is to the original graph, but they define it differently and have different objectives.

Paper [c]. [c] defines semantic preservation using an optimal classifier: if the predicted class of a node by the optimal classifier remains unchanged after perturbation, it considers this as preserving the semantic meaning, thus deeming the resulting perturbed graph to look similar. Through this, it defines over-robustness, demonstrating that existing (robust) GNNs suffer from this issue. [c] analyzes perturbation from a node perspective, particularly defining the noticeability of the graph based on the prediction results of an optimal classifier relying on node true labels.

The present work. In contrast, HideNSeek trains LEO, a classifier distinguishing between realistic edges and attack-like edges. It assumes perturbed edges are unnoticeable if they cannot be distinguished from real edges by LEO. This approach analyzes perturbation from an edge perspective, aiming to measure how realistic the given perturbed edges (attack edges) are and directly striving to measure unnoticeability. Additionally, it is independent of separate supervision, such as node classification.

Considering both methods together, it is anticipated that one can analyze whether an attack is unnoticeable according to HideNSeek's criteria while simultaneously maintaining node semantics according to the optimal classifier's criteria. Alternatively, by considering both criteria simultaneously, one can analyze if various attacks are noticeable.