How Robust Are Graph Anomaly Detectors, and at What Cost?

Anonymous authors
Paper under double-blind review

Abstract

Graph anomaly detection (GAD) aims to identify rare or suspicious entities from their attributes and relational patterns, with applications in fraud detection, risk control, and network security. Existing evaluations, however, largely focus on detection performance on clean graphs, overlooking that anomalous actors may strategically modify these attributes or relations to evade detection. This raises two fundamental questions: how robust are different GAD mechanisms to such modifications, and how much modification is required for successful evasion? We introduce a cost-aware robustness evaluation framework that jointly characterises target escape and graph modification cost. To quantify modification burden across perturbation channels, we develop a detector-independent, normality-calibrated cost that places attribute changes, edge additions, and edge deletions on a common reference scale. Across diverse GAD mechanisms, attacks, and graphs, we find that robustness is strongly channel-dependent, target escape can diverge substantially from global ranking degradation, and attack effectiveness does not reliably reflect modification cost. We further connect these findings through the relationship among graph modification, anomaly evidence, and decision margin, providing a unified view of how modifications lead to successful evasion. This analysis motivates a design principle for economically robust GADs: under a false-positive constraint, detection should rely on evidence that cannot be readily neutralised by low-cost modifications, with multi-channel complementarity providing one potential route toward this goal. Code is available at an anonymous repository: `https://anonymous.4open.science/r/GADRobustness-B9AA`.

1 Introduction

Graph anomaly detection (GAD) identifies rare nodes that deviate from normal patterns (Qiao et al., 2025), with applications in fraud detection, financial risk control, and network security. In such applications, anomalous actors may have incentives to alter their observable attributes or interactions to appear less anomalous and evade detection (Lai et al., 2024). Consequently, strong performance on a clean graph does not necessarily indicate how a detector will behave when anomalies actively attempt to evade detection.

Prior work on graph adversarial attacks has extensively studied structural and feature perturbations in node classification (Zügner et al., 2018; Chen et al., 2018; Zügner & Günnemann, 2019; Jain & Dutta, 2024). Recent studies further show that graph anomaly detectors can be compromised by adversarial modifications (Zhu et al., 2022; Lai et al., 2024), while clean-data benchmarks compare diverse GAD mechanisms under non-adversarial conditions (Tang et al., 2023; Zhou et al., 2026b). However, these studies provide only a partial view of GAD robustness: it remains unclear how GAD mechanisms respond under shared attack conditions and how much graph modification is required for successful evasion. A systematic evaluation therefore needs to consider both adversarial effectiveness and modification burden across detectors, perturbation channels, and graphs.

To address this gap, we formulate GAD robustness along two complementary dimensions: whether a previously detected anomaly successfully escapes detection and how much graph modification is required to achieve that escape. Unlike classification attacks defined by label flips, GAD evasion is characterised by whether a captured target crosses the detector's anomaly decision threshold. We therefore use target escape to quantify adversarial effectiveness and introduce a detector-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

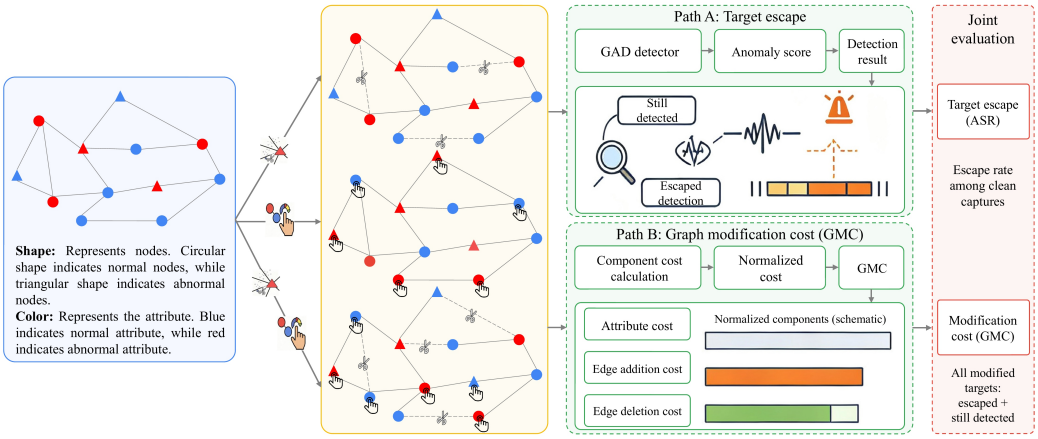


Figure 1: Conceptual overview of the proposed evaluation framework.

independent, normality-calibrated graph modification cost to quantify modification burden. This cost places attribute changes, edge additions, and edge deletions on a common reference scale, enabling adversarial effectiveness and modification burden to be analysed jointly.

We instantiate this framework through a unified *detector* $\times$ *attack* $\times$ *dataset* evaluation spanning ten GAD baselines, six synthetic, injected, and natively labelled graphs, and structural, feature, and hybrid perturbations. We analyse clean-performance sensitivity on six graphs and conduct test-time attacks on a fixed evaluation matrix over five graphs, with detector parameters frozen. Jointly measuring target escape and modification cost enables comparison of how different GAD mechanisms respond to different perturbation channels and how their adversarial vulnerability relates to the burden of modifying the underlying graph, as illustrated in Figure 1.

Our main contributions are as follows:

- We formulate a cost-aware robustness framework for graph anomaly detection that jointly characterises target escape and modification burden, distinguishing targeted evasion from global performance degradation.
- We introduce a detector-independent, normality-calibrated graph modification cost that places attribute changes, edge additions, and edge deletions on a common reference scale, enabling comparison across perturbation channels and graphs.
- We conduct a systematic evaluation across diverse GAD mechanisms, graphs, and perturbation channels, revealing channel-dependent robustness and distinct relationships among target escape, global ranking degradation, and modification burden. These findings motivate an economic-defence principle favouring evidence resistant to low-cost modifications.

2 Related Work

2.1 Graph Anomaly Detection

Deep graph anomaly detection (GAD) spans diverse methodological families (Qiao et al., 2025), including reconstruction-based methods (Ding et al., 2019; Fan et al., 2020; Roy et al., 2024), one-class and contrastive approaches (Wang et al., 2021; Liu et al., 2021), and spectral, knowledge-distillation, and training-free methods (Gao et al., 2023; Xu et al., 2026; Zhao et al., 2025; Liu et al., 2026b). Unified frameworks further extend GAD across multiple anomaly types and task settings (Lin et al., 2024). Existing benchmarks primarily evaluate these methods on clean graphs (Tang et al., 2023), deployment challenges such as scale, anomaly scarcity, and missing attributes (Zhou et al., 2026b), and the competitiveness of simple baselines (Pan et al., 2023).

Adversarial robustness has been studied more extensively in other graph learning tasks, including node classification, link prediction, and general GNN evaluation (Zheng et al., 2021; Mujkanovic

et al., 2022; Wu et al., 2025b; Zhou et al., 2026a; Ngo et al., 2026). Related studies consider test-time evasion (Wu et al., 2026), application-specific reliability (Wu et al., 2025a), and robustness formulations for anomaly and outlier detection (Karunanayake et al., 2025). Despite these advances, systematic cross-mechanism evaluation of GAD under shared evasion conditions remains limited, particularly when attack effectiveness and modification burden are considered jointly.

2.2 Adversarial Attacks on Graphs

Graph adversarial attacks have been extensively studied for node classification, including targeted structural and feature perturbations, gradient-guided attacks, and poisoning (Zügner et al., 2018; Chen et al., 2018; Dai et al., 2018; Zügner & Günnemann, 2019). Subsequent work extends these attacks to node injection and exploits spectral or message-passing mechanisms (Sun et al., 2020; Tao et al., 2021; Dai et al., 2023; Kohli & Sikdar, 2026). These studies establish a broad attack methodology for graph learning, but primarily target discriminative prediction tasks.

Adversarial attacks on graph anomaly detection remain more fragmented. Existing studies consider structural poisoning of GAD models (Zhu et al., 2022; 2024), coupled-space attacks on random-walk detectors (Lai et al., 2024), budgeted indirect evasion in temporal sensor networks (Xaviar & Ardakanian, 2025), and multi-target injection for supervised fraud classification (Choi et al., 2025). While these studies demonstrate vulnerabilities in specific settings, they differ substantially in attack stage, detector type, task formulation, and perturbation space. Consequently, they provide limited evidence for comparing different GAD mechanisms under shared test-time attacks, particularly across structural, feature, and hybrid perturbations.

3 Preliminaries

3.1 Problem Formulation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ denote an attributed graph, where $|\mathcal{V}| = n$, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, and $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ is the node feature matrix. A graph anomaly detector $\mathcal{D}$ assigns each node $v \in \mathcal{V}$ an anomaly score $Q_{\mathcal{D}}(\mathcal{G}, v) \in \mathbb{R}$, where a higher score indicates greater anomalousness. Given a detection threshold $\tau_{\mathcal{D}}(\mathcal{G})$, node v is detected as anomalous if $Q_{\mathcal{D}}(\mathcal{G}, v) > \tau_{\mathcal{D}}(\mathcal{G})$.

We consider test-time evasion attacks in which an attacker $\mathcal{A}$ modifies $\mathcal{G}$ into $\mathcal{G}'$ under a perturbation budget b, while the detector parameters remain frozen. For a target node v detected on the clean graph, evasion succeeds if $Q_{\mathcal{D}}(\mathcal{G}', v) \leq \tau_{\mathcal{D}}(\mathcal{G}')$. The threshold $\tau_{\mathcal{D}}(\mathcal{G}')$ is determined per graph from the score quantile of a fixed set of normal reference nodes, allowing the decision boundary to account for score shifts induced by graph modification. Our objective is therefore to characterise not only whether a captured anomaly escapes detection, but also the modification burden required to achieve that escape.

3.2 Threat Model and Attacks

We consider test-time evasion, where the attacker observes the graph structure and node attributes and knows the target set $\mathcal{T}$, consisting of anomalous test nodes. The attacker may modify each target's attributes and its incident undirected edges, with multi-target perturbations applied to a shared graph. Detector parameters remain frozen throughout the attack. Since existing GAD attacks are typically designed for specific detectors or scenarios (Zhu et al., 2022; 2024; Lai et al., 2024; Xaviar & Ardakanian, 2025), we adapt representative graph attacks into a common evasion setting that can be applied across detection mechanisms. The attacks span structural, feature, and hybrid perturbations. DICE and FGA use full-graph labels, the latter to train its classification surrogate. NFF and Nettack-GAD use no labels; Greedy uses the known targets and detector score/gradient access. Appendix A details these privileges.

Structural attacks. Structural attacks modify target-incident edges while preserving node attributes. DICE (Delete Internally, Connect Externally) (Waniek et al., 2018) randomly deletes edges between same-label nodes or adds edges between different-label nodes. FGA (Fast Gradient Attack) (Chen et al., 2018) performs gradient ascent on a continuously relaxed adjacency matrix using a surrogate model and flips one target-incident edge at each step. Both use a target-specific budget

$b(v) = \max(1, \mathrm{round}(\rho \deg(v)))$ with $\rho = 0.2$, where $\deg(v)$ is the graph degree when target v is processed.

Feature attack. NFF (Neighbourhood Feature Fusion) modifies only target attributes by moving them towards the mean of the target's two-hop neighbourhood $N_{\leq 2}(v)$, excluding the target itself. Specifically, $\boldsymbol{X}_v \leftarrow (1-\alpha)\boldsymbol{X}_v + \alpha \overline{\boldsymbol{X}}_{N_{\leq 2}(v)}$ with $\alpha = 1.0$. NFF uses only graph structure and attributes and requires no model queries.

Hybrid attacks. Hybrid attacks allow both structural and feature modifications. We consider two complementary strategies: a detector-aware greedy attack that directly optimises the tested detector's score, and Nettack-GAD, a surrogate-based adaptation of Nettack (Zügner et al., 2018) to the anomaly detection setting.

Greedy uses the tested detector's anomaly score as an oracle and, at each step, selects the candidate action that maximally decreases a target's anomaly score:

$$a^* = \arg \max_{v \in \mathcal{T},\, a \in \mathcal{C}(v)} \left[Q_{\mathcal{D}}(\mathcal{G}_t, v) - Q_{\mathcal{D}}(\mathcal{G}_t \oplus a, v) \right], \tag{1}$$

where $\mathcal{C}(v)$ denotes the structural and attribute candidate set for target v, and $\mathcal{G}_t$ is the graph after accumulated modifications.

Nettack-GAD instead uses a frozen reconstruction surrogate to guide perturbation selection without directly optimising the tested detector. It adapts Nettack's incremental scoring and candidate filtering to a dual-channel setting and replaces its classification objective with minimisation of the target's anomaly score. Further details on candidate construction, budget allocation, and attack implementation are provided in Appendix A.

4 A Two-Dimensional Robustness Test

We characterise GAD robustness along two complementary dimensions: *target escape* and *graph modification cost*. The first measures whether anomalies captured on the clean graph escape detection after perturbation, characterising the effectiveness of an attack. The second quantifies the modification burden imposed on the graph to carry out the attack. Both dimensions evaluate corresponding attack conditions, providing a joint effectiveness-cost view of adversarial robustness.

4.1 Dimension I: Target Escape

The first dimension measures whether anomaly targets captured on the clean graph escape detection after perturbation. Let $\mathcal{N}$ be the fixed set of normal reference nodes determined by ground-truth labels in the test set. The evaluator computes the $1-\eta$ quantile of their scores on the clean and perturbed graphs, respectively:

$$\tau_{\mathcal{D}}(G) = \mathrm{Quantile}_{1-\eta}\{Q_{\mathcal{D}}(G, u) : u \in \mathcal{N}\}, \qquad \eta = 0.05. \tag{2}$$

This per-graph relative threshold compares detection outcomes under the same nominal false-positive level. The reference node set remains fixed, while its scores are recomputed after graph modification. Labels define the evaluator's normal reference set and additionally form part of the DICE/FGA attacker privileges (Section 3.2).

Let $\mathcal{T}_c = \{v \in \mathcal{T} : Q_{\mathcal{D}}(\mathcal{G}, v) > \tau_{\mathcal{D}}(\mathcal{G})\}$ be the set of targets captured on the clean graph. The attack success rate (ASR) is defined as:

$$\mathrm{ASR} = \frac{|\{v \in \mathcal{T}_c : Q_{\mathcal{D}}(\mathcal{G}', v) \leq \tau_{\mathcal{D}}(\mathcal{G}')\}|}{|\mathcal{T}_c|}, \qquad |\mathcal{T}_c| > 0. \tag{3}$$

An empty captured set has no valid denominator and is therefore excluded from ASR calculation. ASR measures the fraction of clean-graph captures lost after perturbation: escape can result from a decrease in the target score, a shift in the normal-reference threshold, or both.

To characterise the accompanying change in global test-set ranking, we additionally report:

$$\Delta\mathrm{AUC} = \mathrm{AUC}_{\mathrm{clean}} - \mathrm{AUC}_{\mathrm{attacked}}. \tag{4}$$

Positive values indicate performance degradation after attack. Unlike ASR, $\Delta\mathrm{AUC}$ is a global ranking measure and may change little when an attack successfully evades only a small number of targeted anomalies. We therefore use ASR as the primary measure of targeted evasion and $\Delta\mathrm{AUC}$ as a complementary measure of global impact.

4.2 Dimension II: Graph Modification Cost

Similar escape rates can arise from substantially different graph modifications. We therefore introduce Graph Modification Cost (GMC), denoted by C_Σ, to quantify the modification burden associated with an attack. GMC is a detector-independent, normality-calibrated measure derived from the clean graph: attribute changes are assessed relative to normal feature variation, while edge additions and deletions are assessed according to their compatibility with normal relational patterns. All pricing parameters are estimated once from normal nodes and their relations in the clean graph and then frozen, so different detectors evaluated on the same graph share the same cost scale.

GMC is designed around three principles. *Detector independence* ensures that cost does not depend on the detector being attacked, enabling comparison across detection mechanisms. *Data-grounded interpretability* derives each cost component from observable graph statistics. *Cross-channel comparability* normalises attribute modifications, edge additions, and edge deletions against corresponding reference events, allowing heterogeneous modifications to be expressed on a common scale.

Attribute modification cost. Attribute modifications are measured relative to the normal variation of each feature dimension. For dimension j, we define the robust reference scale as $\sigma_j = \max(1.4826\,\mathrm{MAD}_j, \varepsilon_\sigma)$, where MAD_j is the median absolute deviation estimated from the normal calibration nodes, 1.4826 is the standard normal-consistency factor for MAD, and ε_σ is a numerical floor. For a net attribute modification Δ_j, the attribute cost is:

$$C_{\mathrm{attr}} = \sum_j \log\left(1 + \frac{|\Delta_j|}{\sigma_j}\right). \tag{5}$$

Thus, a modification of one σ_j contributes $\log 2$, placing different feature dimensions on a relative scale while reducing the influence of exceptionally large changes.

Edge modification cost. Edge modifications are priced according to a connection plausibility score $p_(a, b) \in (0, 1)$ estimated from the clean normal graph. We construct this score using attribute distance, neighbourhood structural distance, and log-degree difference between node pairs. A regularised least-squares model is fitted on balanced samples of normal edges and non-edges, and its output is mapped through a sigmoid with numerical clipping to obtain p_+; full construction details are provided in Appendix B. We define $p_-(a, b) = 1 - p_+(a, b)$ and convert these scores into surprisal-based quantities:

$$R_\pm(a, b) = -\log p_\pm(a, b), \qquad c_\pm(a, b) = \log\left(1 + \frac{R_\pm(a, b)}{M_\pm}\right), \tag{6}$$

where $M_\pm$ denotes the corresponding median reference surprisal with a numerical floor.

The total costs of edge additions and deletions are then $C_+ = \sum c_+$ and $C_- = \sum c_-$, respectively. This construction assigns greater addition cost to less plausible new relations and greater deletion cost to relations whose removal is less compatible with normal relational patterns.

Cross-channel normalisation. Because attribute and edge costs are defined on different raw scales, we normalise each component using the expected cost of a corresponding reference event. The attribute reference event is an independently sampled full-dimensional attribute perturbation, while edge reference events are sampled according to normal-graph plausibility, with additions weighted by p_+ and deletions by $1 - p_+$. Their baseline costs, denoted B_{attr}, B_+, and B_-, provide a common reference across modification channels; detailed sampling and baseline definitions are given in Appendix B. The final GMC is:

$$C_\Sigma = \frac{C_{\mathrm{attr}}}{B_{\mathrm{attr}}} + \frac{C_+}{B_+} + \frac{C_-}{B_-}. \tag{7}$$

Under this equal-weight additive construction, one reference event on each axis contributes one comparable cost unit. GMC measures the net modification of the final attacked graph relative to the clean graph. Intermediate reversals or repeated actions therefore do not accumulate additional cost under terminal-state differencing. GMC should be interpreted as a normality-calibrated graph modification burden rather than a direct monetary or operational cost; application-specific operational costs can be incorporated when such information is available.

4.3 Hyperparameter Sensitivity

A detector's clean detection performance may vary across hyperparameter configurations. For the tuning configuration set Γ_m of detector m (see Appendix A for tuning details), we quantify hyperparameter sensitivity by the range of calibrated AUC:

$$S_m = \max_{\gamma \in \Gamma_m} \mathrm{AUC}_{\mathrm{calib}}(\gamma) - \min_{\gamma \in \Gamma_m} \mathrm{AUC}_{\mathrm{calib}}(\gamma). \tag{8}$$

A larger S_m indicates greater variation in clean detection performance across the considered hyperparameter configurations.

4.4 Evaluation Statistics

The two robustness dimensions are summarised over different target sets. Let $\mathcal{T}_m$ denote the set of targets that are actually modified by an attack, where a target belongs to $\mathcal{T}_m$ if it has a nonzero net attribute change or any net addition or deletion of its incident edges. We report the median C_Σ over $\mathcal{T}_m$, together with the shares of modified targets satisfying $C_\Sigma \leq 3$ and $C_\Sigma \leq 1$, which we use as descriptive thresholds for low- and very-low-cost modifications, respectively.

ASR uses the clean captured set $\mathcal{T}_c$ as its denominator, since it measures how many previously detected anomalies escape. In contrast, GMC is summarised over $\mathcal{T}_m$, including both modified targets that escape and those that remain detected. All GMC pricing parameters are estimated from the clean graph and frozen before attack, ensuring that the same modification is evaluated under the same cost scale.

5 Experiments

5.1 Experimental Setup

We evaluate ten baseline detectors spanning reconstruction, one-class learning, training-free scoring, heterophily-guided, and spectral filtering mechanisms (Ding et al., 2019; Wang et al., 2021; Zhao et al., 2025; Liu et al., 2026b; Zhou et al., 2025; Pan et al., 2025; Liu et al., 2026a; Dong et al., 2025; Fu et al., 2026; Zheng et al., 2026). The six datasets cover synthetic, injected, and natively labelled graphs: Gen-1000 is synthetic, Inj-Cora and Inj-Amazon are injected, while Reddit, YelpChi-5K, and Weibo contain native anomaly labels. They range from 1,000 to 13,752 nodes and from 2,468 to 377,271 edges. Detailed detector families and dataset statistics are provided in Appendix Tables 8 and 7, respectively.

Attack evaluation uses a fixed subset of seed-0 detector instances, constructed with a full-graph clean-AUC screen and a predefined detector roster. This matrix is separate from the three-seed calibration-set sensitivity analysis; Appendix A.5 details its construction. We exclude the separately executed Inj-Cora TFGAD–Greedy condition from comparisons and aggregates because its clean captured set differs from the other attacks. All attacks are performed at test time with detector parameters frozen.

5.2 Robustness under Targeted Evasion

Table 1 reports ASR and $\Delta\mathrm{AUC}$ for all 83 included combinations, while Table 2 highlights the Inj-Cora subset. Substantial target escape occurs even without direct access to the tested detector. On Inj-Cora, NFF against DR-GGAD and Nettack-GAD against TFGAD both achieve an ASR of 1.000, despite neither attack querying the tested detector (Table 2). At the same time, robustness varies considerably across perturbation channels. On Weibo, 10 of 15 paired detector comparisons reverse

Table 1: Target-escape evaluation across five datasets. Each cell reports ASR, with $\Delta\mathrm{AUC}$ in parentheses; "–" denotes computationally unavailable combinations. †: the separate TFGAD–Greedy run has a different clean captured set and is excluded from comparisons and aggregates (Appendix A.5).

Dataset	Detector	DICE	FGA	NFF	Greedy	Nettack-GAD
Gen-1000	DR-GGAD	0.500 (−0.034)	0.500 (−0.014)	1.000 (+0.788)	1.000 (+0.311)	1.000 (+0.412)
	HUGE	0.500 (+0.002)	0.000 (−0.024)	1.000 (+0.830)	1.000 (+0.680)	1.000 (+0.535)
	SAGAD	0.250 (−0.009)	0.000 (−0.088)	0.750 (+0.414)	0.500 (+0.353)	1.000 (+0.370)
Inj-Cora	DOMINANT	0.071 (−0.072)	0.000 (−0.079)	0.393 (+0.091)	0.036 (+0.023)	0.929 (+0.404)
	DR-GGAD	0.133 (−0.024)	0.100 (−0.037)	1.000 (+0.521)	0.467 (+0.075)	0.800 (+0.127)
	HIMO-GAD	0.364 (−0.044)	0.182 (−0.016)	0.909 (+0.396)	0.909 (+0.077)	0.818 (+0.117)
	HUGE	0.103 (−0.073)	0.000 (−0.080)	0.552 (+0.208)	0.172 (+0.073)	0.345 (+0.019)
	TFGAD	0.000 (+0.002)	0.000 (−0.084)	0.636 (+0.431)	–†	1.000 (+0.457)
Inj-Amazon	DOMINANT	0.036 (−0.017)	–	0.073 (+0.012)	–	–
	DR-GGAD	0.039 (−0.028)	–	1.000 (+0.652)	–	–
	HIMO-GAD	0.005 (−0.022)	–	0.968 (+0.445)	–	–
	HUGE	0.013 (−0.023)	–	0.847 (+0.259)	–	–
	LightGAD	0.000 (0.000)	–	0.000 (0.000)	–	–
	SAGAD	0.099 (+0.007)	–	0.960 (+0.732)	–	–
	SmoothGNN	0.000 (−0.054)	–	0.988 (+0.660)	–	–
	TFGAD	0.040 (−0.003)	–	0.000 (+0.172)	–	–
YelpChi-5K	LightGAD	0.429 (+0.017)	0.540 (−0.021)	0.841 (+0.044)	0.159 (+0.021)	0.730 (+0.090)
	SAGAD	0.022 (+0.001)	0.000 (−0.001)	1.000 (+0.157)	0.124 (+0.024)	0.247 (+0.019)
Weibo	DOMINANT	0.058 (−0.008)	–	0.453 (+0.104)	–	0.058 (−0.006)
	LightGAD	0.184 (+0.008)	–	0.737 (+0.097)	–	0.158 (+0.012)
	OCGNN	0.060 (−0.003)	–	0.119 (+0.030)	–	0.083 (+0.015)
	SAGAD	0.105 (+0.006)	–	0.073 (+0.004)	–	0.129 (+0.012)
	SmoothGNN	0.052 (+0.004)	–	0.409 (+0.075)	–	0.052 (0.000)
	TFGAD	0.000 (0.000)	–	0.895 (+0.305)	–	0.000 (0.000)

Table 2: Target-escape evaluation on Inj-Cora. Cells report ASR ($\Delta\mathrm{AUC}$). †: excluded clean-state mismatch; see Table 1.

Detector	DICE	FGA	NFF	Greedy	Nettack-GAD
DOMINANT	0.071 (−0.072)	0.000 (−0.079)	0.393 (+0.091)	0.036 (+0.023)	0.929 (+0.404)
DR-GGAD	0.133 (−0.024)	0.100 (−0.037)	1.000 (+0.521)	0.467 (+0.075)	0.800 (+0.127)
HIMO-GAD	0.364 (−0.044)	0.182 (−0.016)	0.909 (+0.396)	0.909 (+0.077)	0.818 (+0.117)
HUGE	0.103 (−0.073)	0.000 (−0.080)	0.552 (+0.208)	0.172 (+0.073)	0.345 (+0.019)
TFGAD	0.000 (+0.002)	0.000 (−0.084)	0.636 (+0.431)	–†	1.000 (+0.457)

Table 3: Macro-average ASR by attack across five datasets. Each included detector receives equal weight; "–" denotes computationally unavailable combinations.

Attack	Channel	Inj-Cora	Gen-1000	Inj-Amazon	YelpChi-5K	Weibo
FGA	structural	0.056	0.167	–	0.270	–
DICE	structural	0.134	0.417	0.029	0.226	0.076
Greedy	hybrid (adaptive)	0.396	0.833	–	0.141	–
NFF	feature	0.698	0.917	0.604	0.921	0.448
Nettack-GAD	hybrid (surrogate)	0.778	1.000	–	0.489	0.080

between DICE and NFF, and the attack with the highest macro-average ASR varies across datasets (Table 3 and Appendix Tables 12 and 13). These results indicate that substantial evasion can arise under different attack information privileges and that robustness observed under one perturbation channel does not necessarily transfer to another.

Target escape also does not necessarily coincide with global ranking degradation. Among the 55 evaluated combinations on Gen-1000, Inj-Cora, and Inj-Amazon, 45 exhibit target escape, of which 14 have $\Delta\mathrm{AUC} \leq 0$. Inj-Amazon×TFGAD×NFF further yields an ASR of 0.000 despite $\Delta\mathrm{AUC} = +0.172$. Thus, ASR and $\Delta\mathrm{AUC}$ capture distinct effects of an attack: the former measures whether previously detected targets escape, whereas the latter reflects changes in global test-set ranking.

We therefore use ASR as the primary measure of target-level evasion and $\Delta\mathrm{AUC}$ as a complementary measure of global impact. Detector comparisons are made within each dataset because detectors may have different clean captured sets and decision thresholds. Appendix Table 14 reports the corresponding pooled statistics, where the macro average weights each evaluated combination equally and the micro average weights by clean captured count.

5.3 Modification Cost and Evasion Effectiveness

Attack effectiveness does not determine modification cost. NFF achieves the highest macro-average ASR on YelpChi-5K, Weibo, and Inj-Amazon, yet its median GMC ranges from 1.88–20.19 across the five graphs and exceeds the low-cost threshold on every graph except Gen-1000. Conversely, DICE never exceeds 0.417 macro-average ASR, while its median GMC spans 1.48–23.95. Thus, neither attack effectiveness nor the attack label alone determines the modification burden: the same attack can incur substantially different costs across graphs. Figure 2 visualises the joint relationship between ASR and GMC, while Table 4 reports graph-level costs shared across detectors. Greedy is the exception because it directly queries the tested detector, causing its modifications and GMC to vary across detectors.

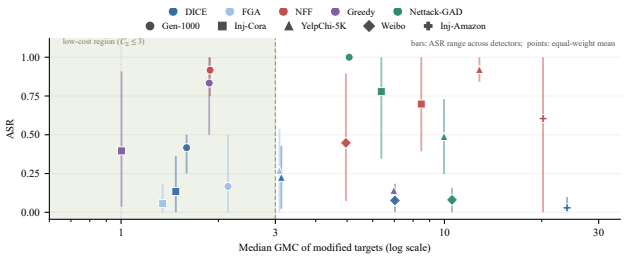


Figure 2: Joint attack effectiveness and graph modification cost across five datasets. Points show macro-average ASR and median GMC, with vertical bars indicating the ASR range across detectors. The shaded region marks GMC ≤ 3, and the dotted line marks one reference-event unit.

Modification cost is also informative only when considered together with target escape. Of the 83 evaluated combinations, 70 exhibit target escape, but only 20 combine escape with a median GMC in the low-cost region. On Inj-Cora, for example, Greedy modifies 8–10 of the 66 targets, while its ASR ranges from 0.04–0.91 across detectors (Appendix Table 20). The same distinction appears across anomaly types: under NFF on Inj-Amazon, attribute anomalies escape more frequently than structural anomalies (0.891 versus 0.588), despite having nearly identical median GMC values (Appendix Table 22). These results show that modification burden alone does not determine evasion: robustness depends on whether a given modification is able to neutralise the evidence on which a detector relies.

Taken together, target escape and GMC provide complementary views of adversarial robustness: ASR characterises whether modifications succeed, whereas GMC characterises the burden required to produce them. Per-detector costs are reported in Appendix Tables 23 and 24. Section 5.4 builds on this distinction to examine the implications for economically robust GAD design.

Table 4: Median GMC across datasets and attacks, reported in reference-event units. Greedy reports the range across included detectors; "–" denotes computationally unavailable combinations.

Dataset	DICE	FGA	NFF	Greedy	Nettack-GAD
Gen-1000	1.59	2.14	1.88	1.85–1.88	5.08
Inj-Cora	1.48	1.34	8.48	0.78–1.09	6.38
Inj-Amazon	23.95	–	20.19	–	–
YelpChi-5K	3.13	3.09	12.83	4.33–8.07	9.97
Weibo	7.03	–	4.96	–	10.56

5.4 Implications for Economic Defense

Whether a modification causes escape depends on how it affects the evidence used by the detector. Different GAD mechanisms derive anomaly evidence through reconstruction, neighbourhood aggregation, spectral filtering, and other signals, so vulnerability cannot be attributed to the structural or attribute channel alone. A modification becomes consequential only when its effect on this evidence is sufficient to change the final detection decision.

To characterise this response, fix the detector parameters θ, let $Q_\theta(G, v)$ denote the target score and $\tau_\theta(G)$ the per-graph threshold, and define the clean-minus-perturbed changes as ΔQ_v and $\Delta\tau$. The corresponding margin compression is: $\Delta m_v = \Delta Q_v - \Delta\tau$.

A captured target starts with margin $m_v(G) = Q_\theta(G, v) - \tau_\theta(G) > 0$ and escapes when its perturbed margin reaches zero or below. Positive Δm_v therefore indicates movement toward the decision boundary. Importantly, the target score and threshold may move differently because the threshold is recomputed from the same normal reference nodes after graph modification.

Table 5 illustrates this decomposition for five independently reproduced Inj-Cora conditions. Under NFF, for example, DR-GGAD loses 0.5612 in target score while its threshold increases by 0.0178,

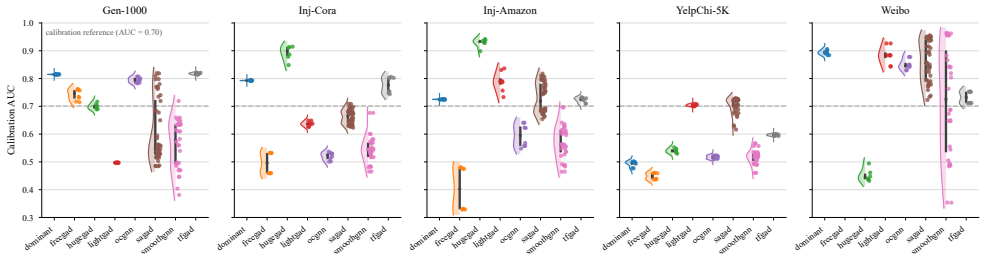


Figure 3: Hyperparameter sensitivity across five datasets. Half-violins show calibrated AUC distributions across configurations, and the dashed line marks the calibration reference of AUC 0.70. Zero-shot detectors are omitted.

yielding a margin compression of 0.5790. In contrast, DOMINANT under DICE and Greedy exhibits a negative mean margin compression even though some individual targets escape, showing that an average margin response cannot replace target-level threshold crossing (Appendix Table 26).

Together with the GMC analysis, this suggests an economic-defence principle: under a false-positive constraint, a robust detector should preserve clean detection while relying on anomaly evidence that cannot be readily neutralised through low-cost graph modifications. Evidence selection and complementary signals across modification channels provide potential routes towards this goal, but their combination must be calibrated on normal data and evaluated against adaptive attacks.

Table 5: Decision-margin decomposition for five Inj-Cora detector–attack conditions. Values are averaged over clean-captured targets; positive Δm indicates margin compression.

Detector	Attack	Escaped/Captured	ΔQ	$\Delta\tau$	Δm
DOMINANT	DICE	2/28	-0.03237	-0.00132	-0.03105
DOMINANT	NFF	11/28	$+0.11494$	$+0.00002$	$+0.11492$
DOMINANT	Greedy	1/28	-0.04558	0.00000	-0.04558
DR-GGAD	DICE	4/30	$+0.00953$	-0.00041	$+0.00994$
DR-GGAD	NFF	30/30	$+0.56122$	-0.01780	$+0.57902$

5.5 Sensitivity Analysis

Clean detection performance can vary substantially across hyperparameter configurations. Across eight tunable detectors and six datasets, 267 of 756 configurations (35.3%) meet the calibration reference threshold of AUC 0.70, averaged over three seeds. This sensitivity analysis does not define attack-matrix membership. Figure 3 shows the five attack-evaluated datasets; Reddit is omitted because no configuration meets the reference threshold. Table 9 reports the AUC range S_m and calibration-threshold pass rate.

6 Conclusion and Limitations

Graph anomaly detection is widely used to identify suspicious entities from their attributes and relational patterns, yet anomalous actors may modify such evidence to evade detection. Clean performance alone therefore provides an incomplete view of robustness, while attack success does not capture the modification burden required for evasion. We address this gap by characterising GAD robustness through *target escape* and *graph modification cost*, introducing a detector-independent, normality-calibrated GMC that places heterogeneous graph modifications on a common reference scale. Our evaluation shows that robustness is strongly channel-dependent, target escape can diverge from global ranking degradation, and modification burden does not necessarily track attack effectiveness. Together with our decision-margin analysis, these findings motivate an economic-defence principle: robust GAD should rely on evidence that cannot be readily neutralised through low-cost modifications, making successful evasion not only difficult, but costly.

Our study focuses on test-time evasion under representative structural, feature, and hybrid perturbations. Broader attack spaces, larger graphs, and adaptive attacker–defender settings provide natural extensions. Future work may also investigate complementary evidence across multiple channels under the proposed cost-aware framework.

AI USE STATEMENT

Generative AI tools were used solely for language editing and polishing the manuscript. They were not used to develop the methodology, conduct experiments, analyse results, or generate scientific conclusions. All AI-assisted edits were reviewed and verified by the authors, who take full responsibility for the final manuscript.

ETHICS STATEMENT

This work studies adversarial robustness in graph anomaly detection with the goal of understanding potential vulnerabilities and informing more robust detector design. Although the evaluated attacks demonstrate how graph modifications can facilitate evasion, all experiments are conducted in controlled research settings using established benchmark datasets. We do not target deployed systems or provide access to private or sensitive data. We encourage the use of these findings for robustness evaluation and defensive research.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the evaluation protocol, attack-matrix construction and clean-state exclusion, attack configurations, graph modification cost, hyperparameter spaces, and data processing in the main paper and appendices. Dataset statistics, detector configurations, and additional experimental results are also included in the appendices. Code and supporting materials are provided in anonymised form to facilitate reproduction of the reported results.

REFERENCES

Jinyin Chen, Yangyang Wu, Xuanheng Xu, Yixian Chen, Haibin Zheng, and Qi Xuan. Fast gradient attack on network embedding. *arXiv preprint arXiv:1809.02797*, 2018.

Jinhyeok Choi, Heehyeon Kim, and Joyce Jiyoung Whang. Unveiling the threat of fraud gangs to graph neural networks: Multi-target graph injection attacks against GNN-based fraud detectors. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2025.

Enyan Dai, Minhua Lin, Xiang Zhang, and Suhang Wang. Unnoticeable backdoor attacks on graph neural networks. In *Proceedings of the ACM Web Conference (WWW)*, 2023. doi: 10.1145/3543507.3583392.

Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International Conference on Machine Learning (ICML)*, 2018.

Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM)*, 2019. doi: 10.1137/1.9781611975673.67.

Xiangyu Dong, Xingyi Zhang, Yanni Sun, Lei Chen, Mingxuan Yuan, and Sibo Wang. SmoothGNN: Smoothing-aware GNN for unsupervised node anomaly detection. In *Proceedings of the ACM Web Conference 2025*, pp. 1225–1236, 2025. doi: 10.1145/3696410.3714615.

Haoyi Fan, Fengbin Zhang, and Zuoyong Li. AnomalyDAE: Dual autoencoder for anomaly detection on attributed networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020. doi: 10.1109/ICASSP40776.2020.9053387.

Changlong Fu, Zhenli He, Xiong Zhang, Cheng Xie, Xin Jin, and Yun Yang. DR-GGAD: Dual residual centering for mitigating anomaly non-discriminativity in generalist graph anomaly detection. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/pdf?id=kCuCHNChLE.

Yuan Gao, Xiang Wang, Xiangnan He, Zhenguang Liu, Huamin Feng, and Yongdong Zhang. Addressing heterophily in graph anomaly detection: A perspective of graph spectrum. In *Proceedings of the ACM Web Conference (WWW)*, 2023. doi: 10.1145/3543507.3583268.

Samyak Jain and Tanima Dutta. Stronger and transferable node injection attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 21179–21187, 2024. doi: 10.1609/aaai.v38i19.30111.

Naveen Karunanayake, Ravin Gunawardena, Suranga Seneviratne, and Sanjay Chawla. Out-of-distribution data: An acquaintance of adversarial examples — a survey. *ACM Computing Surveys*, 57(8):1–40, 2025. doi: 10.1145/3719292.

Bhavya Kohli and Biplab Sikdar. PEANUT: Perturbations by eigenvector alignment for attacking graph neural networks under topology-driven message passing. *arXiv preprint arXiv:2603.26136*, 2026.

Yuni Lai, Marcin Waniek, Liying Li, Jingwen Wu, Yulin Zhu, Tomasz Michalak, Talal Rahwan, and Kai Zhou. Coupled-space attacks against random-walk-based anomaly detection. *IEEE Transactions on Information Forensics and Security*, 2024. doi: 10.1109/TIFS.2024.3468156.

Yiqing Lin, Jianheng Tang, Chenyi Zi, H. Vicky Zhao, Yuan Yao, and Jia Li. UniGAD: Unifying multi-level graph anomaly detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Xudong Liu, Yanan Ren, Bin Yang, and Zhaonian Zou. LightGAD: Lightweight and effective framework for graph anomaly detection. *Knowledge-Based Systems*, 343:115898, 2026a. doi: 10.1016/j.knosys.2026.115898.

Yixin Liu, Zhao Li, Shirui Pan, Chen Gong, Chuan Zhou, and George Karypis. Anomaly detection on attributed networks via contrastive self-supervised learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. doi: 10.1109/TNNLS.2021.3068344.

Yunhui Liu, Qizhuo Xie, Yinfeng Chen, Xudong Jin, Tao Zheng, Bin Chong, and Tieke He. Mitigating homophily disparity in graph anomaly detection: A scalable and adaptive approach. In *Proceedings of the ACM Web Conference (WWW)*, 2026b.

Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski. Are defenses for graph neural networks robust? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Tran Gia Bao Ngo, Zulfikar Alom, Federico Errica, Murat Kantarcioglu, and Cuneyt Gurcan Akcora. Adversarial graph neural network benchmarks: Towards practical and fair evaluation. *arXiv preprint arXiv:2605.05534*, 2026.

Junjun Pan, Yixin Liu, Yizhen Zheng, and Shirui Pan. PREM: A simple yet effective approach for node-level graph anomaly detection. In *IEEE International Conference on Data Mining (ICDM)*, 2023. doi: 10.1109/ICDM58522.2023.00157.

Junjun Pan, Yixin Liu, Xin Zheng, Yizhen Zheng, Alan Wee-Chung Liew, Fuyi Li, and Shirui Pan. A label-free heterophily-guided approach for unsupervised graph fraud detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 12443–12451, 2025. doi: 10.1609/aaai.v39i12.33356.

Hezhe Qiao, Hanghang Tong, Bo An, Irwin King, Charu Aggarwal, and Guansong Pang. Deep graph anomaly detection: A survey and new perspectives. *IEEE Transactions on Knowledge and Data Engineering*, 37(9):5106–5126, 2025. doi: 10.1109/TKDE.2025.3581578.

Amit Roy, Juan Shu, Jia Li, Carl Yang, Olivier Elshocht, Jeroen Smeets, and Pan Li. GAD-NR: Graph anomaly detection via neighborhood reconstruction. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, 2024. doi: 10.1145/3616855.3635767.

Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Proceedings of the ACM Web Conference (WWW)*, 2020. doi: 10.1145/3366423.3380149.

Jianheng Tang, Fengrui Hua, Ziqi Gao, Peilin Zhao, and Jia Li. GADBench: Revisiting and benchmarking supervised graph anomaly detection. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

Shuchang Tao, Qi Cao, Huawei Shen, Junjie Huang, Yunfan Wu, and Xueqi Cheng. Single node injection attack against graph neural networks. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2021.

Xuhong Wang, Baihong Jin, Ying Du, Ping Cui, Yingshui Tan, and Yupu Yang. One-class graph neural networks for anomaly detection in attributed networks. *Neural Computing and Applications*, 2021.

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

Baoyuan Wu, Zihao Zhu, Li Liu, Qingshan Liu, Zhaofeng He, and Siwei Lyu. Attacks in adversarial machine learning: A systematic survey from the lifecycle perspective. *International Journal of Computer Vision*, 134(5):197, 2026. doi: 10.1007/s11263-025-02692-0.

Cong Wu, Jianfei Sun, Jing Chen, Mamoun Alazab, Yang Liu, and Yang Xiang. TCG-IDS: Robust network intrusion detection via temporal contrastive graph learning. *IEEE Transactions on Information Forensics and Security*, 20:1475–1486, 2025a. doi: 10.1109/TIFS.2025.3530702.

Tao Wu, Canyixing Cui, Xingping Xian, Shaojie Qiao, Chao Wang, Lin Yuan, and Shui Yu. Understanding the robustness of graph neural networks against adversarial attacks. *Knowledge-Based Systems*, 2025b. doi: 10.1016/j.knosys.2025.113714.

Sanju Xaviar and Omid Ardakanian. Budgeted indirect adversarial attack on graph-based anomaly detection in sensor networks. *arXiv preprint arXiv:2509.17987*, 2025.

Yiming Xu, Zihan Chen, Zhen Peng, Song Wang, Bin Shi, Bo Dong, and Chao Shen. Generalist graph anomaly detection via prototype-based distillation. In *International Conference on Machine Learning (ICML)*, 2026.

Yunfeng Zhao, Yixin Liu, Shiyuan Li, Qingfeng Chen, Yu Zheng, and Shirui Pan. FreeGAD: A training-free yet effective approach for graph anomaly detection. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2025.

Qinkai Zheng, Xu Zou, Yuxiao Dong, Yukuo Cen, Da Yin, Jiarong Xu, Yang Yang, and Jie Tang. Graph robustness benchmark: Benchmarking the adversarial robustness of graph machine learning. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2021.

Xiangping Zheng, Xuan Feng, Bo Wu, Li Yang, Bin Ren, Wei Li, Bin Tang, and Zhiwen Yu. Rethinking generalization in graphs: A hierarchical interaction perspective for generalist detection. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2026. doi: 10.1145/3770855.3817838.

Cheng Zhou, Guangxia Li, Hao Weng, and Yiyu Xiang. Training-free graph anomaly detection: A simple approach via singular value decomposition. In *Proceedings of the ACM Web Conference 2025*, pp. 4196–4205, 2025. doi: 10.1145/3696410.3714776.

Jialong Zhou, Xing Ai, Yuni Lai, Tomasz Michalak, G. Y. Li, Jianhua Li, Di Tang, and Xingxing Zhang. Adversarial robustness of link sign prediction in signed graphs. *IEEE Transactions on Dependable and Secure Computing*, 2026a. doi: 10.1109/TDSC.2026.3690461.

Jingjing Zhou, Shiyu Huang, Qing Qing, Zuquan Yuan, Huafei Huang, Ziqi Xu, Mingliang Hou, Xikun Zhang, Renqiang Luo, and Ivan Lee. GAD in the wild: Benchmarking graph anomaly detection under realistic deployment challenges. *arXiv preprint arXiv:2605.07133*, 2026b.

Yulin Zhu, Yuni Lai, Kaifa Zhao, Xiapu Luo, Mingquan Yuan, Jian Ren, and Kai Zhou. BinarizedAttack: Structural poisoning attacks to graph-based anomaly detection. In *IEEE International Conference on Data Engineering (ICDE)*, 2022.

Yulin Zhu, Yuni Lai, Kaifa Zhao, Xiapu Luo, Mingquan Yuan, Jun Wu, Jian Ren, and Kai Zhou. From bi-level to one-level: A framework for structural attacks to graph anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*, 36(4):6174–6187, 2024. doi: 10.1109/TNNLS.2024.3400395.

Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018. doi: 10.1145/3219819.3220078.

A Implementation Details

A.1 Attack Implementation

DICE uses full-graph binary normal/anomalous labels and chooses at random between two actions: deleting an edge that connects same-label nodes, or adding an edge that connects different-label nodes.

FGA takes gradients from a classification surrogate trained with full-graph labels: its training indices include every node, performs gradient ascent on the continuously relaxed adjacency matrix, and flips one target-incident edge per step.

DICE and FGA modify each target in turn on a shared adjacency and compute the budget from the cumulative graph degree when that target is processed, $b(v) = \max(1, \mathrm{round}(0.2 \cdot \deg(v)))$, with a budget of at least one attempt; they stop early when the candidate edges cannot exhaust the budget.

NFF updates target features sequentially, taking at each step the attribute mean over the two-hop neighborhood of the current graph (excluding the target itself), that is, the fusion with $\alpha = 1.0$, so that the target approaches the attribute pattern of its neighborhood; targets with an empty neighborhood keep their attributes.

Greedy is a white-box diagnostic probe in which the candidates of all targets compete for a global action budget (10 actions), stopping when the budget is exhausted or no candidate further lowers a score. Structural candidates come from local edge additions and legal deletions, and attribute candidates from feature replacement with other nodes; candidate pre-screening uses indicators built from the target set, the attribute side uses true score gradients where the detector is differentiable, and the final decision on an action is made by querying the score after the actual replacement.

Nettack-GAD evaluates dual-channel candidates with a frozen reconstruction surrogate fitted on the clean graph, retains Nettack's incremental closed-form scoring and filtering with a symmetrization, uses a per-target action budget (10 actions), and carries the accumulated graph state across targets. Candidate counts, filtering rules, and the configuration of the differentiable interface of each detector are provided with the public implementation.

Table 6: Attack information privileges in the evaluated implementation. All attacks know the target set and observe graph structure and attributes. "Labels" refers to additional ground-truth label access.

Attack	**Labels**	**Tested detector access**	**Surrogate**
DICE	Full graph	None	None
FGA	Full graph	None	Label-trained classifier
NFF	None	None	None
Greedy	None	Scores; gradients when available	None
Nettack-GAD	None	None	Frozen reconstruction model

A.2 Datasets and Baselines

The main text keeps only a summary of scale; the complete definitions of the six datasets follow. Node counts, undirected edge counts, and feature dimensions are computed directly from the raw

graph data; the anomaly rate is the share of anomalous nodes. Gen-1000 is a synthetic graph; Inj-Cora and Inj-Amazon are injected graphs on the Cora citation network and the Amazon co-purchase network, respectively (anomalies generated by the pyGOD injector, covering contextual and structural anomalies); Reddit is the Reddit community network with native labels, whose edge count excludes 10,984 self-loops; YelpChi-5K is a 5,000-node induced subgraph drawn from the YelpChi fraud-review dataset (45,954 nodes, 6,677 anomalies) by stratified random sampling of anomalies and normal nodes, with native labels; Weibo is the Weibo graph with native labels, originally directed (407,963 directed edges, no self-loops), and the edge count in the table is the number of unique undirected edges after removing self-loops.

Table 7: Dataset description. Node counts, undirected edge counts, and feature dimensions are computed directly from the raw graph data; the anomaly rate is the share of anomalous nodes.

Dataset	**#Nodes**	**#Edges**	**Dim**	**#Anomalies**	**Anomaly rate**
Gen-1000	1,000	2,468	64	20	2.00%
Inj-Cora	2,708	5,574	1,433	138	5.10%
Inj-Amazon	13,752	257,435	767	694	5.05%
Reddit	10,984	78,516	64	366	3.33%
YelpChi-5K	5,000	44,650	32	726	14.52%
Weibo	8,405	377,271	400	347	4.13%

The main text and tables use abbreviations for the ten baselines; their full names and mechanism families appear in Table 8.

Table 8: Full names and mechanism families of the evaluated baselines.

Baseline	**Family**	**Full name**
DOMINANT	reconstruction	Deep Anomaly Detection on Attributed Networks
OCGNN	one-class learning	One-Class Graph Neural Networks for Anomaly Detection in Attributed Networks
FreeGAD	training-free	FreeGAD: A Training-Free yet Effective Approach for Graph Anomaly Detection
SAGAD	spectral filtering	Mitigating Homophily Disparity in Graph Anomaly Detection: A Scalable and Adaptive Approach
TFGAD	training-free	Training-free Graph Anomaly Detection: A Simple Approach via Singular Value Decomposition
HUGE	heterophily-guided	A Label-free Heterophily-guided Approach for Unsupervised Graph Fraud Detection
LightGAD	lightweight framework	LightGAD: Lightweight and Effective Framework for Graph Anomaly Detection
SmoothGNN	smoothing-aware	SmoothGNN: Smoothing-aware GNN for Unsupervised Node Anomaly Detection
DR-GGAD	zero-shot generalist	DR-GGAD: Dual Residual Centering for Mitigating Anomaly Non-Discriminativity in Generalist Graph Anomaly Detection
HIMO-GAD	zero-shot generalist	Rethinking Generalization in Graphs: A Hierarchical Interaction Perspective for Generalist Detection

A.3 Clean-Performance Sensitivity

For each candidate configuration γ, $\mathrm{AUC}_{\mathrm{calib}}(\gamma)$ is the mean calibration-set AUC over seeds $\{0, 1, 2\}$. We use 0.70 as a clean-performance reference and define the calibration-threshold pass rate as $|\{\gamma : \mathrm{AUC}_{\mathrm{calib}}(\gamma) \geq 0.70\}|/|\Gamma_m|$. The sensitivity S_m is the maximum minus minimum of these configuration means. This analysis measures hyperparameter sensitivity and does not determine attack-matrix membership. The zero-shot baselines DR-GGAD and HIMO-GAD use fixed configurations and are excluded from this hyperparameter-sensitivity analysis.

Baseline performance can vary substantially even before attack. The calibrated AUC range S_m varies from 0.000 to 0.611. SmoothGNN on Weibo spans calibrated AUC values from 0.353 to 0.964 across configurations, covering both sides of the calibration reference threshold.

Across the eight tunable baselines and six datasets, the 48 combinations contain 756 configurations, of which 35.3% meet the reference threshold. Twenty-four combinations have no configuration meeting the reference threshold. On Reddit, all 126 configurations fall below the threshold, with a maximum calibrated AUC of 0.661 (SAGAD). The AUC range S_m remains below 0.05 for six baselines, with SAGAD and SmoothGNN showing greater variation. Reddit therefore contributes to the clean-performance analysis, while attack evaluation (Dimension I and Dimension II) covers the other five datasets. These five datasets contain 16 combinations without a configuration meeting the reference threshold, including four on Inj-Cora. On YelpChi-5K, LightGAD and SAGAD have calibration-threshold pass rates of 100% and 73%, respectively. The other six baselines remain below the threshold across their candidate configurations. Weibo has the widest calibration-threshold coverage among the real graphs: six of the eight baselines have configurations meeting the reference threshold (FreeGAD and HUGE have none).

FreeGAD on Gen-1000 has eight configuration means from 0.7149 to 0.7620, giving $S_m = 0.047$ and a 100% pass rate. Its separately screened seed-0 full-graph AUC is 0.6856; these measurements use different node sets and aggregation rules. Table 9 reports S_m and the calibration-threshold pass rate for each baseline on the five datasets used in attack evaluation.

Table 9: Hyperparameter sensitivity and calibration-threshold pass rates. Each cell gives the calibrated AUC range S_m and pass rate.

Baseline	**#Configs**	**Gen-1000**	**Inj-Cora**	**Inj-Amazon**	**YelpChi-5K**	**Weibo**
DOMINANT	8	0.002 / 100%	0.001 / 100%	0.002 / 100%	0.025 / 0%	0.022 / 100%
FreeGAD	8	0.047 / 100%	0.073 / 0%	0.152 / 0%	0.024 / 0%	0.049 / 0%
HUGE	6	0.027 / 50%	0.066 / 100%	0.044 / 100%	0.023 / 0%	0.063 / 0%
LightGAD	8	0.000 / 0%	0.022 / 0%	0.103 / 100%	0.009 / 100%	0.083 / 100%
OCGNN	8	0.024 / 100%	0.036 / 0%	0.093 / 0%	0.013 / 0%	0.048 / 100%
SAGAD	48	0.336 / 27%	0.086 / 6%	0.163 / 50%	0.113 / 73%	0.232 / 100%
SmoothGNN	32	0.338 / 3%	0.211 / 0%	0.232 / 0%	0.106 / 0%	0.611 / 50%
TFGAD	8	0.007 / 100%	0.062 / 100%	0.023 / 100%	0.007 / 0%	0.041 / 100%

The eight tunable baselines each use a fixed candidate configuration set, whose per-dimension values and configuration counts are given in Table 10. Configuration selection and early stopping use the held-out validation (calibration) set. Table 9 counts threshold passes across this candidate set. Attack ASR and $\Delta\mathrm{AUC}$ use the test nodes; the operational matrix screen described below uses the full graph.

Table 10: Hyperparameter search space of the baselines. Each row lists the per-dimension values of that baseline's candidate configurations, with candidate values in { }; the configuration count is the Cartesian product of the per-dimension values.

Baseline	**Search dimensions and values**	**#Configs**
SmoothGNN	hops $\{2, 4\}$; hidden dimension $\{32, 64\}$; learning rate $\{10^{-4}, 10^{-3}\}$; initialization $\{0.05, 0.5\}$; epochs $\{100, 200\}$	32
DOMINANT / OCGNN	learning rate $\{0.001, 0.004\}$; epochs $\{50, 100\}$; hidden dimension $\{32, 64\}$	8
TFGAD	$k_1\{1, 2\}$; $k_2\{3, 5\}$; $\eta\{5, 10\}$	8
FreeGAD	$\alpha\{0.05, 0.1\}$; $\beta\{0.8, 0.9\}$; number of hops $\{2, 4\}$; number of shots $\{30\}$	8
HUGE	learning rate $\{10^{-4}, 5 \times 10^{-4}\}$; distillation parameter $\{0.3, 0.5, 0.7\}$; hidden dimension $\{128\}$	6
LightGAD	number of estimators $\{100, 200\}$; learning rate $\{0.01, 0.02\}$; maximum depth $\{8, 12\}$	8
SAGAD	learning rate $\{0.001, 0.01, 0.1\}$; $k\{3, 5\}$; number of gating layers $\{1, 2\}$; feature transformation $\{\text{none}, \ell_2\}$; dropout $\{0, 0.3\}$	48

A.4 Shared Evaluation Protocol

We generate data splits deterministically using a fixed random seed; none of the six datasets provides a predefined split. Nodes are first stratified by label, and 5% are drawn as labeled seeds for the semi-

supervised baselines. Among the remaining nodes, anomalies are split equally between validation and test sets, the normal nodes are split 20%/80% into validation and test normal nodes, and the labeled seeds, validation nodes, and test nodes are pairwise disjoint. Training remains transductive over the full graph; the split constrains detector early stopping, configuration selection, and test reporting; attack targets are taken from the test anomalies, and the validation set serves early stopping, selection, and sensitivity analysis. DICE/FGA additionally access full-graph labels as specified in Table 6. Clean calibrated AUC is averaged over the three random seeds $\{0, 1, 2\}$.

Each detector–attack combination uses seed 0 for detector fitting, where required, and stochastic attack operations. Attacks are applied at test time with detector parameters frozen. The perturbation budget is $\rho = 0.2$, NFF uses $\alpha = 1.0$, Greedy uses a global budget of 10 actions, and Nettack-GAD uses a budget of 10 actions per target. Dimension I and Dimension II evaluate corresponding attack conditions; cost-replay provenance is detailed in Appendix A.5. GMC calibration uses sampling seed 2026, a positive pair pool of $n_{\text{pos}} = 4000$, and $n_{\text{attr}} = 5000$ attribute samples. All experiments were run on a single Linux server (Intel Xeon Platinum 8470Q, NVIDIA RTX 4090, PyTorch 2.1.2 + CUDA 12.1). The fixed matrix includes two baselines and five attacks on YelpChi-5K, and six baselines and three attacks on Weibo. Reddit does not enter attack evaluation.

A.5 Attack-Matrix Construction and Clean-State Matching

The attack pipeline used a seed-0 full-graph clean-AUC screen at 0.70, unlike the three-seed calibration-set reference in Table 9. The initial attack plans used a compact, predefined roster of tunable detectors, alongside screened fixed baselines. The reported subset retains Gen-1000, Inj-Cora, and Inj-Amazon from these plans. Later runs added YelpChi-5K and Weibo. Thus, the matrix is a fixed evaluation subset, not an exhaustive set of pairs exceeding the calibration reference. Among tunable detectors, Table 9 has seven passing pairs on Gen-1000 and four on Inj-Cora, whereas each attack roster contains three. Including fixed baselines, the final matrix contains 3, 5, 8, 2, and 6 detectors on Gen-1000, Inj-Cora, Inj-Amazon, YelpChi-5K, and Weibo, respectively.

FreeGAD illustrates the distinction: for the examined Gen-1000 seed-0 instance, calibration, test, and full-graph AUC are 0.637999, 0.739060, and 0.685612. The last value led to its exclusion by the operational screen, despite the 100% calibration-threshold pass rate of its three-seed configuration means.

Clean-state exclusion. Cross-attack comparisons for a detector–dataset pair require a matched clean reference state. TFGAD uses randomized low-rank SVD whose initialisation depends on the global random-number-generator state. On Inj-Cora, the main run had clean AUC 0.786382 and $|\mathcal{T}_c| = 11$, whereas the separately executed Greedy run had clean AUC 0.797354 and $|\mathcal{T}_c| = 14$, despite identical hyperparameters and nominal seed. We exclude this Greedy condition from all comparisons, figures, and aggregate statistics. Its original $3/14$ escape count, ASR 0.214, and $\Delta\text{AUC} = -0.058$ remain in the per-condition appendix for transparency; the denominator is not changed. The retained analysis comprises 83 conditions. After exclusion, the recorded clean AUC and clean captured counts agree across attacks within each of the 24 detector–dataset groups. These checks establish consistency of recorded clean metrics, not bitwise reproducibility of independently refitted detectors.

Computationally unavailable conditions and cost provenance. The matrix includes completed runs rather than a full factorial evaluation. Weibo FGA and Greedy and the missing Inj-Amazon attacks were not evaluated due to computational cost. A dash therefore denotes a computationally unavailable combination. The dagger denotes the separate clean-state exclusion above. GMC uses the recorded per-target cost files, including replayed perturbations. For Inj-Cora HIMO-GAD–Greedy, escape metrics and GMC come from the recorded attack and its later replay, respectively. On YelpChi-5K, FGA and Nettack-GAD replays differ from the original run by two and one net edge changes, respectively; feature-change counts agree. Thus, these joint summaries carry replay provenance rather than guaranteeing identical perturbations across executions.

B Graph Modification Cost: Construction and Interpretation

This section gives the detailed definition of the frozen pricing. Let π be the net modification set of the final graph relative to the clean graph: added and deleted edges are summed separately, and attributes are computed from the net final change. Terminal-state differencing absorbs intermediate reversals or repeated operations, so the statistical object matches the final perturbed graph. The attribute magnitude, the connection score, and the reference baselines form a data-driven pricing choice, and operational records can provide an external calibration of how this surrogate corresponds to actual resource overhead.

The attribute axis takes the net change before and after the modification and normalizes it by the per-dimension variation scale of normal nodes. With the robust scale $\sigma_j = \max(\mathrm{MAD}_j \times 1.4826,\ \varepsilon_\sigma)$ (MAD is the median absolute deviation over the normal nodes of the clean graph) as the reference for each dimension, the attribute modification cost compresses the relative magnitude logarithmically and sums over dimensions:

$$C_{\mathrm{attr}}(\pi) = \sum_j \log\left(1 + \frac{|\Delta_j|}{\sigma_j}\right), \tag{9}$$

where Δ_j is the net change of dimension j. Moving one σ_j contributes $\log 2 \approx 0.69$, expressing an equal modification magnitude on the relative scale.

The edge side prices through a connection plausibility score. Write $h(a, b)$ for the node-pair features, including the mean of the per-dimension standardized attribute difference clipped to $[0, 1]$, the smoothed neighborhood Jaccard distance, and $|\log(1+\deg a) - \log(1+\deg b)|$. With a design matrix H built from equally many normal edges and sampled normal non-edges (including a constant column), and $y_e \in \{0, 1\}$ marking whether an edge exists, compute

$$\beta = (H^\top H + \lambda I)^{-1} H^\top y_e, \qquad p_+(a, b) = \mathrm{clip}(\mathrm{sigmoid}([h(a, b), 1]\beta), \varepsilon_p, 1 - \varepsilon_p), \quad \lambda = 10^{-6}. \tag{10}$$

This score, a regularized least-squares fit mapped through a sigmoid, describes connection plausibility under balanced sampling. Interpretation as a true formation probability would require a graph-density prior and probability calibration, so the score is used through the relative magnitude of its values, assigning newly added edges a price in the opposite direction:

$$R_+(a, b) = -\log p_+(a, b), \qquad c_+(a, b) = \log\left(1 + \frac{R_+(a, b)}{M_+}\right), \tag{11}$$

where $M_+ = \max(\mathrm{median}\, R_+^{\mathrm{real}},\ \varepsilon_M)$ is the median of R_+ over normal real edges (with a numerical floor). The net added-edge cost is $C_+(\pi) = \sum_{(a,b)\in\pi} c_+(a, b)$.

Deletion is priced through the mirror image of the same connection score: the higher the connection plausibility, the higher the surrogate price of dissolving that relation, so that additions and deletions have opposite directional preferences:

$$R_-(a, b) = -\log\big(1 - p_+(a, b)\big), \qquad c_-(a, b) = \log\left(1 + \frac{R_-(a, b)}{M_-}\right), \tag{12}$$

where M_- is the median of R_- over normal real edges (with a numerical floor). The net deleted-edge cost is $C_-(\pi) = \sum_{(a,b)\in\pi} c_-(a, b)$, priced edge by edge over the edges the attack actually deletes.

The three axes are each divided by the baseline cost B of a reference event. The attribute reference event is a full-dimensional perturbation sampled independently per dimension from $\mathcal{N}(0, (\sigma_j/2)^2)$, with the sampling median of its C_{attr} as the baseline. The edge reference events build weighted distributions on normal non-edges and normal edges, respectively, with weights p_+ and $1 - p_+$, and the reference addition and deletion prices follow:

$$B_+ = \frac{\sum p_+\, c_+}{\sum p_+}, \qquad B_- = \frac{\sum (1 - p_+)\, c_-}{\sum (1 - p_+)}, \tag{13}$$

summed over normal non-edge pairs and normal edges, respectively. After normalization each axis is expressed in multiples of its own baseline, and the main cost is defined by an equal-weight additive rule:

$$Z_{\mathrm{attr}} = \frac{C_{\mathrm{attr}}}{B_{\mathrm{attr}}}, \qquad Z_+ = \frac{C_+}{B_+}, \qquad Z_- = \frac{C_-}{B_-}, \qquad C_\Sigma = Z_{\mathrm{attr}} + Z_+ + Z_- \quad \text{(main cost)}. \tag{14}$$

For a given final modification set, item-by-item pricing returns a determinate surrogate cost; inter-operation overhead and real resource constraints can be included through scenario-specific extensions. $C_\Sigma \leq 3$ means a total cost of at most three baseline units, which can be distributed across axes; the threshold describes the normalized modification scale. We report cost distributions over the modified target set, which contains both escaped and still-captured targets. The frozen cost function gives the same price to the same modification, while an adaptive strategy can choose different modifications according to the detector response. Appendix B.1 examines calibration stability through bootstrap resampling and a split-sample check. Calibration against operational records is a direction for future work.

B.1 Validity Checks

GMC uses pricing parameters estimated on the clean graph and frozen during attack evaluation. We assess the sensitivity of this calibration to node sampling on four of the evaluated datasets, YelpChi-5K, Inj-Cora, Inj-Amazon, and Weibo, using 200 bootstrap resamples of the calibration nodes with random seed 42. Table 11 summarizes parameter variation and the consistency of the resulting cost rankings.

Table 11: GMC pricing stability over 200 bootstrap resamples. Kendall W measures cost-ranking concordance; max CV is the largest coefficient of variation among the five pricing scalars. CI columns count parameters within bootstrap 95% intervals. The p_+ column covers three feature coefficients, with the intercept assessed separately.

Dataset	Kendall W	Max CV	Pricing in CI	p_+ coeff. in CI
YelpChi-5K	0.9997	0.27%	2/5	3/3
Inj-Cora	0.9986	0.71%	5/5	3/3
Inj-Amazon	0.9995	0.90%	5/5	3/3
Weibo	0.9990	0.99%	5/5	3/3

YelpChi-5K: maximum interval excursion 0.13%.

Across the resampled calibrations, the C_Σ rankings on the $5\times5\times5$ reference grid (each axis scaled 0.2–5) have Kendall W of at least 0.9986. The largest coefficient of variation among the five pricing scalars (B_{attr}, B_+, B_-, M_+, M_-) is 0.99%. On YelpChi-5K, the frozen values of B_+, M_+, and M_- lie 0.08–0.13% outside their bootstrap 95% intervals. For each of these quantities, the deviation from the resample mean is 1.14 times the interval half-width. The corresponding sensitivity estimate gives a relative change in C_Σ of at most 0.13% after the logarithmic transformation and reference-event normalization. The three feature coefficients of p_+ fall within their respective bootstrap intervals on all four datasets, as reported in Table 11. The separately assessed intercept also lies within its interval on each dataset.

We also examine split-sample consistency of the connection score p_+, a four-parameter regularized least-squares fit ($\lambda = 10^{-6}$) on about eight thousand pooled normal edges and sampled non-edges. After fitting on one half of this pool, the median $R_+ = -\log p_+$ on the held-out half differs from that on the fitted half by at most 0.0043, indicating similar median connection costs across the two subsets.

The small parameter variation and consistent reference-grid rankings across these resampled calibrations support the stability of C_Σ as the modification-cost dimension for comparisons on the four tested datasets.

C Detailed Target-Escape Results

The complete ASR–ΔAUC matrix appears in Table 1 in the main paper. The following tables provide dataset-specific comparisons and pooled target-escape results.

C.1 Dataset-Specific Comparisons

Table 12: Escape rate (ASR) of the two baselines on YelpChi-5K, attack by attack. Under the same attack the two baselines can differ by an order of magnitude, and their relative order under structural attacks is inverted under the feature attack.

Attack	Channel	LightGAD	SAGAD
DICE	structural	0.429	0.022
FGA	structural	0.540	0.000
Greedy	hybrid (adaptive)	0.159	0.124
NFF	feature	0.841	1.000
Nettack-GAD	hybrid (surrogate)	0.730	0.247

Table 13: Escape rate (ASR) of the six baselines on Weibo, attack by attack, sorted by clean AUC in descending order. Across the six baselines, ASR ranges from 0.000 to 0.184 for DICE and from 0.000 to 0.158 for Nettack-GAD.

Attack	Channel	SAGAD	SmoothGNN	DOMINANT	OCGNN	LightGAD	TFGAD
DICE	structural	0.105	0.052	0.058	0.060	0.184	0.000
NFF	feature	0.073	0.409	0.453	0.119	0.737	0.895
Nettack-GAD	hybrid (surrogate)	0.129	0.052	0.058	0.083	0.158	0.000

C.2 Pooled and Per-Condition Results

Table 14: Target-escape summary by dataset and attack. Macro ASR weights detector–attack cells equally; micro ASR weights by clean captured targets. Escaped/captured counts detector–target pairs.

Dataset	Attack	Cells	ASR macro	ASR median	ASR micro	Esc./Capt.	ΔAUC
Gen-1000	DICE	3	0.417	0.500	0.375	3/8	-0.014
Gen-1000	FGA	3	0.167	0.000	0.125	1/8	-0.042
Gen-1000	Greedy	3	0.833	1.000	0.750	6/8	$+0.448$
Gen-1000	NFF	3	0.917	1.000	0.875	7/8	$+0.677$
Gen-1000	Nettack-GAD	3	1.000	1.000	1.000	8/8	$+0.439$
Inj-Amazon	DICE	8	0.029	0.025	0.024	24/980	-0.017
Inj-Amazon	NFF	8	0.604	0.904	0.752	737/980	$+0.366$
Inj-Cora	DICE	5	0.134	0.103	0.119	13/109	-0.042
Inj-Cora	FGA	5	0.056	0.000	0.046	5/109	-0.059
Inj-Cora	Greedy	4	0.396	0.320	0.306	30/98	$+0.062$
Inj-Cora	NFF	5	0.698	0.636	0.679	74/109	$+0.329$
Inj-Cora	Nettack-GAD	5	0.778	0.818	0.734	80/109	$+0.225$

Table 15: Per-condition target escape and GMC on Gen-1000. GMC uses modified targets; edges/dims are the raw attack's modification counts.

Detector	Attack	ASR	ΔAUC	Esc./Capt.	GMC	Mod./Targets	Edges/Dims
DR-GGAD	DICE	0.500	−0.034	1/2	1.59	10/10	21/0
DR-GGAD	FGA	0.500	−0.014	1/2	2.14	10/10	22/0
DR-GGAD	NFF	1.000	+0.788	2/2	1.88	10/10	0/640
DR-GGAD	Greedy	1.000	+0.311	2/2	1.88	7/10	3/448
DR-GGAD	Nettack-GAD	1.000	+0.412	2/2	5.08	10/10	58/448
HUGE	DICE	0.500	+0.002	1/2	1.59	10/10	21/0
HUGE	FGA	0.000	−0.024	0/2	2.14	10/10	22/0
HUGE	NFF	1.000	+0.830	2/2	1.88	10/10	0/640
HUGE	Greedy	1.000	+0.680	2/2	1.85	10/10	0/640
HUGE	Nettack-GAD	1.000	+0.535	2/2	5.08	10/10	58/448
SAGAD	DICE	0.250	−0.009	1/4	1.59	10/10	21/0
SAGAD	FGA	0.000	−0.088	0/4	2.14	10/10	22/0
SAGAD	NFF	0.750	+0.414	3/4	1.88	10/10	0/640
SAGAD	Greedy	0.500	+0.353	2/4	1.87	10/10	1/576
SAGAD	Nettack-GAD	1.000	+0.370	4/4	5.08	10/10	58/448

Table 16: Per-condition target escape and GMC on Inj-Cora. GMC uses modified targets; edges/dims are the raw attack's modification counts. †: original run retained for transparency, excluded from comparisons and aggregate analysis (Appendix A.5).

Detector	Attack	ASR	ΔAUC	Esc./Capt.	GMC	Mod./Targets	Edges/Dims
DOMINANT	DICE	0.071	−0.072	2/28	1.48	66/66	118/0
DOMINANT	FGA	0.000	−0.079	0/28	1.34	66/66	119/0
DOMINANT	NFF	0.393	+0.091	11/28	8.48	66/66	0/44400
DOMINANT	Greedy	0.036	+0.023	1/28	1.02	9/66	7/62
DOMINANT	Nettack-GAD	0.929	+0.404	26/28	6.38	66/66	343/3300
DR-GGAD	DICE	0.133	−0.024	4/30	1.48	66/66	118/0
DR-GGAD	FGA	0.100	−0.037	3/30	1.34	66/66	119/0
DR-GGAD	NFF	1.000	+0.521	30/30	8.48	66/66	0/44400
DR-GGAD	Greedy	0.467	+0.075	14/30	0.78	10/66	5/105
DR-GGAD	Nettack-GAD	0.800	+0.127	24/30	6.38	66/66	343/3300
HIMO-GAD	DICE	0.364	−0.044	4/11	1.48	66/66	118/0
HIMO-GAD	FGA	0.182	−0.016	2/11	1.34	66/66	119/0
HIMO-GAD	NFF	0.909	+0.396	10/11	8.48	66/66	0/44400
HIMO-GAD	Greedy	0.909	+0.077	10/11	1.09	8/66	5/73
HIMO-GAD	Nettack-GAD	0.818	+0.117	9/11	6.38	66/66	343/3300
HUGE	DICE	0.103	−0.073	3/29	1.48	66/66	118/0
HUGE	FGA	0.000	−0.080	0/29	1.34	66/66	119/0
HUGE	NFF	0.552	+0.208	16/29	8.48	66/66	0/44400
HUGE	Greedy	0.172	+0.073	5/29	1.02	9/66	7/81
HUGE	Nettack-GAD	0.345	+0.019	10/29	6.38	66/66	343/3300
TFGAD	DICE	0.000	+0.002	0/11	1.48	66/66	118/0
TFGAD	FGA	0.000	−0.084	0/11	1.34	66/66	119/0
TFGAD	NFF	0.636	+0.431	7/11	8.48	66/66	0/44400
TFGAD	Greedy†	0.214	−0.058	3/14	1.12	7/66	5/116
TFGAD	Nettack-GAD	1.000	+0.457	11/11	6.38	66/66	343/3300

Table 17: Per-condition target escape and GMC on Inj-Amazon. GMC uses modified targets; edges/dims are the raw attack's modification counts. LightGAD's ineffective attacks have ASR 0.000 (0/120 escaped/captured).

Detector	Attack	ASR	ΔAUC	Esc./Capt.	GMC	Mod./Targets	Edges/Dims
DOMINANT	DICE	0.036	−0.017	2/55	23.95	330/330	4681/0
DOMINANT	NFF	0.073	+0.012	4/55	20.19	324/330	0/248501
DR-GGAD	DICE	0.039	−0.028	7/178	23.95	330/330	4681/0
DR-GGAD	NFF	1.000	+0.652	178/178	20.19	324/330	0/248501
HIMO-GAD	DICE	0.005	−0.022	1/185	23.95	330/330	4681/0
HIMO-GAD	NFF	0.968	+0.445	179/185	20.19	324/330	0/248501
HUGE	DICE	0.013	−0.023	3/236	23.95	330/330	4681/0
HUGE	NFF	0.847	+0.259	200/236	20.19	324/330	0/248501
LightGAD	DICE	0.000	+0.000	0/120	23.95	330/330	4681/0
LightGAD	NFF	0.000	+0.000	0/120	20.19	324/330	0/248501
SAGAD	DICE	0.099	+0.007	10/101	23.95	330/330	4681/0
SAGAD	NFF	0.960	+0.732	97/101	20.19	324/330	0/248501
SmoothGNN	DICE	0.000	−0.054	0/80	23.95	330/330	4681/0
SmoothGNN	NFF	0.988	+0.660	79/80	20.19	324/330	0/248501
TFGAD	DICE	0.040	−0.003	1/25	23.95	330/330	4681/0
TFGAD	NFF	0.000	+0.172	0/25	20.19	324/330	0/248501

Table 18: Per-condition target escape and GMC on YelpChi-5K. GMC uses modified targets; edges/dims are the raw attack's modification counts.

Detector	Attack	ASR	ΔAUC	Esc./Capt.	GMC	Mod./Targets	Edges/Dims
LightGAD	DICE	0.429	+0.017	27/63	3.13	345/345	1189/0
LightGAD	FGA	0.540	−0.021	34/63	3.09	345/345	1215/0
LightGAD	NFF	0.841	+0.044	53/63	12.83	343/345	0/10735
LightGAD	Greedy	0.159	+0.021	10/63	4.33	10/345	0/186
LightGAD	Nettack-GAD	0.730	+0.090	46/63	9.97	339/345	2882/2886
SAGAD	DICE	0.022	+0.001	2/89	3.13	345/345	1189/0
SAGAD	FGA	0.000	−0.001	0/89	3.09	345/345	1215/0
SAGAD	NFF	1.000	+0.157	89/89	12.83	343/345	0/10735
SAGAD	Greedy	0.124	+0.024	11/89	8.07	10/345	0/226
SAGAD	Nettack-GAD	0.247	+0.019	22/89	9.97	339/345	2882/2886

Table 19: Per-condition target escape and GMC on Weibo. GMC uses modified targets; edges/dims are the raw attack's modification counts.

Detector	Attack	ASR	ΔAUC	Esc./Capt.	GMC	Mod./Targets	Edges/Dims
DOMINANT	DICE	0.058	−0.008	5/86	7.03	165/165	1451/0
DOMINANT	NFF	0.453	+0.104	39/86	4.96	165/165	0/66000
DOMINANT	Nettack-GAD	0.058	−0.006	5/86	10.56	165/165	1650/0
LightGAD	DICE	0.184	+0.008	14/76	7.03	165/165	1451/0
LightGAD	NFF	0.737	+0.097	56/76	4.96	165/165	0/66000
LightGAD	Nettack-GAD	0.158	+0.012	12/76	10.56	165/165	1650/0
OCGNN	DICE	0.060	−0.003	5/84	7.03	165/165	1451/0
OCGNN	NFF	0.119	+0.030	10/84	4.96	165/165	0/66000
OCGNN	Nettack-GAD	0.083	+0.015	7/84	10.56	165/165	1650/0
SAGAD	DICE	0.105	+0.006	13/124	7.03	165/165	1451/0
SAGAD	NFF	0.073	+0.004	9/124	4.96	165/165	0/66000
SAGAD	Nettack-GAD	0.129	+0.012	16/124	10.56	165/165	1650/0
SmoothGNN	DICE	0.052	+0.004	6/115	7.03	165/165	1451/0
SmoothGNN	NFF	0.409	+0.075	47/115	4.96	165/165	0/66000
SmoothGNN	Nettack-GAD	0.052	0.000	6/115	10.56	165/165	1650/0
TFGAD	DICE	0.000	0.000	0/95	7.03	165/165	1451/0
TFGAD	NFF	0.895	+0.305	85/95	4.96	165/165	0/66000
TFGAD	Nettack-GAD	0.000	0.000	0/95	10.56	165/165	1650/0

D Additional Effectiveness and Cost Results

The cost of a non-adaptive attack is reused across detectors and does not constitute an independent replicate; Greedy costs and coverage remain detector-specific.

D.1 Modification Coverage and Anomaly Classes

Table 20: Greedy modification coverage and target escape by detector. Modified/targets counts all attack targets; escaped/captured counts clean captured targets. GMC is the median over modified targets under a global budget of ten actions. †: original excluded run, retained for transparency; see Appendix A.5.

Dataset	Detector	Modified/Targets	Escaped/Captured	ASR	Median GMC
Inj-Cora	DOMINANT	9/66	1/28	0.036	1.02
Inj-Cora	DR-GGAD	10/66	14/30	0.467	0.78
Inj-Cora	HIMO-GAD	8/66	10/11	0.909	1.09
Inj-Cora	HUGE	9/66	5/29	0.172	1.02
Inj-Cora	TFGAD†	7/66	3/14	0.214	1.12
YelpChi-5K	LightGAD	10/345	10/63	0.159	4.33
YelpChi-5K	SAGAD	10/345	11/89	0.124	8.07

Table 21: Anomaly-class detail on the injected graphs. Counts sum detector–target pairs; shared non-adaptive perturbations are not independent cost replicates.

Dataset	Attack	Attribute anomaly			Structural anomaly		
		Escaped/Captured	ASR	GMC	Escaped/Captured	ASR	GMC
Gen-1000	DICE	0/1	0.000	1.03	3/7	0.429	2.20
Gen-1000	FGA	0/1	0.000	1.11	1/7	0.143	2.97
Gen-1000	NFF	1/1	1.000	1.99	6/7	0.857	1.80
Gen-1000	Greedy	1/1	1.000	1.95	5/7	0.714	1.84
Gen-1000	Nettack-GAD	1/1	1.000	1.91	7/7	1.000	7.34
Inj-Cora	DICE	6/43	0.140	0.98	7/66	0.106	3.27
Inj-Cora	FGA	3/43	0.070	0.92	2/66	0.030	1.88
Inj-Cora	NFF	41/43	0.953	3.67	33/66	0.500	11.22
Inj-Cora	Greedy	24/38	0.632	1.02	6/60	0.100	0.74
Inj-Cora	Nettack-GAD	35/43	0.814	2.57	45/66	0.682	9.95
Inj-Amazon	DICE	3/515	0.006	4.15	21/447	0.047	31.57
Inj-Amazon	NFF	459/515	0.891	20.13	263/447	0.588	20.41

Table 22: Target escape and GMC by anomaly class on the injected graphs. ASR uses class-specific clean captured targets; GMC uses modified targets. Greedy costs pool detector-specific attacks; other attacks share perturbations across detectors.

Dataset	Attack	ASR attribute	ASR structural	GMC attribute	GMC structural
Inj-Cora	DICE	0.140	0.106	0.98	3.27
Inj-Cora	FGA	0.070	0.030	0.92	1.88
Inj-Cora	NFF	0.953	0.500	3.67	11.22
Inj-Cora	Greedy	0.632	0.100	1.02	0.74
Inj-Cora	Nettack-GAD	0.814	0.682	2.57	9.95
Inj-Amazon	DICE	0.006	0.047	4.15	31.57
Inj-Amazon	NFF	0.891	0.588	20.13	20.41

D.2 Cost by Detector and Attack

Table 23: Per-detector median GMC over modified targets, aligned with Table 1. **Bold** marks GMC at most three reference units. Non-adaptive attacks share one perturbed graph per dataset; Greedy runs separately for each detector. † denotes the excluded condition in Table 1.

Dataset	Detector	DICE	FGA	NFF	Greedy	Nettack-GAD
Gen-1000	DR-GGAD	**1.59**	**2.14**	**1.88**	**1.88**	5.08
	HUGE	**1.59**	**2.14**	**1.88**	**1.85**	5.08
	SAGAD	**1.59**	**2.14**	**1.88**	**1.87**	5.08
Inj-Cora	DOMINANT	**1.48**	**1.34**	8.48	**1.02**	6.38
	DR-GGAD	**1.48**	**1.34**	8.48	**0.78**	6.38
	HIMO-GAD	**1.48**	**1.34**	8.48	**1.09**	6.38
	HUGE	**1.48**	**1.34**	8.48	**1.02**	6.38
	TFGAD	**1.48**	**1.34**	8.48	–†	6.38
Inj-Amazon	DOMINANT	23.95	–	20.19	–	–
	DR-GGAD	23.95	–	20.19	–	–
	HIMO-GAD	23.95	–	20.19	–	–
	HUGE	23.95	–	20.19	–	–
	LightGAD	23.95	–	20.19	–	–
	SAGAD	23.95	–	20.19	–	–
	SmoothGNN	23.95	–	20.19	–	–
	TFGAD	23.95	–	20.19	–	–
YelpChi-5K	LightGAD	3.13	3.09	12.83	4.33	9.97
	SAGAD	3.13	3.09	12.83	8.07	9.97
Weibo	DOMINANT	7.03	–	4.96	–	10.56
	LightGAD	7.03	–	4.96	–	10.56
	OCGNN	7.03	–	4.96	–	10.56
	SAGAD	7.03	–	4.96	–	10.56
	SmoothGNN	7.03	–	4.96	–	10.56
	TFGAD	7.03	–	4.96	–	10.56

Table 24: Effectiveness and cost by dataset and attack. ASR weights included detectors equally; GMC is the median over modified targets. Detector-dependent costs and low-cost shares are shown as ranges.

Dataset	Attack	ASR macro mean	Median GMC	Low-cost share
Gen-1000	DICE	0.417	1.59	80%
Gen-1000	FGA	0.167	2.14	70%
Gen-1000	NFF	0.917	1.88	100%
Gen-1000	Greedy	0.833	1.85–1.88	86%–100%
Gen-1000	Nettack-GAD	1.000	5.08	20%
Inj-Cora	DICE	0.134	1.48	68%
Inj-Cora	FGA	0.056	1.34	95%
Inj-Cora	NFF	0.698	8.48	17%
Inj-Cora	Greedy	0.396	0.78–1.09	100%
Inj-Cora	Nettack-GAD	0.778	6.38	30%
YelpChi-5K	DICE	0.226	3.13	46%
YelpChi-5K	FGA	0.270	3.09	39%
YelpChi-5K	NFF	0.921	12.83	0%
YelpChi-5K	Greedy	0.141	4.33–8.07	0%–40%
YelpChi-5K	Nettack-GAD	0.489	9.97	5%
Weibo	DICE	0.076	7.03	0%
Weibo	NFF	0.448	4.96	1%
Weibo	Nettack-GAD	0.080	10.56	0%
Inj-Amazon	DICE	0.029	23.95	17%
Inj-Amazon	NFF	0.604	20.19	0%

Table 25: Graph modification cost by dataset and attack. GMC is the median over modified targets; detector-dependent values are shown as ranges. Low-cost shares use modified targets as their denominator.

Dataset	Attack	Median GMC range	Low/very-low cost
Gen-1000	DICE	1.59	80% / 20%
Gen-1000	FGA	2.14	70% / 0%
Gen-1000	NFF	1.88	100% / 0%
Gen-1000	Greedy	1.85–1.88	100% / 0%
Gen-1000	Nettack-GAD	5.08	20% / 10%
Inj-Cora	DICE	1.48	68% / 30%
Inj-Cora	FGA	1.34	95% / 47%
Inj-Cora	NFF	8.48	17% / 2%
Inj-Cora	Greedy	0.78–1.09	100% / 44%
Inj-Cora	Nettack-GAD	6.38	30% / 9%
Inj-Amazon	DICE	23.95	17% / 4%
Inj-Amazon	NFF	20.19	0% / 0%

D.3 Score-Response Diagnostics

Table 26: Score-response diagnostics by dataset and attack. Target and normal drops are graph-normalized z-score changes; positive values mean lower scores after attack. Target means use all attack targets. FP change and A→N share use the clean-graph threshold; escaped/captured counts detector–target pairs. The normal mean drop is distinct from a decision-quantile change.

Dataset	Attack	Esc./Capt.	Target drop	Normal drop	FP change	A→N share
Gen-1000	DICE	3/8	−0.128	+0.002	+0.029	0.0013
Gen-1000	FGA	1/8	−0.439	+0.006	+0.009	0.0004
Gen-1000	NFF	7/8	+2.894	−0.039	+0.002	0.0031
Gen-1000	Greedy	6/8	+1.694	−0.023	+0.009	0.0022
Gen-1000	Nettack-GAD	8/8	+1.398	−0.019	+0.040	0.0035
Inj-Cora	DICE	13/109	−0.190	+0.006	+0.002	0.0012
Inj-Cora	FGA	5/109	−0.251	+0.008	+0.004	0.0004
Inj-Cora	NFF	74/109	+0.887	−0.030	−0.001	0.0072
Inj-Cora	Greedy	30/98	+0.306	−0.010	+0.016	0.0024
Inj-Cora	Nettack-GAD	80/109	+0.423	−0.014	+0.007	0.0075
Inj-Amazon	DICE	24/980	−0.091	+0.003	+0.003	0.0002
Inj-Amazon	NFF	737/980	+1.257	−0.042	+0.003	0.0090
YelpChi-5K	DICE	29/152	+0.045	−0.005	−0.001	0.0040
YelpChi-5K	FGA	34/152	−0.019	+0.002	−0.004	0.0050
YelpChi-5K	NFF	142/152	+0.513	−0.055	−0.003	0.0198
YelpChi-5K	Greedy	21/152	+0.085	−0.009	0.000	0.0028
YelpChi-5K	Nettack-GAD	68/152	+0.225	−0.024	−0.003	0.0097
Weibo	DICE	43/580	−0.034	+0.001	−0.011	0.0034
Weibo	NFF	246/580	+0.763	−0.021	−0.004	0.0069
Weibo	Nettack-GAD	46/580	−0.017	0.000	−0.010	0.0034