Value at Adversarial Risk: A Graph Defense Strategy Against Cost-Aware Attacks

Junlong Liao[1*], **Wenda Fu**[1*], **Cong Wang**[2], **Zhongyu Wei**[1], **Jiarong Xu**[1†]
[1]Fudan University,
[2]Peking University
jlliao22@m.fudan.edu.cn, {wdfu20,zywei,jiarongxu}@fudan.edu.cn, wangcong@gsm.pku.edu.cn

Abstract

Deep learning methods on graph data have achieved remarkable efficacy across a variety of real-world applications, such as social network analysis and transaction risk detection. Nevertheless, recent studies have illuminated a concerning fact: even the most expressive Graph Neural Networks (GNNs) are vulnerable to graph adversarial attacks. While several methods have been proposed to enhance the robustness of GNN models against adversarial attacks, few have focused on a simple yet realistic approach: valuing the adversarial risks and focused safeguards at the node level. This empowers defenders to allocate heightened security level to vulnerable nodes, while lower to robust nodes. With this new perspective, we propose a novel graph defense strategy RisKeeper, such that the adversarial risk can be directly kept in the input graph. We start at valuing the adversarial risk, by introducing a cost-aware gradient-based graph adversarial attack that takes into account not only cost avoidance and compliance with cost budgets but also addresses the challenges posed by discrete graph data. Subsequently, we present a learnable approach to ascertain the ideal security level for each individual node by solving a bi-level optimization problem. Through extensive experiments on four real-world datasets, we demonstrate that our method achieves superior performance surpassing state-of-the-art methods. Our in-depth case studies provide further insights into vulnerable and robust structural patterns, serving as inspiration for practitioners to exercise heightened vigilance.

Introduction

Graph-structured data has gained significant prevalence across diverse domains, such as social networks (Wu et al. 2022), financial transactions (Dou et al. 2020), knowledge graphs (Zhang et al. 2022) and drug molecular topology design (Jiang et al. 2020). Analyzing such data often involves the utilization of graph neural networks (GNNs). However, the effectiveness and reliability of GNNs can be compromised by adversarial attacks in various forms.

For instance, in financial transaction networks, adversarial accounts can inject deceptive transactions to poison the training data of GNN to pretend legitimacy (Dou et al. 2020). In social networks, spammers can camouflage their social links to evade detection algorithms (Waniek et al. 2018). Such attacks can undermine the performance of GNNs, resulting in severe financial losses and deteriorated user experience. Therefore, ensuring the robustness of GNN models against attacks is a crucial concern for graph-based learning.

Approaching safer and more reliable GNNs, existing efforts to improve robustness mainly fall into two categories. The first line of research focuses on enhancing the GNN model itself, aiming to learn more robust GNNs through adversarial or constrained training (Arghal, Lei, and Bidokhti 2022; Dai et al. 2019; Xu et al. 2019, 2022b) or modifying GNN architectures (Geisler, Zügner, and Günnemann 2020; Zhang and Zitnik 2020; Chen et al. 2021). The second line of research centers on pre-processing input graphs before feeding them to the GNN model (Said, De Luca, and Albayrak 2010; Entezari et al. 2020).

However, few extant works consider the fact that, in real-world scenarios, nodes and edges in graphs are naturally protected in a sense that attacking them incurs certain costs. In practice, defenders can intervene in attacks by assigning customized levels of protection to each node or edge based on their importance or vulnerability. If the cost of proceeding with such actions is prohibitively high due to customized protection for each user, the attacker's efforts could fail. For example, in a communication network, where nodes represent devices, and an edge is created if two devices have communication traffic. Defenders can allocate varying levels of protection measures on each device in the network to prevent attacks, *e.g.*, (1) firewalls with stringent rules such as permitting access only from trusted IP addresses, and (2) access control measures like role-based access control or multi-factor authentication.

In this manner, we are among the first to view the defense problem from a cost-aware perspective: assigning security levels to nodes and edges in a way that maximizes the costs borne by attackers. Methods from this perspective ideally prevent attacks from occurring if costs are properly assigned, which differs from the existing defensive methods that react to attacks after they occur.

An imminent yet significant challenge is how to allocate proper costs to each node of an input graph. There are some heuristic methods for cost allocations: Since research sug-

[*]These authors contributed equally.
[†]Corresponding author.

gests that both high node degrees (Zou et al. 2021) and large entries in eigenvectors corresponding to the largest eigenvalues of the raw adjacency matrix (Chen and Hero 2015) indicate high node robustness, one can simply assign higher costs to nodes with lower degrees or smaller entries in eigenvectors for protection, and vice versa. However, the design of such heuristic costs requires extensive prior knowledge and is specific to given graphs. Consequently, their efficacy may diminish when applied to other networks. To overcome these limitations, we introduce a robust Cost-Aware Graph Defense Strategy (RisKeeper) to identify vulnerable nodes and automatically assign costs to input graphs.

Another concern for ideal cost allocation is that such strategy should be effective against any structural attack available under a given budget, assuming attackers are concerned about costs. This essentially becomes a worst-case defense problem, as the method should be able to handle the strongest attack. The second challenge is therefore how to approximate the worst-case attack. Moreover, the structural attack of perturbing edges manifests as a discrete operation over the graph adjacency matrix, which poses difficulties for optimization problems since algorithms like gradient descent cannot operate on discrete domains. As a solution, we first formulate the attack optimization goal with a discrete domain and propose two approaches to solve it under poisoning and evasion settings, respectively.

In light of the above, as defenders, we respond to a given perturbation by updating cost allocations, while also considering attackers will optimize their strategy under a defensive cost scheme. This interaction between defenders and attackers gives rise to a bi-level problem. By iteratively finding the worst-case attack under the updated cost scheme and optimizing cost allocation based on the new attack strategy, both parties can essentially evolve together through adversarial dynamics. Consequently, the final challenge for us is to formulate this bi-level adversarial evolution problem, and determine the defensive cost allocations. To tackle this challenge, we formulate a minimax bi-level optimization problem, where the inner level represents attacking phase and the outer level represents the defending phase. We then employ robust training methods to solve this problem effectively.

We summarize our main contributions as follows:

- **Novel perspective**: We are among the first to consider attacking costs in our GNN defense method. This perspective is advantageous as it has the potential to deter attacks rather than passively reacting to them.
- **Novel method**: We propose RisKeeper, a cost-aware graph defense strategy through robust learning, which can accommodate various untargeted structural attacks.
- **Extensive evaluation**: Extensive experiment results demonstrate the effectiveness of our learned cost allocation in weakening attacks across multiple settings. Our method outperforms mainstream defense methods.
- **Implications**: Our cost allocation shed light on vulnerable and robust graphlets discovery. Our case study with learned costs suggest that 3-star , 4-circle and tailed-triangle are the most vulnerable graphlets in graphs, which can inspire scholars and practitioners to develop more tailored strategies on these structures in future works regarding graph robustness.

Proposed Method

In this section, we present our proposed graph defense strategy, RisKeeper, with the primary objective of directly mitigating adversarial risk within the input graph. We first value the adversarial risk by introducing a novel cost-aware graph adversarial attack. Then, we propose our defense strategy that entails the acquisition of security levels for individual nodes via a learnable process.

Cost-Aware Graph Adversarial Attack.

When solving for graph adversarial attacks, existing works often overlook the economic feasibility of attacks, potentially resulting in unrealistic attack strategies that exhibit high costs and low benefits. This section intends to introduce a cost-aware graph adversarial attack, that meets economic feasibility requirements of minimizing costs of executing attacks and is subject to a budget of cost.

Optimization Problem of Attacker. Let $G = (V, E)$ represent an undirected, unweighted graph, where the set of nodes is denoted as $V = \{v_1, v_2, ..., v_N\}$ and the set of edges is $E \subseteq V \times V$. The adjacency matrix A of graph G is an $N \times N$ symmetric matrix, where the elements are defined as $A_{ij} = 1$ if $i, j \in E$ or if $i = j$, and $A_{ij} = 0$ otherwise. In cases where nodes possess attributes, the graph G is augmented to include the node attribute matrix $X \in \mathbb{R}^{N \times D}$.

Considering the cost associated with attacking a node v_i represented as c_i, and perturbing an edge associated to the node incurs this cost. It follows that the cost of perturbing an edge equals to the cost sum of the two nodes. For example, when an edge (v_i, v_j) is perturbed, the total cost of executing this attack is given by $c_i + c_j$. The primary objective of a cost-aware graph adversarial attack is to identify a set of edges to perturb E', resulting in a perturbed adjacency matrix A', that maximizes the classification loss $\mathcal{L}$ while minimizing the execution costs. This can be formulated into the following optimization problem:

$$\begin{array}{ll} \max_{A'} & \mathcal{L}(A') - \lambda \sum_{(v_m,v_n)\in E'}(c_m + c_n) \\ \text{s.t.} & \sum_{(v_m,v_n)\in E'}(c_m + c_n) \leq \epsilon, \\ & A'_{ii} = 1, \quad i = 1, \ldots, N, \\ & A'_{ij} \in \{0, 1\} \text{ for } i \neq j, \end{array} \tag{1}$$

where the optimization variable is the $N \times N$ symmetric matrix A' (or E' equivalently), and we simplify the classification loss $\mathcal{L}(g(A', X), Y)$ as $\mathcal{L}(A')$. The first inequality constraint shows the budget of total cost $\epsilon > 0$. The hyperparameter λ quantifies the attacker's sensitivity (or level of importance) towards costs. If the attacker demonstrates a higher degree of sensitivity towards the costs, λ can be set as a larger value. Conversely, if the attacker is less sensitive to the costs involved in the attack, λ takes on a smaller value.

Gradient-based Solution. Solving Problem (1) is a challenging task. Firstly, the discrete nature of graph data renders the utilization of conventional optimization methods, such as gradient descent invalid. Secondly, the inclusion of the cost budget constraint adds to the complexity of the optimization problem.

Before introducing our solution. We first define some notations used throughout the paper. We introduce an $N \times N$ mask matrix to record the edge perturbation, the entries of which form a binary vector denoted as $\mathbf{s} \in \{0,1\}^n$ where $n = N^2$. Specifically, if the edge between node v_i and v_j is modified, $s_{ij} = s_{ji} = 1$, otherwise $s_{ij} = s_{ji} = 0$. Thereby, $\mathcal{L}(A')$ can be equivilantly denoted as $\mathcal{L}(A, \mathbf{s})$. We also introduce an $N \times N$ edge cost matrix, whose entries form a vector denoted as $\mathbf{c}$. Specifically, $c_{ij} = c_i + c_j$, indicating that edge cost equals the cost of two end nodes. In this way, the attacking cost can be written as $\mathbf{c}^T\mathbf{s}$.

Our gradient-based solution is given below. We first relax variables $\mathbf{s} \in \{0,1\}^n$ into a convex hull $\mathbf{s} \in [0,1]^n$, and then maximize the modeled attack loss on the convex set:

$$\begin{aligned} \max_{\mathbf{s}} \quad & f(\mathbf{s}) = \mathcal{L}(\mathbf{A}, \mathbf{s}) - \lambda \mathbf{c}^T\mathbf{s} \\ \text{subject to} \quad & \mathbf{s} \in \mathcal{S}, \end{aligned} \tag{2}$$

where $\mathcal{S}$ is the feasible set $\mathcal{S} = \{\mathbf{s} \mid \mathbf{c}^T\mathbf{s} \leq \epsilon, \mathbf{s} \in [0,1]^n\}$. We can solve this using projected gradient descent (PGD) algorithm (Madry et al. 2017), with the update rule as follows:

$$\mathbf{s}^{(t)} = \Pi_{\mathcal{S}}\big(\mathbf{s}^{(t-1)} + \eta_t \hat{\mathbf{g}}_t\big), \tag{3}$$

where η_t is the learning rate at epoch t, $\hat{\mathbf{g}}_t = \nabla f(s^{t-1})$ is the gradient of the attack objective with respect to $\mathbf{s}$ at epoch $t-1$, and $\Pi_{\mathcal{S}}(\mathbf{a})$ is the projection function:

$$\Pi_{\mathcal{S}}(\mathbf{a}) = \arg\min_{\mathbf{s}\in\mathcal{S}} \|\mathbf{s} - \mathbf{a}\|_2^2. \tag{4}$$

Below, we demonstrate that projection function $\Pi_{\mathcal{S}}$ yields the closed-form solution in Proposition.

Proposition 1 *Let $\mathcal{S}$ be feasible set $\mathcal{S} = \{\mathbf{s} \mid \mathbf{c}^T\mathbf{s} \leq \epsilon, \mathbf{s} \in [0,1]^n\}$, the projection operator with respect to $\mathcal{S}$ is:*

$$\Pi_{\mathcal{S}}(\mathbf{a}) = \begin{cases} P_{[0,1]}(\mathbf{a} - \mu\mathbf{c}), & \text{if} \quad \mu > 0 \quad \text{and} \\ & \mathbf{c}^T P_{[0,1]}(\mathbf{a} - \mu\mathbf{c}) = \epsilon, \\ P_{[0,1]}(\mathbf{a}), & \text{if} \quad \mathbf{c}^T P_{[0,1]}(\mathbf{a}) \leq \epsilon, \end{cases} \tag{5}$$

where $P_{[0,1]}(x) = x$ if $x \in [0,1]$, 0 if $x < 0$, and 1 if $x > 1$.

Detailed proofs can be found in Appendix A.

In the projection function 5, the value of Lagrange multiplier μ as root of $\mathbf{c}^T\mathrm{P}_{[0,1]}(\mathbf{a} - \mu\mathbf{c}) = \epsilon$ could be obtained through bisection method (Boyd and Vandenberghe 2004) over range

$$\left[\frac{a_{\min} - 1}{c_{\max}}, \frac{a_{\max}}{c_{\min}}\right],$$

where $c_{\min} = \min\{c \mid c \neq 0, c \in \mathbf{c}\}$ is the smallest nonzero value in $\mathbf{c}$.

After getting $\mathbf{s}$ under the relaxed convex hull, we can obtain the binary solution through a random sampling using learned parameter $\mathbf{s}$ as probabilities for independent Bernoulli trials: $s_i \sim \text{Bern}(\mathbf{s}_i)$, and repeat until we find a binary solution that complies with the constraint, as stated in Xu et al. (2019). This generates a near optimal solution for the perturbation under original constraint.

Defense Problem.

In this subsection, we model the defense problem and propose a GNN model to learn cost allocation. Then we summarize the overall adversarial training algorithm.

Optimization Problem of the Defender. For defenders to achieve an effective graph defense strategy, their primary goal is to quantify the node/edge value at adversarial risk. They can simulate the means of attackers to find vulnerable edges. By optimizing the cost allocation over time, the final costs should make attackers as difficult to find a reasonable attack strategy as possible. We formulate the bi-level problem as follows:

$$\begin{aligned} \min_{\{c_1,\cdots,c_N\}} \max_{A'} \quad & \mathcal{L}(A') - \lambda \textstyle\sum_{(v_m,v_n)\in E'}(c_m + c_n) \\ \text{subject to} \quad & \textstyle\sum_{(v_m,v_n)\in E'}(c_m + c_n) \leq \epsilon, \\ & A'_{ii} = 1, \quad i = 1,\ldots,N, \\ & A'_{ij} \in \{0,1\} \text{ for } i \neq j. \end{aligned} \tag{6}$$

Learnable Cost. The problem of designing costs is nontrivial. A common approach to cost design typically relies on manually crafted costs by taking into account specific aspects of the properties in the graph, such as degree-based costs. However, it is difficult to identify consistently effective properties, given that different graphs have different characteristics. To tackle this issue, our goal is to quantify and optimize the cost allocation in a learnable manner.

Specifically, we seek to maximize the costs associated with perturbing vulnerable edges, with the intention of disrupting future attacks and redirecting the attacker's focus towards modifying less important unimportant edges. To achieve this objective, we consider the edges modified in attacks prior to the defense as vulnerable, and our goal is to increase the costs associated with modifying these perturbed edges in the future. It is important to note that the total perturbing costs that defenders can impose are inevitably limited, thus efficient cost allocation is rather important under such constraints.

The remaining problem is how to design the cost model. Previous research (Dai et al. 2022) indicates that the robustness of a node is related to its topological structure and features. For example, the higher degree a node has, the less affected it is by structural attack (Zou et al. 2021). The ideal costs of nodes are closely related to their robustness. A higher cost can be allocated to a vulnerable node, making the edges around the node less susceptible to direct attacks.

Since ideal costs for nodes are related to their features and structures, and nodes with similar features and structures should have similar costs, it could be helpful to use a GNN to obtain the cost for each node. At the end of the GNN model, a fully connected layer should be attached to map high-dimensional vectors to individual values for each node. The parameterized costs can be represented as follows:

$$\{c_i\}_{i\in\{1,\cdots N\}} \leftarrow \text{MLP}(\text{GNN}(A, X)),$$

where c_i is parameterized cost for node v_i, and the corresponding cost for edge (v_i, v_j) is $c_i + c_j$.

Algorithm 1: Robust training for learning costs

```
1: Input: $G = (A, X)$, $\mathbf{s}^{(0)}$, modification budget $\epsilon$, learning rate $\beta_t$ and $\eta_t$, iterations $T, T'$, labels $y$, budget $\Delta$,
2: $\theta \leftarrow$ train surrogate model on the graph,
3: $\bar{y} \leftarrow$ predict labels of unlabeled nodes using $\theta$,
4: for $t = 1, 2, ..., T$ do
5:     for $t' = 1, 2, ..., T'$ do
6:         $\mathbf{a}^{(t')} = \mathbf{s}^{(t'-1)} + \eta_{t'} \nabla_{\mathbf{s}} f_\theta(\mathbf{s}^{(t'-1)})$,
7:         call projection operation in Eq. (5).
8:     end for
9:     output the resulting $A'$.
10:    given $\mathbf{s}^{(T')}$, obtain cost parameter $\phi^{(t)}$ by running gradient descent repeatly.
11: end for
12: use the $\phi^{(T)}$ to compute costs $\{c_i\}_{i \in \{1,\cdots,N\}}$,
13: if $\sum_{i=1}^{N}(c_i) \leq \Delta$ then
14:     return $\{c_i\}_{i \in \{1,\cdots,N\}}$,
15: else
16:     return$\{\frac{c_i \Delta}{\sum_{j=1}^{N}(c_j)}\}_{i \in \{1,\cdots,N\}}$.
17: end if
```

Overall Algorithm. We employ the robust training method (Song et al. 2023) to optimize the worst-case loss. The problem is solved as follows. Firstly, we initiate the surrogate model by training it on clean data. Subsequently, we execute an attack on the trained surrogate model. Following this, we assign costs to edges to minimize the objective function of the current attack. We then iterate until convergence is achieved. This process can be understood as identifying vulnerable nodes based on the attack outcomes and subsequently protecting these nodes. More specifically, we can use PGD method to solve the inner maximum (attack) problem and use gradient descent to solve the outer minimize (cost-allocation) problem. By utilizing the trained parameters, we can determine the corresponding cost for each node and edge can be obtained. Our detailed algorithm is presented in Algorithm 1.

The overall time complexity is $O(N^2 + |E|DHF)$, comprising two main components: finding worst-case attack, with a time complexity of $O(N^2)$, and cost allocation, with a time complexity of $O(|E|DHF)$ (which is exactly the complexity of our backbone GCN model) (Kipf and Welling 2017). Here, N and $|E|$ are the number of nodes and edges, H is the number of hidden units of GCN, F is the output dimension of representations, and D is the dimension of node attributes.

Experiments

In this section, we assess the performance of the proposed RisKeeper method on node classification tasks. We first compare RisKeeper against heuristic cost allocation methods, followed by a comparison against several existing defensive methods. We evaluate the effectiveness of the defense strategy by examining the decrease in testing accuracy resulting from the attack, which is mitigated through the defense strategy. Furthermore, we conduct case studies based on learned node costs, aiming to identify prevalent robust and fragile structures in graphs.

Experimental Setup.

Datasets. We use four commonly-used datasets to conduct our experiments, *i.e.*, cora (Mccallum et al. 2000), citeseer (Sen et al. 2008), amazon computers and amazon photo (Shchur et al. 2018). Cora and citeseer both belong to citation networks. Amazon computers and amazon photo are segments of the amazon co-purchase graph, where nodes represent goods, an edge between two nodes indicates that two goods are frequently bought together, node features are bag-of-words encoded product reviews, and class labels are given by the product category. For cora and citeseer, we divide the training set, validation set, and test set according to the default setting (Sen et al. 2008). For amazon computers and amazon photo, the datasets are randomly split into training set (10%), validation set (10%), test set (80%).

Baselines. To compare RisKeeper against other heuristic cost allocation methods, we consider the following methods as baselines.

- *Raw*: no cost protections.
- *Avg.*: assign equal costs to each node.
- *Rand.*: randomly assign costs to each node.
- *Deg.*: assign costs according to the exponential of negative node degrees.
- *Cluster-Coef.*: assign costs according to the exponential of negative nodes clustering coefficient.

To ensure fairness, we normalize the handcrafted costs (*i.e.*, *Avg.*, *Rand.*, *Deg.* and *Cluster-Coef.*) to guarantee that their summation is equal to a specified defense budget Δ.

For the comparison between RisKeeper and other defensive methods, we adopt the following as baselines.

- *GCN-SVD* (Entezari et al. 2020): preprocesses the input adjacency matrix using truncated SVD to get its low-rank approximation.
- *GCN-Jaccard* (Wu et al. 2019): remove the edges in the input adjacency matrix whose two end nodes have small Jaccard similarity.
- *GNN-Median* (Chen et al. 2021): use median aggregation as robust aggregation function for GNNs.

For these defensive methods, an average allocation of cost is assigned that ensures equal total node costs as is learned by RisKeeper.

Implementation Details. We use GCN for both surrogate models and the cost model. The number of hidden units is set to 32 for all hidden layers. A 1-layer MLP is attached to the end of the cost model. We employ Adam algorithm (Kingma and Ba 2014) with an initial learning rate of 0.01 to optimize models. For Cost-Aware PGD, the dropout rate is set to 0.5. Cross-entropy loss is used for $\mathcal{L}$. To balance the differences between $\mathcal{L}$ and cost loss caused by varying numbers of attacked edges in different datasets, λ is set to $\frac{0.001}{|E|}$. Without loss of generalizability, the single node cost

	cora	citeseer	amazon computers	amazon photo
Raw	0.290 ± 0.032	0.212 ± 0.041	0.427 ± 0.026	0.475 ± 0.004
Avg.	0.746 ± 0.004	0.659 ± 0.011	0.788 ± 0.005	0.854 ± 0.016
Rand.	0.573 ± 0.009	0.489±0.016	0.714 ± 0.014	0.801 ± 0.013
Deg.	0.743 ± 0.004	0.649±0.022	0.756 ± 0.023	0.826 ± 0.049
Cluster-Coef.	0.746 ± 0.011	0.589 ± 0.132	0.777 ± 0.013	0.842 ± 0.033
RisKeeper	**0.795 ±0.004**	**0.707 ± 0.003**	**0.819 ± 0.017**	**0.874 ± 0.018**
Unattacked	0.811 ± 0.004	0.716 ± 0.003	0.882 ± 0.003	0.927 ± 0.002

Table 1: The test accuracy under Cost-Aware PGD attack with different costs in evasion setting. Unattacked denotes the test accuracy on the clean graph. Total attack cost budget is set to be $0.08|E|$.

	cora	citeseer	amazon computers	amazon photo
Raw	0.344 ± 0.264	0.196 ± 0.049	0.490 ± 0.030	0.528 ± 0.036
Avg.	0.698 ± 0.084	0.649 ± 0.008	0.798 ± 0.032	0.865 ± 0.016
Rand.	0.605 ± 0.083	0.423 ± 0.018	0.798 ± 0.007	0.832 ± 0.011
Deg.	0.736 ± 0.011	0.630 ± 0.027	0.809 ± 0.016	0.849 ± 0.025
Cluster-Coef.	0.728 ± 0.009	0.561 ± 0.169	0.808 ± 0.014	0.856 ± 0.015
RisKeeper	**0.791 ± 0.004**	**0.708 ± 0.003**	**0.830 ± 0.008**	**0.875 ± 0.018**
Unattacked	0.811 ± 0.004	0.716 ± 0.003	0.882 ± 0.003	0.927 ± 0.002

Table 2: The test accuracy under Cost-Aware PGD attack with different costs in poisoning setting. Unattacked denotes the test accuracy on the clean graph. Total attack cost budget is set to be $0.08|E|$.

budget is set to 1, and the total node cost budget Δ is set to $0.995|V|$.

We use Cost-Aware PGD to generate perturbed graphs, on which different defensive models are then trained. We evaluate our model and baselines under different attacking cost budget constraint (*i.e.*, $0.08|E|$, $0.16|E|$, and $0.24|E|$) and under evasion and poisoning setting respectively. Our codes are available at: https://github.com/songwdfu/RisKeeper.

Experiment Results

Performance Comparison against Cost Allocation Methods. In Tables 1 and 2, we present the test accuracy under Cost-Aware PGD attack with different costs allocation schemes in evasion and poisoning settings. The results demonstrate that even if the defender invests the same amount of costs to defend against the attack, different cost allocations will lead to different results. The cost allocation obtained through robust training has the best defending effect. Average allocation of costs ranked second, while degree-based allocations achieved slightly inferior performance. It is noticeable that the performance of clustering-coefficient-based cost allocation method deteriorated a lot on the Citeseer dataset, we therefore severely doubt its applicability in general cases.

Performance Comparison against Defensive Methods. To the best of our knowledge, other state-of-the-art defensive methods do not incorporate cost allocation. Hence, we compare their performance under equal cost allocation, the performance comparison results are shown in Table 3. The results show that RisKeeper outperforms other models in most of the settings for both evasion and poisoning attacks. By allocating costs via our proposed method, the attackers' capability of reducing classification performance by selectively perturbing edges is greatly undermined. For traditional defensive methods, experimental results have shown that GCN-Jaccard is usually superior to the others, though GNN-SVD can sometimes achieve optimal results when the perturbed edges ratio is larger. We infer that since GCN-SVD is designed mainly for the change of small singular values of the adjacency matrix caused by Nettack, its performance under untargeted attack with small numbers of perturbations may not be the most ideal. In most cases, the performance of GCN-Median is not the most ideal, in some cases it is even worse than GCN. We infer the median aggregation function may only be effective under some specific circumstances. In summary, in both cases of evasion and poisoning attacks, the optimal defense effect can be achieved by RisKeeper by learning a cost allocation when the attacking budget is limited. In contrast with other defensive methods, RisKeeper proactively interfere the attack by allocating costs to nodes and edges. This defense method does not depend on specific preprocessings of graph data or designs of specific robust GNN architectures, thereby fully lifting the constraints on GCN models during defending, which could be advantageous in various situations.

The results on the last column of Table 3 suggest that RisKeeper would not undermine the performance on clean data, as we do not modify GNNs or input data as baselines do. In contrast, methods such as GCN-Jaccard and GCN-SVD exhibit inferior performance when applied to clean data. This decrease in performance is attributed to their inherent denoising process, which can be counterproductive for clean data. GCN-Median shows a slight decline in per-

		Eva-0.08	Eva-0.16	Eva-0.24	Poi-0.08	Poi-0.16	Poi-0.24	Unattacked
cora	GCN	0.746 ± 0.004	0.703 ± 0.008	0.659 ± 0.032	0.739 ± 0.005	0.696 ± 0.010	0.650 ± 0.028	0.811 ± 0.004
	GCN-SVD	0.754 ± 0.007	0.737 ± 0.006	**0.706 ± 0.025**	0.721 ± 0.009	0.707 ± 0.019	0.680 ± 0.033	0.765 ± 0.002
	GCN-Jaccard	0.756 ± 0.004	0.737 ± 0.013	0.703 ± 0.022	0.744 ± 0.007	0.722 ± 0.011	**0.701 ± 0.026**	0.789 ± 0.007
	GNN-Median	0.755 ± 0.004	0.722 ± 0.006	0.674 ± 0.026	0.756 ± 0.010	0.711 ± 0.018	0.662 ± 0.026	0.800 ± 0.002
	Ours	**0.795 ± 0.004**	**0.746 ± 0.014**	0.680 ± 0.024	**0.791 ± 0.004**	**0.733 ± 0.011**	0.659 ± 0.021	**0.811 ± 0.004**
citeseer	GCN	0.659 ± 0.003	0.621 ± 0.014	0.594 ± 0.007	0.649 ± 0.003	0.604 ± 0.011	0.577 ± 0.011	0.716 ± 0.003
	GCN-SVD	0.651 ± 0.008	0.647 ± 0.005	**0.634 ± 0.010**	0.645 ± 0.011	0.628 ± 0.007	**0.617 ± 0.008**	0.654 ± 0.002
	GCN-Jaccard	0.680 ± 0.006	0.651 ± 0.009	0.622 ± 0.012	0.668 ± 0.007	0.635 ± 0.010	0.600 ± 0.019	0.716 ± 0.002
	GNN-Median	0.662 ± 0.007	0.623 ± 0.007	0.599 ± 0.010	0.663 ± 0.017	0.617 ± 0.011	0.588 ± 0.019	0.703 ± 0.002
	Ours	**0.708 ± 0.003**	**0.671 ± 0.014**	0.624 ± 0.007	**0.708 ± 0.003**	**0.663 ± 0.011**	0.605 ± 0.011	**0.716 ± 0.003**
computers	GCN	0.788 ± 0.005	0.756 ± 0.010	0.712 ± 0.018	0.792 ± 0.032	0.798 ± 0.008	**0.784 ± 0.005**	0.882 ± 0.003
	GCN-SVD	0.735 ± 0.003	0.709 ± 0.009	0.674 ± 0.016	0.785 ± 0.006	0.770 ± 0.009	0.753 ± 0.006	0.779 ± 0.006
	GCN-Jaccard	0.788 ± 0.005	0.756 ± 0.010	0.708 ± 0.025	0.819 ± 0.006	**0.802 ± 0.006**	0.783 ± 0.003	0.882 ± 0.003
	GNN-Median	0.788 ± 0.013	0.757 ± 0.010	0.711 ± 0.026	0.788 ± 0.007	0.748 ± 0.041	0.741 ± 0.016	0.869 ± 0.004
	Ours	**0.819 ± 0.017**	**0.782 ± 0.014**	**0.728 ± 0.024**	**0.830 ± 0.008**	0.798 ± 0.012	0.758 ± 0.013	**0.882 ± 0.003**
photo	GCN	0.854 ± 0.016	0.832 ± 0.011	0.790 ± 0.048	0.865 ± 0.016	0.846 ± 0.013	**0.825 ± 0.023**	0.927 ± 0.002
	GCN-SVD	0.829 ± 0.016	0.807 ± 0.011	0.770 ± 0.044	0.837 ± 0.016	0.826 ± 0.014	0.783 ± 0.020	0.878 ± 0.002
	GCN-Jaccard	0.854 ± 0.016	0.831 ± 0.011	0.790 ± 0.048	0.864 ± 0.011	0.844 ± 0.014	0.824 ± 0.020	0.927 ± 0.002
	GNN-Median	0.845 ± 0.015	0.822 ± 0.013	0.767 ± 0.053	0.851 ± 0.022	0.826 ± 0.013	0.783 ± 0.026	0.913 ± 0.004
	Ours	**0.874 ± 0.018**	**0.839 ± 0.018**	**0.809 ± 0.040**	**0.875 ± 0.018**	**0.851 ± 0.014**	0.818 ± 0.014	**0.927 ± 0.002**

Table 3: The testing accuracy of defensive methods against Cost-Aware PGD attack in evasion and poisoning settings. Unattacked represents model performance on clean data. Eva-0.08, Eva-0.16 and Eva-0.24 represents Cost-Aware PGD attack with cost budget constraint $0.08|E|$, $0.16|E|$, and $0.24|E|$ under evasion setting. Poi-0.08, Poi-0.16 and Poi-0.24 represents Cost-Aware PGD attack with cost budget constraint $0.08|E|$, $0.16|E|$, and $0.24|E|$ under poisoning setting.

	Meta-0.01	Meta-0.02	Meta-0.03	Unattacked
GCN	0.797 ± 0.007	0.787 ± 0.005	0.781 ± 0.004	0.811 ± 0.004
GCN-Jaccard	0.779 ± 0.005	0.776 ± 0.008	0.769 ± 0.007	0.789 ± 0.007
GCN-SVD	0.716 ± 0.009	0.713 ± 0.003	0.710 ± 0.006	0.765 ± 0.002
GNN-Median	0.797 ± 0.010	0.787 ± 0.007	0.781 ± 0.005	0.800 ± 0.002
Ours	**0.800 ± 0.010**	**0.789 ± 0.007**	**0.785 ± 0.005**	**0.811 ± 0.004**

Table 4: The testing accuracy on GNNs Cost-Aware Meta attack under poisoning settings. Unattacked represents model performance on clean data. Meta-0.01, Meta-0.02 and Meta-0.03 represents Cost-Aware Meta attack with cost budget constraint $0.01|E|$, $0.02|E|$, and $0.03|E|$ under poisoning setting.

formance on clean data, likely due to its robust aggregation function which may not always align with the characteristics of clean data.

The Budget Sensitivity. As shown in Table 3, RisKeeper demonstrates a larger advantage under conditions of a limited attacker's budget. The performance gap between RisKeeper and the baseline models narrows with increasing budget. This is because when the attacker's budget increases, the attacker is more likely to ignore our assigned cost as any perturbation would be within the budget. This finding is very insightful as real-world attackers usually suffer from budget constraint.

Transferability to Other Attacks. To evaluate the transferability of our method to other attacks, we further report the performance of RisKeeper and baseline models under Cost-Aware Meta Attack. The Cost-Aware Meta Attack is identical to original Meta Attach (Zügner and Günnemann 2019) except for two aspects. Firstly, its objective function has an additional term $-\lambda \sum_{(v_m,v_n)\in E'}(c_m + c_n)$ that is identical to the Cost-Aware PGD Attack. Secondly, the edge perturbation steps are conducted until the cost of the current perturbation meets the cost budget constraint, *i.e.*, until $\sum_{(v_m,v_n)\in E'}(c_m + c_n) \geq \epsilon$. The final step that caused the cost to exceed the constraint is ignored. λ is set to be $\frac{0.0004}{|E|}$. The experiments are conducted on Cora dataset with different cost budget constraints under poisoning settings.

The results are presented in Table 4. We find that RisKeeper outperforms all baseline models under different cost budget constraints, while causing no decrease in clean-graph performances. This suggests that the costs learned by our method are also effective when used against other untargeted attacks.

Case Study. Based on learned node cost from RisKeeper, we can further analyze the robustness of specific substructures. We focus on graphlets, defined as small connected sub-graphs of a large network (Pržulj, Corneil, and Jurisica 2004), with 2-4 nodes (as shown in Figure 1). We adopt Orbit Counting Algorithm (Orca) (Hočevar and Demšar 2014)

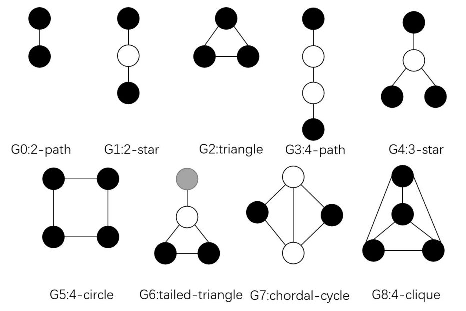

Figure 1: Graphlets with 2–4 nodes and automorphism orbits.

	cora	citeseer	amazon computers	amazon photo
G_0	0.070	1	0	0.881
G_1	0.648	0.803	0.322	0.961
G_2	0	0.636	0.917	0.628
G_3	0.715	0.665	0.299	0.991
G_4	1	0.723	0.232	1
G_5	0.954	0.400	0.927	0.977
G_6	0.718	0.600	0.675	0.949
G_7	0.483	0.236	0.979	0.835
G_8	0.142	0	1	0

Table 5: Average node cost for each type of graphlets. The costs of various graphlets have been normalized.

to count graphlets and orbit signatures of network nodes. Given that costs are predominantly assigned to nodes identified as vulnerable, a higher total cost allocated to a particular graphlet signifies its vulnerability. We calculate the average node cost for each type of graphlet, and the presented in Table 5. We find that defenders can allocate more resources and efforts towards protecting structures of 3-star, 4-circle, and tailed-triangle. In contrast, triangle and 4-clique are generally the most robust graphlets in many cases, as they exhibit relatively low average node costs. These findings provide valuable insights that appeal to managers to raise awareness on 3-star, 4-circle, and tailed-triangle. Conversely, given the relative robustness of triangle and 4-clique structures, managers may consider deprioritizing these areas when resource constraints necessitate a more focused approach.

Related Work

Attack method. In this paper, we focus on untargeted structural attack, which aims to significantly degrade the classification performance of GNNs by deleting existing edges and adding non-existing edges (Xu et al. 2019; Zügner and Günnemann 2019; Liu et al. 2023; Sun et al. 2020; Ma et al. 2021; Xu et al. 2022a). For example, Xu et al. (2019) introduces a topology attack framework based on edge perturbations from a first-order optimization perspective. This method adopts a convex relaxation to overcome the challenge of attacking discrete graph structure data. Meta-attack (Zügner and Günnemann 2019) also explores attacks on GNNs by perturbing the discrete graph structure. Its core principle involves using meta-gradients to solve the bi-level problem underlying training-time attacks, essentially treating the graph as a hyperparameter for optimization.

Defense method. There are several methods that have been proposed to enhance the robustness of GNNs (Xu et al. 2021, 2020, 2022c; Said, De Luca, and Albayrak 2010; Entezari et al. 2020; Jin et al. 2020; Chen et al. 2021). GCN-Jaccard (Said, De Luca, and Albayrak 2010) addresses this issue by eliminating edges between nodes with low Jaccard similarity. GNN-SVD (Entezari et al. 2020) proposes purifying the perturbed adjacency matrix by obtaining its low-rank approximation through truncated SVD. Pro-GNN (Jin et al. 2020) takes a different approach by jointly learning the clean graph structure and robust GNN parameters, while ensuring important graph properties such as low-rank, sparsity, and feature smoothness are maintained. Attention-based defense methods penalize the model's weights on adversarial edges or nodes, resulting in the development of new GNNs. For instance, median aggregation and trimmed mean aggregation (Chen et al. 2021) are proposed as robust aggregation functions in GNNs. In contrast to the aforementioned works, our research focuses on the optimal allocation of costs to nodes to achieve the most effective defense. Through effective cost allocation, our method achieves state-of-the-art robustness compared to existing approaches.

Conclusion

In this paper, we present RisKeeper, a novel and effective cost allocation method designed to defend against graph attacks. We introduce a fresh perspective that incorporates cost considerations into both attack and defense of GNNs. Specifically, we introduce a cost-aware PGD attack, and approach the cost allocation problem faced by defenders as the task of quantifying the value of nodes/edges in terms of adversarial risk. We formulate the problem as a bi-level optimization problem, representing a recursive game between defenders and attackers. We theoretically analyze the optimization problem and propose a robust training method to solve it. Through extensive experiments, we demonstrate that RisKeeper outperforms state-of-the-art defense methods, particularly when attackers have limited attack budget. Additionally, our learned cost allocation can help identify robust and vulnerable structures within graphs.

We contribute to the robust graph learning literature by introducing of a novel perspective and an effective method. This work opens up avenues for future research to explore other variations of attacks, such as targeted attacks. By expanding the scope of investigation, researchers can further enhance the understanding and development of robust graph learning techniques.

Acknowledgements

This work is partially supported by the National Natural Science Foundation of China (Grant No. 62206056, 72101007, 8200906871) and CCF-Tencent Rhino-Bird Open Research Fund.

References

Arghal, R.; Lei, E.; and Bidokhti, S. S. 2022. Robust graph neural networks via probabilistic lipschitz constraints. In *Proceedings of the Learning for Dynamics and Control Conference*, 1073–1085.

Boyd, S. P.; and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Chen, L.; Li, J.; Peng, Q.; Liu, Y.; Zheng, Z.; and Yang, C. 2021. Understanding Structural Vulnerability in Graph Convolutional Networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2249–2255.

Chen, P.-Y.; and Hero, A. O. 2015. Deep Community Detection. *IEEE Transactions on Signal Processing*, 21(63): 5706–5719.

Dai, E.; Zhao, T.; Zhu, H.; Xu, J.; Guo, Z.; Liu, H.; Tang, J.; and Wang, S. 2022. A Comprehensive Survey on Trustworthy Graph Neural Networks: Privacy, Robustness, Fairness, and Explainability. *arXiv e-prints*.

Dai, Q.; Shen, X.; Zhang, L.; Li, Q.; and Wang, D. 2019. Adversarial Training Methods for Network Embedding. In *Proceedings of the International World Wide Web Conference*, 329–339.

Dou, Y.; Liu, Z.; Sun, L.; Deng, Y.; Peng, H.; and Yu, P. S. 2020. Enhancing Graph Neural Network-Based Fraud Detectors against Camouflaged Fraudsters. In *Proceedings of the ACM International Conference on Information & Knowledge Management*, 315–324.

Entezari, N.; Al-Sayouri, S. A.; Darvishzadeh, A.; and Papalexakis, E. E. 2020. All You Need Is Low (Rank): Defending Against Adversarial Attacks on Graphs. In *Proceedings of the International Conference on Web Search and Data Mining*, 169–177.

Geisler, S.; Zügner, D.; and Günnemann, S. 2020. Reliable Graph Neural Networks via Robust Aggregation. In *Proceedings of the International Conference on Neural Information Processing Systems*, 13272–13284.

Hočevar, T.; and Demšar, J. 2014. A combinatorial approach to graphlet counting. *Bioinformatics*, 30(4): 559–565.

Jiang, M.; Li, Z.; Zhang, S.; Wang, S.; Wang, X.; Yuan, Q.; and Wei, Z. 2020. Drug–target affinity prediction using graph neural network and contact maps. *RSC advances*, 10(35): 20701–20712.

Jin, W.; Ma, Y.; Liu, X.; Tang, X.; Wang, S.; and Tang, J. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery & data mining*, 66–74.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations*.

Liu, Z.; Luo, Y.; Wu, L.; Liu, Z.; and Li, S. Z. 2023. Towards reasonable budget allocation in untargeted graph structure attacks via gradient debias. *arXiv preprint arXiv:2304.00010*.

Ma, Y.; Wang, S.; Derr, T.; Wu, L.; and Tang, J. 2021. Graph Adversarial Attack via Rewiring. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 1161–1169.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Mccallum, A. K.; Nigam, K.; Rennie, J.; and Seymore, K. 2000. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 3(2): 127–163.

Pržulj, N.; Corneil, D. G.; and Jurisica, I. 2004. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18): 3508–3515.

Said, A.; De Luca, E. W.; and Albayrak, S. 2010. How social relationships affect user similarities. In *Proceedings of the International Conference on Intelligent User Interfaces Workshop on Social Recommender Systems*.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

Song, H.; Kim, M.; Park, D.; Shin, Y.; and Lee, J.-G. 2023. Learning From Noisy Labels With Deep Neural Networks: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11): 8135–8153.

Sun, Y.; Wang, S.; Tang, X.; Hsieh, T.-Y.; and Honavar, V. 2020. Adversarial Attacks on Graph Neural Networks via Node Injections: A Hierarchical Reinforcement Learning Approach. In *Proceedings of the International World Wide Web Conference*, 673–683.

Waniek, M.; Michalak, T. P.; Wooldridge, M. J.; and Rahwan, T. 2018. Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2): 139–147.

Wu, H.; Wang, C.; Tyshetskiy, Y.; Docherty, A.; Lu, K.; and Zhu, L. 2019. Adversarial Examples for Graph Data: Deep Insights into Attack and Defense. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 4816–4823.

Wu, S.; Sun, F.; Zhang, W.; Xie, X.; and Cui, B. 2022. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5): 1–37.

Xu, J.; Sun, Y.; Jiang, X.; Wang, Y.; Wang, C.; Lu, J.; and Yang, Y. 2022a. Blindfolded attackers still threatening: Strict black-box adversarial attacks on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4299–4307.

Xu, J.; Yang, Y.; Chen, J.; Jiang, X.; Wang, C.; Lu, J.; and Sun, Y. 2022b. Unsupervised adversarially robust representation learning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4290–4298.

Xu, J.; Yang, Y.; Chen, J.; Jiang, X.; Wang, C.; Lu, J.; and Sun, Y. 2022c. Unsupervised adversarially robust representation learning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4290–4298.

Xu, J.; Yang, Y.; Pu, S.; Fu, Y.; Feng, J.; Jiang, W.; Lu, J.; and Wang, C. 2021. NetRL: Task-aware Network Denoising via Deep Reinforcement Learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(1): 810–823.

Xu, J.; Yang, Y.; Wang, C.; Liu, Z.; Zhang, J.; Chen, L.; and Lu, J. 2020. Robust network enhancement from flawed networks. *IEEE Transactions on Knowledge and Data Engineering*, 34(7): 3507–3520.

Xu, K.; Chen, H.; Liu, S.; Chen, P.-Y.; Weng, T.-W.; Hong, M.; and Lin, X. 2019. Topology attack and defense for graph neural networks: An optimization perspective. *arXiv preprint arXiv:1906.04214*.

Zhang, X.; and Zitnik, M. 2020. GNNGUARD: Defending Graph Neural Networks against Adversarial Attacks. In *Proceedings of the International Conference on Neural Information Processing Systems*, 9263–9275.

Zhang, Z.; Wang, J.; Ye, J.; and Wu, F. 2022. Rethinking graph convolutional networks in knowledge graph completion. In *Proceedings of the ACM Web Conference*, 798–807.

Zou, X.; Zheng, Q.; Dong, Y.; Guan, X.; Kharlamov, E.; Lu, J.; and Tang, J. 2021. Tdgia: Effective injection attacks on graph neural networks. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2461–2471.

Zügner, D.; and Günnemann, S. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. *ArXiv*.

Appendix A

Proof to Proposition 1
Given Feasible set $\mathcal{S} : \{\mathbf{s} \mid \mathbf{c}^T\mathbf{s} \leq \epsilon\}$,
and the objective of PGD:
given $\mathbf{a}$, an update from Gradient Descent,

$$\begin{aligned} \arg\min\nolimits_{\mathbf{s}} \quad & \frac{1}{2}||\mathbf{s}-\mathbf{a}||_2^2 \\ s.t. \quad & \mathbf{c}^T\mathbf{s} \leq \epsilon, \\ & \mathbf{s} \in [0,1]^n. \end{aligned} \tag{7}$$

which is equivalent as

$$\begin{aligned} \arg\min\nolimits_{\mathbf{s}} \quad & \frac{1}{2}||\mathbf{s}-\mathbf{a}||_2^2 + \mathcal{I}_{[0,1]}(\mathbf{s}) \\ s.t. \quad & \mathbf{c}^T\mathbf{s} \leq \epsilon. \end{aligned} \tag{8}$$

where

$$\mathcal{I}_{[0,1]}(s) = \begin{cases} 0, & \text{if } s \in [0,1], \\ \infty, & \text{else.} \end{cases}$$

Using Lagrange multiplier, the above is equivalent as

$$\begin{aligned} \arg\min\nolimits_{\mathbf{s}} M &= \frac{1}{2}||\mathbf{s}-\mathbf{a}||_2^2 + \mathcal{I}_{[0,1]}(\mathbf{s}) + \mu(\mathbf{c}^T\mathbf{s}-\epsilon) \\ &= \sum_i \left(\frac{1}{2}(s_i - a_i)^2 + \mathcal{I}_{[0,1]}(s_i) + \mu\mathbf{c}_i s_i\right) - \mu\epsilon. \end{aligned} \tag{9}$$

Denote

$$M_i = \frac{1}{2}(s_i - a_i)^2 + \mathcal{I}_{[0,1]}(s_i) + \mu\mathbf{c}_i s_i,$$

Take partial derivative of M with respect to s_i:

$$\frac{\partial M_i}{\partial s_i} = (s_i - a_i) + \frac{\partial \mathcal{I}_{[0,1]}(s_i)}{\partial s_i} + \mu\mathbf{c}_i.$$

Set partial derivative to 0, find root of s:

$$s_i = \mathrm{P}_{[0,1]}(a_i - \mu\mathbf{c}_i), \forall i,$$

or equivalently,

$$\mathbf{s} = \mathrm{P}_{[0,1]}(\mathbf{a} - \mu\mathbf{c}).$$

is the stationary constraint.
The KKT constraint is:

$$\begin{cases} \mu(\mathbf{c}^T\mathbf{s}-\epsilon) = 0, \\ \mu \geq 0, \\ \mathbf{c}^T\mathbf{s} - \epsilon \leq 0. \end{cases} \tag{10}$$

Solving KKT constraint:
When $\mu = 0$, we have $\mathbf{c}^T\mathbf{s} \leq 0 \Rightarrow \mathbf{c}^T\mathrm{P}_{[0,1]}(\mathbf{a}) \leq \epsilon$,
When $\mu > 0$, we have $\mathbf{c}^T\mathbf{s} = 0 \Rightarrow \mathbf{c}^T\mathrm{P}_{[0,1]}(\mathbf{a} - \mu\mathbf{c}) = \epsilon$.

Combining stationary and KKT constraint, we obtain the closed-form solution:

$$\Pi_{\mathcal{S}}(\mathbf{a}) = \begin{cases} \mathrm{P}_{[0,1]}(\mathbf{a} - \mu\mathbf{c}), & \text{if} \quad \mu > 0 \quad \text{and} \\ & \mathbf{c}^T\mathrm{P}_{[0,1]}(\mathbf{a} - \mu\mathbf{c}) = \epsilon, \\ \mathrm{P}_{[0,1]}(\mathbf{a}), & \text{if} \quad \mathbf{c}^T\mathrm{P}_{[0,1]}(\mathbf{a}) \leq \epsilon. \end{cases}$$

Proof done.

Appendix B

Sensitivity Test. Figure 2 shows the relationship between hyperparameter λ and test accuracy of our defense method under different cost constraint ratio on Cora dataset. The same settings as described in Experimental Setup section are used. λ (relative) $= 1$ in the graph marks the original value of λ as described in Experimental Setup section.

Results confirmed that when λ, *i.e.*, attackers' sensitivity towards attacking costs, rises, the performance of RisKeeper increases as well.

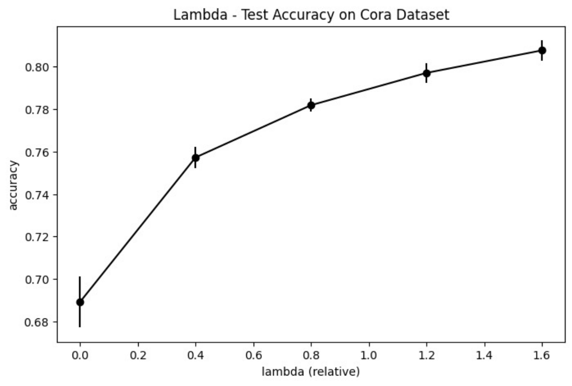

Figure 2: λ (relative ratio) - test accuracy of RisKeeper under cost-aware PGD attack with cost constraint $0.08|E|$.