WHEN IS REWEIGHTING NOT ENOUGH? A HODGE CERTIFICATE FOR REWIRING NOISY AND INCOMPLETE GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph repair methods change an edge in one of two ways. They change how much it counts, which we call reweighting, or whether it exists, which we call rewiring. Structure learners mix the two moves and rarely ask which one a graph needs. We give an exact answer to one form of this question. The predictions of a frozen node classifier, compared across each edge, form a flow on the graph. Hodge decomposition splits the flow into a gradient part, which one score per node explains, and a cycle part, which no such score explains. We prove that no positive reweighting of a fixed edge set lets one score per node explain the cycle part, and that weights of spread γ move its share of the flow by at most a factor γ. A nonzero cycle part therefore certifies that an exact fit requires rewiring. We turn the certificate into the Hodge Router. It rewires an instance only when the cycle fraction is large relative to its chance level, ranks the edits by their share of the cycle part, and sends the edits on which the structural signals disagree to a language model under a fixed query budget. The bound holds on every weighting we draw. Corruption raises the diagnostic to about 1.6 times its clean value, and the diagnostic predicts which operation helps better than seven competing statistics, with a partial Spearman correlation of 0.38. Under random corruption the router matches or exceeds the better fixed operation on every dataset. Under Metattack it rewires nearly every instance, and under PRBCD it reweights the graphs on which reweighting wins, so it follows the operation each attack favors. It has the best average accuracy of all compared methods under both attacks, ahead of the joint structure learners JDR and Pro-GNN.

1 INTRODUCTION

Graph neural networks (GNNs) classify nodes by passing messages along the observed edges, so they inherit the errors in those edges. A citation graph contains references added out of habit and lacks citations that nobody recorded. Graph structure learning repairs the graph before or during training, and every such method does one of two things to an edge. It changes how much the edge counts, or it changes whether the edge exists. We call the first *reweighting*. It rescales the edges of a fixed edge set, the *support*, so every path that existed before still exists afterwards. We call the second *rewiring*. It changes the support, and with it the set of nodes that can exchange messages at all.

Existing methods mix the two operations. Pro-GNN learns a weighted adjacency with one objective that both rescales and removes edges (Jin et al., 2020). PTDNet and GNNGuard learn edge weights or drop probabilities with the model, and a weight driven to zero is a deletion (Luo et al., 2021; Zhang & Zitnik, 2020). Rewiring methods add and remove edges by learned link scores, by spectral alignment of the graph with the features, or by a language model, and they rewire every graph they receive (Dong & Kluger, 2023; Linkerhägner et al., 2025; Zhang et al., 2025b). None of them asks which operation a given graph needs. We ask two questions. *Can we tell, before any repair is trained, whether reweighting could ever fit the evidence on the edges? And does the answer predict which operation improves accuracy?*

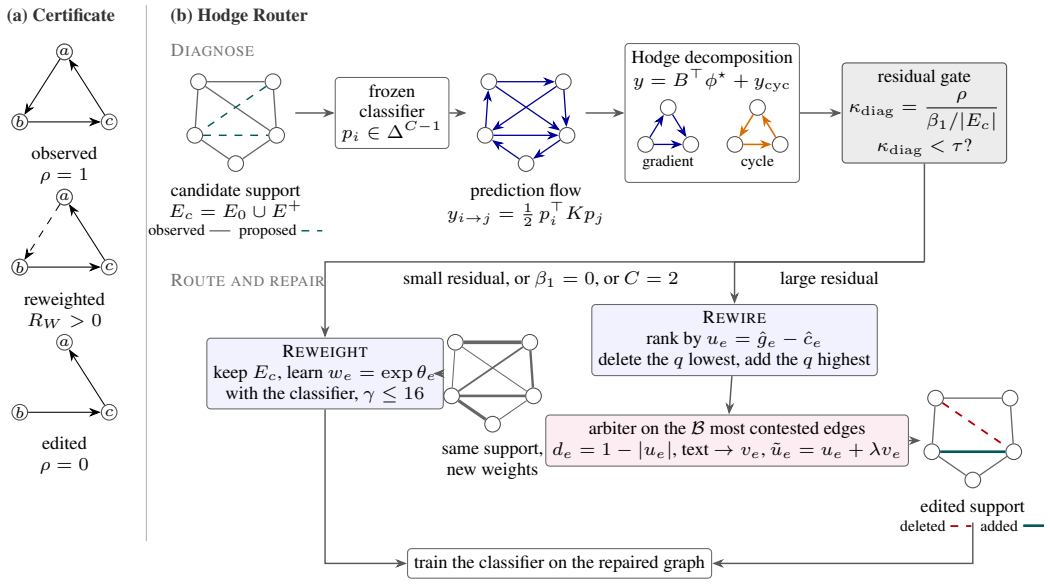


Figure 1: **(a)** Comparisons that prefer a to b, b to c, and c to a form a cycle that no node scores explain, so $\rho = 1$. Downweighting one edge (dashed) changes the least squares compromise but leaves the weighted residual R_W positive. Deleting it leaves a path, where $\rho = 0$. **(b)** The Hodge Router. A frozen classifier turns the candidate support into a prediction flow, the Hodge decomposition separates its gradient and cycle parts, and the residual gate compares the cycle fraction with its chance level. Small residuals keep the support and learn positive weights. Large residuals rank the edges by link score minus cycle share, let a language model arbitrate the $\mathcal{B}$ most contested edges, and edit the support.

The first question has an exact answer. Suppose the evidence on three edges says that a ranks above b, b above c, and c above a (Figure 1a). No values on the three nodes agree with all three comparisons. Lowering the weight of one edge changes how a least squares fit compromises, but the contradiction stays. Deleting the edge removes it. A frozen classifier supplies such comparisons on a real graph. Comparing the two endpoint distributions of an edge gives a signed value, and the resulting edge signal is the *prediction flow*. Hodge decomposition (Lim, 2020) splits it into a gradient part, which one score per node explains, and a cycle part, which no such score explains. We prove that positive reweighting of a fixed support never lets one score per node explain the cycle part, and that weights of spread $\gamma = w_{\max}/w_{\min}$ keep the weighted residual fraction between ρ_0/γ and $\gamma\rho_0$, where ρ_0 is the unweighted cycle fraction. A nonzero ρ_0 therefore certifies that an exact fit needs rewiring, and a large one rules out a close fit by weights of bounded spread.

The second question is empirical. We compute the certificate on five benchmarks and compare reweighting with rewiring on 370 randomly corrupted instances and under two attacks, Metattack (Zügner & Günnemann, 2019) and PRBCD (Geisler et al., 2021). Three findings emerge. First, the bound holds on all 4,000 weightings we draw, and the weightings move the cycle fraction far less than the bound allows. On Cora the fraction stays between 0.085 and 0.101, while the bound allows anything from 0.006 to 1.606. Second, corruption concentrates the cycle part. On a corrupted graph the diagnostic is about 1.6 times its clean value, and it predicts which operation helps better than the corruption rate, prediction disagreement, ESNR (Dong & Kluger, 2023), and four structural and feature statistics, with a partial Spearman correlation of 0.38. Third, routing on the diagnostic pays. Under random corruption the router matches or exceeds the better fixed operation on all five datasets. Under Metattack it rewires nearly every instance, and under PRBCD it reweights on Cora and CiteSeer, which is the operation each attack favors. It has the best average accuracy of all compared methods under both attacks, ahead of the strongest structure learners.

Our contributions are as follows.

- **A certificate for reweighting.** We formalize graph repair as a choice between positive reweighting of a fixed support and editing of the support. We prove that the cycle part

of the prediction flow survives every positive reweighting, and that weights of spread γ move its share by at most a factor γ (Theorem 1). We characterize the comparison matrices whose flows can circulate at all (Proposition 1).

- **A diagnostic that predicts the operation.** We check the theorem on real instances and show that corruption concentrates the cycle part, so that its calibrated size predicts which operation helps, ahead of every competing diagnostic and under criteria fixed in advance (Sections 5.2 and 5.3).
- **The Hodge Router.** The router gates on the residual, ranks the edits by their cycle contribution, and sends the most contested edits to a language model under a fixed query budget. It matches or exceeds the better fixed operation on every dataset under random corruption, exceeds both on every graph with node text under Metattack, and has the best average accuracy of all compared methods under both attacks (Sections 5.3 and 5.4).

2 Related Work

Graph structure learning. Pro-GNN learns the adjacency jointly with the model under sparsity, low-rank, and feature smoothness penalties (Jin et al., 2020). GCN-SVD and GCN-Jaccard clean the graph beforehand (Entezari et al., 2020; Wu et al., 2019), and PTDNet learns a probability of dropping each edge (Luo et al., 2021). CoGSL, SE-GSL, JDR, and SLAPS refine the graph through mutual information, structural entropy, spectral resonance, or self-supervision (Liu et al., 2022; Zou et al., 2023; Linkerhägner et al., 2025; Fatemi et al., 2021). RGCN keeps the graph and changes the message function (Zhu et al., 2019). GNNGuard weights each message by the similarity of its endpoints and prunes the edges between dissimilar nodes (Zhang & Zitnik, 2020). Language models have been asked to propose or verify edges on graphs with node text (Sun et al., 2025; Zhang et al., 2025b). Most methods that change the graph learn a weighted adjacency (Zhou et al., 2026). In such an adjacency, rescaling, removing, and inserting an edge are moves of one optimizer. Signed weights and sheaf Laplacians form a third class that changes the sign or restriction map of an edge (Bo et al., 2021; Bodnar et al., 2022). We separate the first two classes and ask when the first cannot fit the evidence.

Hodge decomposition of edge flows. Hodge theory splits an edge signal into a gradient of node potentials and a residual in the cycle space (Lim, 2020; Schaub et al., 2020). HodgeRank uses this split to rank items from pairwise comparisons, with weights fixed by the number of comparisons per pair, and reads the residual as inconsistency among the comparisons (Jiang et al., 2011). The same split underlies the decomposition of games (Candogan et al., 2011), and cycle inconsistency has been used to find wrong edges in view graphs (Zach et al., 2010). We apply the decomposition to a flow built from classifier predictions and ask a different question. HodgeRank asks how well potentials fit a given set of comparisons. We ask whether any weighting of the comparisons can make the fit exact, and show that the answer depends on the edge set alone.

Diagnostics and evaluation of rewiring. Over-squashing, effective resistance, and curvature point to bottlenecks where edges should be added (Alon & Yahav, 2021; Black et al., 2023; Topping et al., 2022). ESNR estimates how noisy a graph is relative to its features or labels (Dong & Kluger, 2023), and AdaEdge edits edges by the predicted labels of their endpoints (Chen et al., 2020). None of them says when the weights of the existing support would suffice. Several studies report that rewiring gains depend on the protocol. ComFy shows that rewiring that lowers the spectral gap can help as much as rewiring that raises it (Rubio-Madrigal et al., 2025), reported gains can be artifacts of hyperparameter sweeps (Tori et al., 2025), and gains credited to structure learning can come from the inner optimization (Kim & Jang, 2026). Our measurements add a controlled comparison. Both operations share the candidate set and the backbone, and the attack alone reverses which of them wins.

3 Problem Formulation

We consider semi-supervised node classification on an observed undirected graph $G_0 = (V, E_0)$ with node features X and labels on a subset $V_{\mathrm{tr}} \subset V$ over C classes. One such graph is an *instance*.

Compared with a latent clean graph $G^\star$, the observed edge set contains spurious links and lacks true ones. A self-supervised link predictor proposes a set E^+ of node pairs, and the candidate support is $E_c = E_0 \cup E^+$. Repair acts on E_c in one of two ways. The *reweighting class*

$$\mathcal{W}(E_c) = \{\mathrm{diag}(w) : w_e > 0,\ e \in E_c\} \tag{1}$$

contains the positive diagonal scalings of the candidate edges. Every member preserves the support and hence its incidence matrix. Weights that approach zero approach deletion, so every statement below carries the spread $w_{\max}/w_{\min}$ explicitly. *Rewiring* is everything else. It deletes or adds edges with unit weights, and in doing so changes the incidence matrix. Signed or matrix-valued weights lie outside both classes.

We study two questions. *First, before any repair is trained, can one decide from the observed graph and a classifier fitted to it whether some element of $\mathcal{W}(E_c)$ fits the evidence that the classifier places on the edges?* Theorem 1 answers it. *Second, does that decision predict which operation gives the more accurate classifier on the instance?* Section 5 answers it by measurement.

4 METHOD

This section builds the diagnostic and proves what reweighting can do to it. Section 4.1 reads a frozen classifier as an edge flow and decomposes it. Section 4.2 proves the certificate. Section 4.3 turns the certificate into the Hodge Router, a repair method that chooses the operation per instance.

4.1 PREDICTION FLOW

Flow. Let $p_i \in \Delta^{C-1}$ be the output distribution of a frozen classifier at node i, and fix a reference orientation for every candidate edge. Placing the difference of a node score on each edge would give a gradient by construction, which can never register the inconsistency of Figure 1a. We instead compare the two endpoint distributions bilinearly. Given an antisymmetric $K \in \mathbb{R}^{C\times C}$, we define

$$y_{i\to j} = \tfrac{1}{2}\, p_i^\top K p_j, \qquad y_{j\to i} = -y_{i\to j}. \tag{2}$$

The entry K_{ab} is the value of comparing a node of class a with a node of class b. If each endpoint's class is drawn from its prediction, $y_{i\to j}$ is half the expected value of that comparison. Its sum around a cycle need not vanish, and this circulation is what lets the flow detect an inconsistent support.

Which comparisons circulate. If $K = h\mathbf{1}^\top - \mathbf{1}h^\top$ for some $h \in \mathbb{R}^C$, then $y_{i\to j} = \frac{1}{2}(h^\top p_i - h^\top p_j)$ because each p_i sums to one, and the flow is a gradient for every classifier. This is the only obstruction.

Proposition 1 (Circulation criterion)**.** *Fix a support that contains a cycle. The flow of Eq. (2) is a gradient for every choice of $p_i \in \Delta^{C-1}$ if and only if $K = h\mathbf{1}^\top - \mathbf{1}h^\top$ for some $h \in \mathbb{R}^C$. For every other antisymmetric K there are one-hot predictions whose circulation around that cycle is nonzero.*

The excluded matrices form a subspace of dimension $C-1$ inside the antisymmetric matrices, which have dimension $C(C-1)/2$. The two coincide when $C = 2$, so the construction needs at least three classes. We order the classes by their training frequency and set $K_{ab} = \mathrm{sign}(b-a)$, the simplest order-based comparison that is not a difference of class scores. Proofs are in Appendix A.3, and Appendix B discusses other choices of K.

Decomposition and diagnostics. Let $B \in \mathbb{R}^{|V|\times|E_c|}$ be the incidence matrix of the candidate graph, with $B_{ie} = -1$ and $B_{je} = 1$ for $e = (i \to j)$. Hodge decomposition writes

$$BB^\top\phi^\star = By, \qquad y_{\mathrm{grad}} = B^\top\phi^\star, \qquad y_{\mathrm{cyc}} = y - y_{\mathrm{grad}}. \tag{3}$$

The potential $\phi^\star$ is the best single score per node for explaining the comparisons, and the residual y_{cyc} lies in the cycle space $\ker B$, which no potential on this support reaches. We summarize it by

$$\rho = \frac{\|y_{\mathrm{cyc}}\|_2^2}{\|y\|_2^2 + \varepsilon}, \qquad \kappa_{\mathrm{diag}} = \frac{\rho}{\beta_1/|E_c| + \varepsilon}, \qquad \beta_1 = |E_c| - |V| + \beta_0, \tag{4}$$

with β_0 the number of connected components. The cycle space has dimension β_1 and grows with density, so even a random flow has a large cycle fraction on a dense support. The normalized value κ_{diag} measures ρ against that chance level.

Proposition 2 (Chance level). *If $y \in \mathbb{R}^{|E_c|}$ has a distribution invariant under orthogonal transformations and $\Pr(y = 0) = 0$, then $\mathbb{E}[\rho] = \beta_1/|E_c|$ with $\varepsilon = 0$.*

4.2 Reweighting Invariance

Fix a flow y and a weighting $W \in \mathcal{W}(E_c)$. The best potential fit under W leaves the weighted residual

$$R_W(y) = \min_{\phi} \|y - B^\top \phi\|_W^2, \qquad \|x\|_W^2 = x^\top W x. \tag{5}$$

A method that learns edge weights on a fixed support changes this quantity only through W. We call $R_W(y)/\|y\|_W^2$ the weighted residual fraction. Write $\rho_0 = \|y_{\text{cyc}}\|_2^2/\|y\|_2^2$ for the unweighted cycle fraction, which is ρ at $\varepsilon = 0$, and $\gamma = w_{\max}/w_{\min}$ for the spread of W. Substituting $W^{1/2}$ into the least squares problem gives

$$R_W(y) = \left\|(I - P_W)W^{1/2} y_{\text{cyc}}\right\|_2^2, \qquad P_W = \text{orthogonal projection onto } \operatorname{im}(W^{1/2}B^\top), \tag{6}$$

because $(I - P_W)W^{1/2}B^\top = 0$. Whatever weights are chosen, the fit acts only on y_{cyc}.

Theorem 1 (Reweighting bound). *For every nonzero flow y and every $W \in \mathcal{W}(E_c)$ with spread γ,*

$$\frac{\rho_0}{\gamma} \;\le\; \frac{R_W(y)}{\|y\|_W^2} \;\le\; \gamma\,\rho_0, \tag{7}$$

and at uniform weights the ratio equals ρ_0. Moreover, $R_W(y) = 0$ for some $W \in \mathcal{W}(E_c)$ if and only if $y_{\text{cyc}} = 0$, if and only if $R_W(y) = 0$ for every $W \in \mathcal{W}(E_c)$.

Corollary 1 (Rewiring certificate). *Fix $\delta \ge 0$ and $\gamma \ge 1$. If $\rho_0 > \gamma\delta$, then $R_W(y)/\|y\|_W^2 > \delta$ for every $W \in \mathcal{W}(E_c)$ with spread at most γ. In particular, if $y_{\text{cyc}} \neq 0$, an exact potential fit requires changing the support.*

The proof uses two facts, that positive reweighting preserves $\operatorname{im}(B^\top)$ and that the two norms differ by at most the factor γ (Appendix A). The theorem covers every method that learns strictly positive edge weights of bounded spread on a fixed support. The certificate is computed once, before training, and learned weights cannot change its answer. The theorem leaves open how large ρ_0 is on real prediction flows, and whether the instances whose flow no weighting fits are those on which rewiring pays. Section 5 measures both.

4.3 The Hodge Router

The Hodge Router acts on the certificate (Figure 1b). Besides the frozen classifier and the incidence matrix it needs a self-supervised link predictor, adapted from the graph propensity score (GPS) rewiring of Dong & Kluger (2023), which scores every observed edge and proposed pair by $g_e \in (0, 1)$ from its two endpoints and fills E^+ up to a density budget $|E_c| \le \nu|E_0|$.

Residual gate. Each instance is routed by its diagnostic. If $\kappa_{\text{diag}} < \tau$, the support is kept and positive diagonal weights are learned on E_c, which we call Reweight. Otherwise the support is edited with unit weights, which we call Rewire. On one instance κ_{diag} is ρ_0 divided by the chance level $\beta_1/|E_c|$, so the gate compares ρ_0 with τ times the chance level of that instance. The threshold τ is selected per dataset on tuning instances from a quantile grid fixed in advance. Instances with $\beta_1 = 0$ or $C = 2$ have a gradient flow and are reweighted.

Cycle-aware ranking. On a rewired instance, the link score and the cycle contribution are normalized by rank within the instance,

$$\hat{g}_e = \operatorname{rank}(g_e)/|E_c|, \qquad \hat{c}_e = \operatorname{rank}\big((y_{\text{cyc}})_e^2\big)/|E_c|, \tag{8}$$

and combined into the structural score

$$u_e = \hat{g}_e - \hat{c}_e. \tag{9}$$

The q observed edges with the lowest u_e are deleted and the q proposed pairs with the highest u_e are admitted, where q is the edit budget. A ranking by $\hat{g}_e$ alone is plain link-score rewiring, which the GPS rows of our tables report. The cycle term moves an edge that looks plausible locally but carries much of the residual down the ranking.

Arbiter. The two terms of u_e can cancel, and then the structure has little basis for a decision. When the graph carries node text, a language model reads the two endpoints of the $\mathcal{B}$ candidates on which the terms cancel most, and its verdict overrides u_e on those edges alone. The verdicts change the edit set in at most $2\mathcal{B}$ edges (Proposition 4). Appendix B.3 gives the disagreement score, Algorithm 1, and the cost of each stage, Appendix A.3 the proof, and Appendix E the prompt.

5 Experiments

The experiments ask three questions. **Q1** asks how large the cycle part of a real prediction flow is, and how much reweighting moves it. **Q2** asks whether the residual predicts, instance by instance, which operation gives the more accurate classifier. **Q3** asks which operation helps under which perturbation, and how the Hodge Router compares with published repair methods.

5.1 Setup

Data and perturbations. We use the citation graphs Cora, CiteSeer, and PubMed (Sen et al., 2008), the political blog graph PolBlogs (Adamic & Glance, 2005), and the citation graph DBLP (Bojchevski & Günnemann, 2018), with the Pro-GNN protocol and a 10/10/80 split on the largest connected component (Jin et al., 2020). PolBlogs has two classes, so its flow is a gradient by Proposition 1, and it enters only the random corruption and cost tables. Random corruption applies uniform deletion and insertion, deletion of the edges of highest betweenness, or insertion between dissimilar nodes of different classes, at rates $\eta \in \{0.05, 0.10, 0.15, 0.20\}$. With ten seeds this gives 120 instances on each of Cora, CiteSeer, and PubMed. DBLP adds ten instances of a mixed mechanism at $\eta = 0.15$, for 370 instances in the diagnostic analysis. ogbn-arxiv (Hu et al., 2020) enters only the density and cost studies. The adversarial protocols run on the four citation graphs. Metattack (Zügner & Günnemann, 2019) uses the perturbed graphs released with Pro-GNN, and on DBLP, which has no released graphs, the Meta-Self variant is run at the same rates. Projected randomized block coordinate descent (PRBCD) (Geisler et al., 2021) is transferred from a surrogate GCN. Both attacks perturb 5%, 15%, and 25% of the edges. Table 4 and Appendix D give the statistics and generators.

Flow and policies. Our link scorer adapts GPS and proposes candidates up to $|E_c| \le \nu|E_0|$ with $\nu = 2$. A two layer GCN (Kipf & Welling, 2017) with 256 hidden units is trained on the perturbed graph and frozen, and its predictions define the flow. Reweight learns $w_e = \exp(\theta_e)$ on all of E_c jointly with a fresh GCN, with θ_e clipped to $[-\ln 4, \ln 4]$, so the spread never exceeds 16. Rewire edits the support by Eq. (9) with $q = \lfloor b|E_0| \rfloor$ deletions and q additions. The budget factor b equals the perturbation rate, which the gate never reads, and every method that edits the graph receives the same q. Both operations share the candidate set and the backbone. Always-Reweight and Always-Rewire apply one operation to every instance. The residual gate routes by κ_{diag} without the arbiter, and the Hodge Router adds the arbiter with $\mathcal{B} = 50$ and $\lambda = 2$. The threshold τ is selected on random corruption instances with seeds 100 to 104, disjoint from the evaluation seeds, and is reused unchanged under both attacks. Controls that replace κ_{diag} by a random permutation of its decisions, by the perturbation rate, by the share of edges with differing endpoint predictions (the disagreement gate), or by the label-based edge signal-to-noise ratio (ESNR) (Dong & Kluger, 2023) use the same selection rule. The rate gate, the ESNR gate, and the per instance oracle are references.

Baselines. Under Metattack we compare with GCN, GAT (Veličković et al., 2018), RGCN, GCN-Jaccard, GCN-SVD, Pro-GNN, PTDNet, CoGSL, SLAPS, JDR, GPS, ComFy (Rubio-Madrigal et al., 2025), and LLM4RGNN (Zhang et al., 2025b). Every baseline runs from its official implementation under the Pro-GNN protocol, every repair method uses a two layer GCN backbone, and every method that edits the graph receives the edit budget q. Appendix D gives the remaining details and the arbiter configuration.

Metrics and criteria. The operation gap of an instance is $g^* = \mathrm{F1}(\textsc{Rewire}) - \mathrm{F1}(\textsc{Reweight})$ in Macro-F1 on the test split. Policy comparisons under random corruption report Macro-F1 pooled over the mechanism and rate cells of each dataset, and the attack tables report accuracy, the metric of the Pro-GNN protocol. Results are mean and standard deviation over ten seeds. Density is $|E_c|/|V|$. Confidence intervals come from a bootstrap over whole cells of one dataset, mechanism, and rate, and Holm correction (Holm, 1979) runs within each table. We fixed two criteria in advance. For Q2, the partial Spearman correlation of κ_{diag} with g^*, controlling for rate, density, β_1, and dataset, must have an interval that excludes zero and must exceed that of every competing diagnostic. For routing to pay, the residual gate must match or exceed the better fixed policy on every dataset, and exceed both fixed policies and the random, disagreement, and ESNR gates on average.

5.2 Q1. How Large Is the Cycle Part, and Does Reweighting Move It?

The bound holds, and weights move the residual little. Table 5 in Appendix F reports 1,000 positive diagonal weightings of spread 16 on one instance of each dataset. None of the 4,000 weighted residual fractions leaves the interval of Theorem 1. The weightings also move the fraction far less than the interval allows. On Cora the unweighted fraction is $\rho_0 = 0.100$, the interval runs from 0.006 to 1.606, and every draw lies between 0.085 and 0.101. The other datasets behave the same way. The REWEIGHT learner reaches spreads between 1.10 and 7.21, well inside the cap of 16 set by its clip. Reweighting thus leaves the cycle part where it found it, as the theorem predicts.

Corruption concentrates the cycle part. The lower block of Table 5 shows how large the cycle part is. At $\nu = 2$ the cycle fraction is 0.099 on Cora, 0.052 on CiteSeer, and 0.021 on PubMed. A flow with i.i.d. standard Gaussian entries on the same supports reaches 0.754, 0.708, and 0.783, in line with the chance level of Proposition 2. The flow of the shuffled predictions reaches 0.123, 0.080, and 0.052. Most of the flow therefore follows the class order that K imposes. Corruption concentrates what remains. From the clean to the perturbed graph the diagnostic rises from 0.083 to 0.131 on Cora, from 0.048 to 0.073 on CiteSeer, and from 0.015 to 0.026 on PubMed. Raising ν to 8 on Cora lifts $\beta_1/|E_c|$ from 0.756 to 0.939 while κ_{diag} moves only from 0.131 to 0.122 (Table 6), so a dense candidate graph does not push the diagnostic toward chance.

5.3 Q2. Does the Residual Predict the Better Operation?

Diagnostic	r_s	95% CI	partial r_s	95% CI	AUROC	AUPRC	ECE	Regret	N
ρ (ours)	0.29	[0.08, 0.47]	0.27	[0.07, 0.43]	0.64	0.61	0.09	0.16	370
κ_{diag} (ours)	0.41	[0.22, 0.57]	0.38	[0.21, 0.52]	0.72	0.68	0.06	0.09	370
Prediction disagreement	0.16	[-0.08, 0.39]	0.07	[-0.12, 0.25]	0.56	0.54	0.11	0.30	370
Perturbation rate	0.36	[0.09, 0.57]	0.31	[0.08, 0.50]	0.71	0.69	0.08	0.49	370
ESNR	0.11	[-0.10, 0.32]	0.09	[-0.08, 0.25]	0.57	0.55	0.12	0.38	370
Assortativity	-0.06	[-0.30, 0.15]	0.03	[-0.15, 0.20]	0.47	0.49	0.12	0.32	370
Feature smoothness	0.08	[-0.15, 0.28]	0.05	[-0.14, 0.21]	0.54	0.53	0.12	0.39	370
Feature homophily	0.04	[-0.19, 0.27]	0.02	[-0.18, 0.18]	0.51	0.50	0.11	0.34	370
Spectral gap	-0.03	[-0.26, 0.18]	-0.04	[-0.22, 0.12]	0.48	0.47	0.13	0.39	370

Table 1: Prediction of the operation gap $g^* = \mathrm{F1}(\textsc{Rewire}) - \mathrm{F1}(\textsc{Reweight})$ over the 370 random corruption instances, with Macro-F1 on the $[0, 1]$ scale. r_s is the Spearman correlation with the gap, and the partial correlation controls for corruption rate, candidate density, β_1, and dataset. Competing diagnostics are oriented so that larger values predict REWIRE. Prediction disagreement is the share of candidate edges whose endpoint predictions differ under the same frozen classifier, without any decomposition. The perturbation rate and ESNR are references, since the gate never reads the rate and ESNR uses the labels of all nodes. For the rate, the partial correlation controls for the remaining covariates. AUROC, AUPRC, and ECE score the prediction of $g^* > 0$, with the calibration fitted on the tuning instances. Regret is the Macro-F1 (%) lost by the threshold rule relative to the per instance oracle on the test split, averaged over the four datasets of Table 2.

The residual ranks instances by the operation they need. Over the 370 random corruption instances, the partial Spearman correlation of κ_{diag} with the operation gap is 0.38, with an interval of $[0.21, 0.52]$ (Table 1). The unnormalized fraction ρ reaches 0.27, so the calibration by chance level

Policy	Cora	CiteSeer	PubMed	DBLP	Rewire frac.
Always-Reweight	80.68±0.20	61.58±0.44	83.62±0.11	73.33±0.76	0.000
Always-Rewire	80.24±0.26	61.05±0.46	84.17±0.14	71.83±0.71	1.000
Random gate	80.47±0.23	61.36±0.45	83.96±0.12	72.95±0.74	0.490
Rate gate	80.31±0.22	61.26±0.38	83.98±0.12	72.84±0.73	0.514
Disagreement gate	80.62±0.16	61.58±0.44	83.87±0.12	73.08±0.73	0.449
ESNR gate	80.51±0.23	61.40±0.45	83.93±0.12	73.02±0.73	0.491
Residual gate	80.88±0.23	61.78±0.45	84.21±0.12	73.41±0.72	0.490
Hodge Router	81.25±0.23	62.13±0.45	84.52±0.12	73.41±0.72	0.490
Per instance oracle	80.96±0.13	61.85±0.44	84.23±0.16	73.58±0.70	0.504

Table 2: Macro-F1 (%, mean±std) of routing policies at matched edit budget under random corruption, pooled over the mechanism and rate cells of each dataset. Every threshold is selected per dataset on tuning instances. The random gate permutes the residual gate's decisions within a dataset. The disagreement gate thresholds prediction disagreement. The oracle picks the better fixed policy for each instance with hindsight, the rate gate reads the perturbation rate, and the ESNR gate reads the labels of all nodes, so these three are references rather than deployable policies. Rewire frac. is the share of the 370 instances routed to REWIRE. On DBLP, which has no node text, the Hodge Router coincides with the residual gate. PolBlogs appears in Table 9.

adds signal. The raw correlation of κ_{diag} is 0.41, above the 0.36 of the perturbation rate, which the gate never reads. The criterion of Q2 is met. The strata of Table 7 show a positive correlation in every stratum, significant after Holm correction in seven of the thirteen.

Routing on the residual beats both fixed operations. The mean gap g^* lies within 1.5 points of Macro-F1 of zero in every random corruption stratum. In Table 2 the two fixed policies differ by at most 1.50 points on any dataset, and the per instance oracle exceeds the better of them by at most 0.28 points. The residual gate recovers most of that margin. It matches or exceeds the better fixed policy on all four datasets, with 80.88 on Cora, 61.78 on CiteSeer, 84.21 on PubMed, and 73.41 on DBLP, and its regret to the oracle is 0.09 points, against 0.16 for ρ and 0.30 to 0.49 for the other diagnostics. The random gate rewires the same share of instances but falls below the better fixed policy on every dataset. The gain therefore comes from which instances are rewired, not from how many. With the arbiter, the Hodge Router reaches 81.25, 62.13, and 84.52 on the three graphs with node text. On all three this passes the oracle. The selected thresholds lie inside the grid, where validation and test Macro-F1 peak together (Table 8).

5.4 Q3. WHICH OPERATION HELPS UNDER WHICH ATTACK?

The Hodge Router is the most accurate method under Metattack. In Table 3 the Hodge Router averages 78.7% over the twelve cells, against 77.1% for JDR, the strongest baseline, 76.0% for Pro-GNN, and 77.7% for Always-Rewire. It is the best method in eight of the twelve cells and second in three more, and it leads JDR by up to 4.4 points. Metattack inserts most of its edges between classes and lifts κ_{diag} above the threshold selected under random corruption. The residual gate therefore rewires 113 of the 120 instances without seeing the attack and stays within 0.4 points of Always-Rewire in every cell (Table 13). The arbiter adds 0.4 to 1.0 points at 5% and 2.1 to 2.6 points at 25% on the three graphs with node text.

The cycle term improves rewiring. REWIRE and GPS share the candidate set and budget and differ only in the cycle term of Eq. (9). Under Metattack REWIRE leads GPS in all twelve cells, by 1.9 to 4.7 points at 5% and by 6.9 to 16.7 points at 25%. On Cora at 15% REWIRE deletes 80% of the inserted edges, against 30% for GPS, and the Hodge Router deletes 85% (Table 11).

PRBCD favors reweighting on Cora and CiteSeer. PRBCD is the harder attack. GCN falls to 53.2% on Cora at 15%, against 67.6% under Metattack, and the order of the fixed policies reverses on Cora and CiteSeer (Table 10 in Appendix F). Always-Reweight leads Always-Rewire at 15% and 25% by 4.2 and 3.4 points on Cora and by 6.3 and 6.8 points on CiteSeer. Here the residual gate chooses REWEIGHT on 58 of the 60 instances and stays within 0.1 points of Always-Reweight. On PubMed rewiring leads by 3.9 points at 15% and 6.4 points at 25%, and there the gate rewires most instances and the router leads Always-Reweight by 0.5 to 7.1 points. DBLP is mixed, with

Method	Cora			CiteSeer		
	5%	15%	25%	5%	15%	25%
GCN	79.2±0.3	67.6±0.6	52.1±0.2	73.0±0.3	66.3±1.1	58.5±0.5
GAT	78.6±1.0	67.7±1.8	52.6±2.4	72.4±2.4	67.9±1.6	60.6±2.3
RGCN	77.4±0.4	68.8±0.4	57.5±0.8	70.5±0.4	65.7±0.4	55.4±0.7
GCN-Jaccard	79.1±0.6	71.0±0.6	60.8±1.1	70.5±1.0	66.0±0.9	59.9±1.5
GCN-SVD	78.4±0.5	66.7±1.2	52.1±1.2	68.8±0.7	63.3±1.0	57.2±1.9
Pro-GNN	82.3±0.5	76.4±1.3	69.7±1.7	72.9±0.6	72.0±1.1	69.0±2.8
PTDNet	77.4±0.5	64.2±0.4	58.5±0.5	71.9±0.4	66.4±0.2	57.9±0.9
CoGSL	79.4±0.2	68.6±0.6	60.0±2.2	75.9±0.1	70.8±0.2	61.4±0.7
SLAPS	79.4±0.4	69.7±0.8	55.5±0.8	72.8±0.9	68.2±0.7	61.8±0.8
JDR	81.9±1.2	75.2±1.1	74.0±1.7	74.3±0.6	73.6±0.0	71.2±0.2
GPS	78.0±0.2	69.4±0.4	57.3±0.7	72.2±0.7	66.2±0.7	60.8±0.8
ComFy	79.0±0.4	70.1±0.7	58.0±0.7	72.5±0.8	67.0±0.8	61.6±0.6
LLM4RGNN	80.5±0.4	71.4±0.6	57.8±0.8	73.4±0.8	69.1±0.7	63.3±0.7
Always-Reweight	79.4±0.3	70.6±0.5	51.1±0.8	71.3±0.5	69.5±0.5	62.1±0.6
Always-Rewire	82.3±0.3	77.0±0.7	74.0±0.6	75.8±1.0	73.2±0.9	70.0±0.7
Hodge Router	**82.7**±0.3	**78.1**±0.6	**76.3**±0.7	**76.3**±0.7	**74.5**±0.7	**72.6**±0.6

Method	PubMed			DBLP			Avg.
	5%	15%	25%	5%	15%	25%	
GCN	83.3±0.0	78.5±0.0	74.5±0.1	73.3±0.7	68.1±0.2	57.1±0.8	69.3
GAT	81.8±0.6	77.0±0.7	72.6±1.2	77.8±2.2	68.6±2.8	51.5±2.1	69.1
RGCN	81.1±0.2	73.9±0.3	68.0±0.2	72.0±0.2	66.4±1.3	58.2±0.8	67.9
GCN-Jaccard	86.4±0.1	84.8±0.1	83.7±0.1	71.3±1.4	65.7±1.8	58.3±1.6	71.5
GCN-SVD	83.4±0.2	83.1±0.2	82.7±0.2	**80.1**±0.8	65.1±0.9	57.6±0.4	69.9
Pro-GNN	**87.6**±0.1	86.8±0.2	85.8±0.2	77.1±2.0	70.3±2.4	61.9±1.9	76.0
PTDNet	85.4±0.1	85.0±0.2	83.2±0.1	71.6±0.8	67.0±0.2	55.8±0.5	70.4
CoGSL	85.1±0.1	81.2±0.1	78.1±0.1	68.9±0.4	60.7±2.0	56.8±1.1	70.6
SLAPS	84.0±0.3	80.7±0.4	78.2±0.9	74.1±0.9	67.4±1.1	58.9±1.3	70.9
JDR	87.0±0.1	86.7±0.0	86.6±0.1	74.7±0.0	72.0±1.7	68.0±2.4	77.1
GPS	84.5±0.6	80.9±1.1	78.1±1.7	74.6±0.8	68.3±0.9	60.2±1.2	70.9
ComFy	84.7±0.2	81.0±0.5	78.2±0.8	75.2±0.7	68.9±1.0	60.7±1.1	71.4
LLM4RGNN	84.7±0.3	81.6±0.4	79.7±1.1	–	–	–	–
Always-Reweight	83.4±0.1	79.1±0.4	76.3±1.0	74.9±0.7	69.0±0.9	59.6±1.2	70.5
Always-Rewire	86.4±0.3	86.3±0.4	85.0±0.5	79.3±0.8	**73.8**±1.0	**69.6**±1.3	77.7
Hodge Router	87.4±0.2	**87.3**±0.4	**87.1**±0.7	79.1±0.8	73.6±1.0	69.2±1.3	**78.7**

Table 3: Accuracy (%, mean±std over ten runs) under Metattack at 5%, 15%, and 25% perturbation with the Pro-GNN split. Every repair method uses a two layer GCN backbone. LLM4RGNN needs node text and is not run on DBLP, so its average is left out. Best in bold and second best underlined, per column. Avg. is the mean over the twelve cells.

reweighting ahead at 5% and rewiring ahead at 15% and 25%, and the gate follows the switch. The router has the best average, 69.7%, against 68.4% for Always-Reweight and 67.1% for JDR.

6 Conclusion

We separated graph repair into reweighting, which rescales a fixed support, and rewiring, which changes it. Read as a flow on the edges, the predictions of a frozen classifier split into a gradient part and a cycle part. No positive reweighting lets one score per node explain the cycle part, and weights of spread γ move its share by at most a factor γ. A nonzero cycle part therefore certifies, before any repair is trained, that an exact fit needs rewiring. On real graphs the bound holds, and the calibrated cycle fraction predicts which operation helps. The Hodge Router follows the operation each attack favors and has the best average accuracy under both attacks. A method that only reweights, or only rewires, looks strong or weak depending on the attack, so repair methods should be compared with both operations on a shared candidate set and budget, and under more than one attack.

AI USE STATEMENT

Generative AI tools were used only to polish the writing of this paper. The authors reviewed all edited text and take responsibility for the final content.

ETHICS STATEMENT

All datasets are public benchmarks that have been used in prior work under the same protocol. PolBlogs labels the political leaning of blogs and contains no information about individuals beyond what the blogs published. The arbiter sends only the public title and abstract of two documents to a language model, and never sends labels, splits, or user data. The only human judgments in the study are those of two of the authors, who judged the plausibility of a sample of audited edges from the endpoint texts. The method repairs graphs for node classification and does not raise concerns beyond those of the task it serves.

REPRODUCIBILITY STATEMENT

Complete proofs of Theorem 1, Corollary 1, and Propositions 1 to 4 are given in Appendix A. Section 5.1 states the datasets, perturbations, policies, baselines, and metrics, and Appendix D gives every generator, architecture, training schedule, threshold selection rule, and statistical procedure. The prompt is reproduced verbatim in Appendix E. The code, with the configuration and seeds of every run and the commands that produce each table, is provided as supplementary material.

REFERENCES

Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 U.S. election: Divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, pp. 36–43, 2005.

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *Proceedings of the International Conference on Learning Representations*, 2021.

Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56 (1–3):89–113, 2004.

Mitchell Black, Zhengchao Wan, Amir Nayyeri, and Yusu Wang. Understanding oversquashing in GNNs through the lens of effective resistance. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2528–2547, 2023.

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 3950–3957, 2021.

Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Liò, and Michael M. Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in GNNs. In *Advances in Neural Information Processing Systems*, volume 35, pp. 18527–18541, 2022.

Aleksandar Bojchevski and Stephan Günnemann. Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *Proceedings of the International Conference on Learning Representations*, 2018.

Ozan Candogan, Ishai Menache, Asuman Ozdaglar, and Pablo A. Parrilo. Flows and decompositions of games: Harmonic and potential games. *Mathematics of Operations Research*, 36(3):474–503, 2011.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3438–3445, 2020.

Shuo Chen and Thorsten Joachims. Modeling intransitivity in matchup and comparison data. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*, pp. 227–236, 2016.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. Exploring the potential of large language models (LLMs) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61, 2024a. doi: 10.1145/3655103.3655110.

Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. Label-free node classification on graphs with large language models (LLMs). In *Proceedings of the International Conference on Learning Representations*, 2024b.

Mingze Dong and Yuval Kluger. Towards understanding and reducing graph structural noise for GNNs. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8202–8226. PMLR, 2023.

Negin Entezari, Saba A. Al-Sayouri, Amirali Darvishzadeh, and Evangelos E. Papalexakis. All you need is low (rank): Defending against adversarial attacks on graphs. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining*, pp. 169–177, 2020. doi: 10.1145/3336191.3371789.

Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. SLAPS: Self-supervision improves structure learning for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 22667–22681, 2021.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized PageRank. In *Proceedings of the International Conference on Learning Representations*, 2019.

Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. In *Advances in Neural Information Processing Systems*, volume 34, pp. 7637–7649, 2021.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, pp. 1024–1034, 2017.

Frank Harary. On the notion of balance of a signed graph. *Michigan Mathematical Journal*, 2(2): 143–146, 1953.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning. In *Proceedings of the International Conference on Learning Representations*, 2024.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard T. B. Ma, Hongzhi Chen, and Ming-Chang Yang. Measuring and improving the use of graph information in graph neural networks. In *Proceedings of the International Conference on Learning Representations*, 2020.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22118–22133, 2020.

Junteng Jia, Michael T. Schaub, Santiago Segarra, and Austin R. Benson. Graph-based semi-supervised & active learning for edge flows. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 761–771, 2019. doi: 10.1145/3292500.3330872.

Xiaoye Jiang, Lek-Heng Lim, Yuan Yao, and Yinyu Ye. Statistical ranking and combinatorial Hodge theory. *Mathematical Programming*, 127(1):203–244, 2011.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 66–74, 2020. doi: 10.1145/3394486.3403049.

Minkyoung Kim and Beakcheol Jang. Bilevel graph structure learning, revisited: Inner-channel origins of the reported gain. *arXiv preprint arXiv:2605.07577*, 2026.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations*, 2017.

Jintang Li, Jiaying Peng, Liang Chen, Zibin Zheng, Tingting Liang, and Qing Ling. Spectral adversarial training for robust graph neural network. *IEEE Transactions on Knowledge and Data Engineering*, 35(9):9240–9253, 2023. doi: 10.1109/TKDE.2022.3222207.

Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. DeepRobust: A PyTorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020.

Lek-Heng Lim. Hodge Laplacians on graphs. *SIAM Review*, 62(3):685–715, 2020.

Jonas Linkerhägner, Cheng Shi, and Ivan Dokmanić. Joint graph rewiring and feature denoising via spectral resonance. In *Proceedings of the International Conference on Learning Representations*, 2025.

Nian Liu, Xiao Wang, Lingfei Wu, Yu Chen, Xiaojie Guo, and Chuan Shi. Compact graph structure learning via mutual information compression. In *Proceedings of the ACM Web Conference*, pp. 1601–1610, 2022. doi: 10.1145/3485447.3512206.

Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 779–787, 2021. doi: 10.1145/3437963.3441734.

M. E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126, 2003.

Braxton Osting, Christoph Brune, and Stanley Osher. Enhanced statistical rankings via targeted data collection. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 489–497. PMLR, 2013.

Celia Rubio-Madrigal, Adarsh Jamadandi, and Rebekka Burkholz. GNNs getting ComFy: Community and feature similarity guided rewiring. In *Proceedings of the International Conference on Learning Representations*, 2025.

Michael T. Schaub, Austin R. Benson, Paul Horn, Gabor Lippner, and Ali Jadbabaie. Random walks on simplicial complexes and the normalized Hodge 1-Laplacian. *SIAM Review*, 62(2):353–391, 2020.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

Shengyin Sun, Yuxiang Ren, Jiehao Chen, and Chen Ma. Large language models as topological structure enhancers for text-attributed graphs. In *Database Systems for Advanced Applications (DASFAA 2025)*, volume 15991 of *Lecture Notes in Computer Science*, pp. 106–122. Springer, 2025. doi: 10.1007/978-981-95-4158-4_7.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *Proceedings of the International Conference on Learning Representations*, 2022.

Floriano Tori, Vincent Holst, and Vincent Ginis. The effectiveness of curvature-based rewiring and the role of hyperparameters in GNNs revisited. In *Proceedings of the International Conference on Learning Representations*, 2025.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proceedings of the International Conference on Learning Representations*, 2018.

Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: Deep insights into attack and defense. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 4816–4823, 2019. doi: 10.24963/ijcai.2019/669.

Christopher Zach, Manfred Klopschitz, and Marc Pollefeys. Disambiguating visual relations using loop constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1426–1433, 2010.

Xiang Zhang and Marinka Zitnik. GNNGuard: Defending graph neural networks against adversarial attacks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9263–9275, 2020.

Zhihan Zhang, Xunkai Li, Lei Zhu, Guang Zeng, Bowen Fan, Yanzhe Wen, Hongchao Qin, Rong-Hua Li, and Guoren Wang. Unlocking graph structure learning with tree-guided large language models. *arXiv preprint arXiv:2503.21223*, 2025a. v5, revised January 2026.

Zhongjian Zhang, Xiao Wang, Huichi Zhou, Yue Yu, Mengmei Zhang, Cheng Yang, and Chuan Shi. Can large language models improve the adversarial robustness of graph neural networks? In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2008–2019, 2025b. doi: 10.1145/3690624.3709256.

Panpan Zhou, Kaibing Yin, Haowei Zhu, Yan Liang, Liwei Li, and Yuhua Chen. A survey on graph structure learning. *Neurocomputing*, 696:134114, 2026. doi: 10.1016/j.neucom.2026.134114.

Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1399–1407, 2019. doi: 10.1145/3292500.3330851.

Dongcheng Zou, Hao Peng, Xiang Huang, Renyu Yang, Jianxin Li, Jia Wu, Chunyang Liu, and Philip S. Yu. SE-GSL: A general and effective graph structure learning framework through structural entropy optimization. In *Proceedings of the ACM Web Conference*, pp. 499–510, 2023. doi: 10.1145/3543507.3583453.

Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *Proceedings of the International Conference on Learning Representations*, 2019.

APPENDIX

A PROOFS

A.1 NOTATION

Let $B \in \mathbb{R}^{|V|\times|E_c|}$ be the incidence matrix of the candidate graph and $W = \mathrm{diag}(w)$ with $w_e > 0$. Write $\langle x, z\rangle_W = x^\top W z$, $\|x\|_W^2 = \langle x, x\rangle_W$, and $L_0 = BB^\top$. For weights in an interval $0 < w_{\min} \le w_e \le w_{\max}$ put $\gamma = w_{\max}/w_{\min} \ge 1$.

The unweighted decomposition of a flow $y \in \mathbb{R}^{|E_c|}$ is

$$y = B^\top \phi^\star + y_{\mathrm{cyc}}, \qquad B^\top \phi^\star = B^\top L_0^+ B y, \qquad y_{\mathrm{cyc}} = (I - B^\top L_0^+ B)y, \tag{10}$$

where L_0^+ is the pseudoinverse of L_0, so $y_{\mathrm{grad}} = B^\top \phi^\star \in \mathrm{im}(B^\top)$ and $y_{\mathrm{cyc}} \in \ker B$. The two subspaces are orthogonal in the standard inner product and span $\mathbb{R}^{|E_c|}$. Because $\mathrm{im}(B^\top)$ is the orthogonal complement of $\ker B$, the gradient y_{grad} is the unweighted least squares fit and $\|y_{\mathrm{cyc}}\|_2^2 = \min_\phi \|y - B^\top \phi\|_2^2$.

A.2 WEIGHTED RESIDUAL

Proposition 3 (Weighted residual identity)**.** *For every y and every positive diagonal W,*

$$R_W(y) = \min_\phi \|y - B^\top \phi\|_W^2 = \left\| (I - P_W)\, W^{1/2} y_{\mathrm{cyc}} \right\|_2^2, \tag{11}$$

where P_W is the orthogonal projection onto $\mathrm{im}(W^{1/2}B^\top)$ in the standard inner product.

Proof. Since $\|x\|_W^2 = \|W^{1/2}x\|_2^2$,

$$R_W(y) = \min_\phi \left\| W^{1/2} y - (W^{1/2}B^\top)\phi \right\|_2^2, \tag{12}$$

which is ordinary least squares for the matrix $W^{1/2}B^\top$. Its residual is the squared norm of the part of $W^{1/2}y$ orthogonal to $\mathrm{im}(W^{1/2}B^\top)$,

$$R_W(y) = \left\| (I - P_W) W^{1/2} y \right\|_2^2. \tag{13}$$

For any $v \in \mathbb{R}^{|V|}$ the vector $W^{1/2}B^\top v$ lies in $\mathrm{im}(W^{1/2}B^\top)$, so $(I - P_W)W^{1/2}B^\top v = 0$. Taking $v = \phi^\star$ gives $(I - P_W)W^{1/2} y_{\mathrm{grad}} = 0$. Substituting $y = y_{\mathrm{grad}} + y_{\mathrm{cyc}}$ into (13) yields (11). □

Proof of Theorem 1. Upper bound. $I - P_W$ is an orthogonal projection, so by (11)

$$R_W(y) \le \|W^{1/2} y_{\mathrm{cyc}}\|_2^2 = \sum_e w_e (y_{\mathrm{cyc}})_e^2 \le w_{\max} \|y_{\mathrm{cyc}}\|_2^2. \tag{14}$$

Also $\|y\|_W^2 = \sum_e w_e y_e^2 \ge w_{\min}\|y\|_2^2$. Dividing the two inequalities gives $R_W(y)/\|y\|_W^2 \le \gamma\rho_0$.

Lower bound. For every ϕ,

$$\|y - B^\top \phi\|_W^2 = \sum_e w_e (y - B^\top \phi)_e^2 \ge w_{\min} \|y - B^\top \phi\|_2^2. \tag{15}$$

Minimizing both sides over ϕ and using $\min_\phi \|y - B^\top \phi\|_2^2 = \|y_{\mathrm{cyc}}\|_2^2$ gives $R_W(y) \ge w_{\min}\|y_{\mathrm{cyc}}\|_2^2$. Also $\|y\|_W^2 \le w_{\max}\|y\|_2^2$. Dividing gives $R_W(y)/\|y\|_W^2 \ge \rho_0/\gamma$.

The zero does not depend on the weights. If $y_{\mathrm{cyc}} = 0$ then $R_W(y) = 0$ for every W by (11). Conversely, if $R_W(y) = 0$ for some positive diagonal W, the lower bound gives $w_{\min}\|y_{\mathrm{cyc}}\|_2^2 \le 0$, so $y_{\mathrm{cyc}} = 0$. Hence the three statements are equivalent.

Uniform weights. For $W = \alpha I$ with $\alpha > 0$ the projection P_W is the standard projection onto $\mathrm{im}(B^\top)$, so $R_{\alpha I}(y) = \alpha\|y_{\mathrm{cyc}}\|_2^2$ and $\|y\|_{\alpha I}^2 = \alpha\|y\|_2^2$. The weighted residual fraction equals ρ_0, and both bounds hold with equality at $\gamma = 1$. For $\gamma > 1$ neither bound need be attained. In the triangle of Appendix B the fraction $9w/((1+2w)(w+2))$ is maximized at $w = 1$, so no non-uniform weighting reaches ρ_0 there. □

Proof of Corollary 1. Let $W \in \mathcal{W}(E_c)$ have spread at most γ. By the lower bound of Theorem 1, $R_W(y)/\|y\|_W^2 \geq \rho_0/\gamma > \gamma\delta/\gamma = \delta$. For the second statement take $\delta = 0$. If $y_{\mathrm{cyc}} \neq 0$ then $\rho_0 > 0$, and every $W \in \mathcal{W}(E_c)$ has some finite spread γ, so $R_W(y) > 0$. The support determines B, and B determines $\ker B$, so for a fixed flow the only way to change ρ_0 is to change the support. □

A.3 Auxiliary Results

Proof of Proposition 1. If $K = h\mathbf{1}^\top - \mathbf{1}h^\top$, then for any $p_i, p_j \in \Delta^{C-1}$ we have $p_i^\top K p_j = (h^\top p_i)(\mathbf{1}^\top p_j) - (\mathbf{1}^\top p_i)(h^\top p_j) = h^\top p_i - h^\top p_j$, since each distribution sums to one. Hence $y = -\frac{1}{2}B^\top\psi$ with $\psi_i = h^\top p_i$ under the sign convention of Section 4.1, and the flow is a gradient for every classifier.

Conversely, suppose the flow is a gradient for every choice of distributions in the simplex. Then its circulation around every cycle of the support vanishes for every choice of predictions. Take a cycle $v_1, \ldots, v_L$ with $L \geq 3$ and one-hot predictions $p_{v_1} = e_i$, $p_{v_2} = e_j$, and $p_{v_l} = e_k$ for $l \geq 3$. Since $K_{kk} = 0$, the circulation is $\frac{1}{2}(K_{ij} + K_{jk} + K_{ki})$, so $K_{ij} + K_{jk} + K_{ki} = 0$ for all i, j, k. Fix $k = 1$ and put $h_i = K_{i1}$. Then $K_{ij} = -K_{j1} - K_{1i}$, and antisymmetry gives $K_{1i} = -h_i$, so $K_{ij} = h_i - h_j$. Hence $K = h\mathbf{1}^\top - \mathbf{1}h^\top$. If K is not of this form, some triple violates the identity, and the corresponding one-hot predictions have nonzero circulation around any cycle of the support. □

Proof of Proposition 2. Let $P = I - B^\top L_0^+ B$ be the orthogonal projection onto $\ker B$, so $y_{\mathrm{cyc}} = Py$ and $\operatorname{tr} P = \dim\ker B = \beta_1$. With $\varepsilon = 0$ and $y \neq 0$ almost surely, the cycle fraction $\rho = y^\top Py / y^\top y$ is invariant under scaling of y, so its distribution depends only on the direction $y/\|y\|_2$. Orthogonal invariance makes this direction uniform on the unit sphere of $\mathbb{R}^{|E_c|}$, and for a uniform unit vector z one has $\mathbb{E}[zz^\top] = I/|E_c|$. Therefore $\mathbb{E}[\rho] = \mathbb{E}[z^\top Pz] = \operatorname{tr}(P\,\mathbb{E}[zz^\top]) = \beta_1/|E_c|$. □

Proof of Proposition 4. Rank normalization places $\hat{g}_e$ and $\hat{c}_e$ in $(0, 1]$, so $u_e \in (-1, 1)$ for every candidate. If e is audited with $v_e = +1$ and $\lambda \geq 2$, then $\tilde{u}_e = u_e + \lambda > -1 + 2 = 1 > u_f$ for every unaudited f. If $v_e = -1$, then $\tilde{u}_e < 1 - 2 = -1 < u_f$. If $v_e = 0$, then $\tilde{u}_e = u_e$. This proves the first and third statements.

For the second, the deletion set is the q observed edges of lowest score and the admission set is the q proposed pairs of highest score. Changing the score of a single candidate changes the corresponding selected set by at most one insertion and one removal, so its symmetric difference with the original set has at most two elements. Each audited edge belongs to exactly one of the two rankings. We apply the $\mathcal{B}$ score changes one at a time. The symmetric difference is subadditive under composition, so the final edit set differs from the structural one in at most $2\mathcal{B}$ candidates. □

A.4 Interpretation

Whether an exact potential fit exists is a question of membership in $\mathrm{im}(B^\top)$. The support fixes that subspace, and every $W \in \mathcal{W}(E_c)$ keeps it. When no exact fit exists, the weights move the residual fraction only within the interval $[\rho_0/\gamma, \gamma\rho_0]$. As the spread shrinks to one, the fraction is pinned to ρ_0, and a larger spread moves it by at most a factor γ. A large ρ_0 therefore certifies that the inconsistency belongs to the candidate support and the frozen flow together. Removing it requires a change of support, which is what topology editing provides.

B Details of the Method

B.1 Flow and Diagnostic

Requirements. A quantity that decides between the two operation classes should have three properties. It should be *computable before repair*, from the observed graph and the frozen classifier alone, so that it can drive the decision. It should be *stable under the class it judges*. No member of $\mathcal{W}(E_c)$ should change whether it vanishes, and a member of spread γ should move it by at most a factor γ. It should also be *comparable across densities*, because candidate graphs of different size

and sparsity must be gated by one threshold. The construction below meets the first requirement by using only classifier outputs and the incidence matrix. It meets the second through Theorem 1, and the third through the calibration of Proposition 2 together with the per-dataset quantile grid of Section 4.3.

Choice of the comparison matrix. By Proposition 1, a comparison matrix either reduces to a single node score and carries no information about cycles, or it can detect one. By default we order the classes by their frequency in the training split and set $K_{ab} = \mathrm{sign}(b - a)$ under that order. This matrix is not of the excluded form for $C \geq 3$, and it compares any two classes by their position in the order. The linear comparison $K_{ab} = b - a$ is of the excluded form, with h the negated class index, and so is every difference $f(b) - f(a)$ of class scores. A comparison can circulate exactly when it is not such a difference. The sign of $b - a$ is the simplest order-based choice with this property. The cycle fraction depends on K, and every statement below holds for each fixed K. Table 13 varies K, including the native label order of each dataset.

Worked example. Return to the triangle of Figure 1a and place a unit comparison on each of its three edges, so $y = (1, 1, 1)$ around the cycle. Every node receives as much flow as it sends, so $By = 0$. The best potential is constant and the whole flow is residual. Here $\rho = 1$, $\beta_1 = 1$, and $\kappa_{\mathrm{diag}} = 3$, the largest value a triangle allows. Now give the first edge weight w and the others weight one. The weighted fit compromises differently, and a direct computation gives a weighted residual fraction of $9w/((1 + 2w)(w + 2))$. This equals one at $w = 1$, is smaller for every other w, and stays positive for every $w > 0$. No weight removes the contradiction. Deleting any one edge leaves a path. On a path $\beta_1 = 0$, every flow is a gradient, and $\rho = 0$. Theorem 1 shows that on every support no positive weighting fits a flow with a nonzero cycle part exactly.

Reading the chance level. The level $\beta_1/|E_c|$ is thus the cycle fraction of a flow whose direction is uniform on the sphere, and κ_{diag} compares the observed cycle fraction with it. Two facts shape how the gate uses this number. Since $\rho \leq 1$, κ_{diag} lies between zero and $|E_c|/\beta_1$, so the gate thresholds it by quantile within a dataset and not at a fixed value. The isotropic level is also an upper reference. A prediction flow vanishes on every edge whose endpoints share a prediction, so a classifier with no alignment to the graph produces a smaller cycle fraction. Appendix F reports that lower level as well, computed from the same flow with the predictions shuffled across nodes.

Reading the certificate. The corollary turns one number computed before training into a guarantee about every weighting the repair could learn. The guarantee concerns the support and the frozen flow. The value ρ_0 decides whether an exact fit exists. If $\rho_0 > \gamma\delta$, it also shows that no weighting of spread at most γ reaches tolerance δ. Learned weights cannot change either answer. Our hypothesis is that the instances whose flow no weighting can fit are the instances on which editing pays, and Section 5.3 tests it as Q2. Here the Hodge Router departs from prior structure learners. It chooses the operation before the repair is trained, from a quantity that no reweighting can alter. The residual value at which the gate switches is a tuning parameter of the router, selected on tuning instances disjoint from the evaluation seeds (Appendix D).

B.2 Routing

Link scores. The residual says that no weighting of the support fits the flow, but not where the support is at fault. For that we adapt GPS (Dong & Kluger, 2023) and train a self-supervised link predictor that scores every candidate pair,

$$g_{ij} = s_\theta(x_i, x_j; \mathcal{D}_{\mathrm{ssl}}), \qquad g_{ij} \in (0, 1), \tag{16}$$

where x_i is the feature vector of node i. The scorer s_θ is trained without labels, on a set $\mathcal{D}_{\mathrm{ssl}}$ of positive and negative node pairs drawn from the observed graph. For an edge $e = (i, j)$ we write g_e for g_{ij}. Observed edges and proposed additions are scored on the same footing, and the same predictor generates the proposal set E^+. We keep the pairs with the highest scores subject to a density budget $|E_c| \leq \nu|E_0|$. The candidate graph is then at most ν times as dense as the observed one. This caps β_1, and with it the chance level of Proposition 2.

Two views of one edge. The link score and the cycle contribution describe the same edge at two scales. The link score g_e is local. It looks at the two endpoints, asks whether nodes like these tend to be connected, and knows nothing about the rest of the graph. The cycle contribution $(y_{\mathrm{cyc}})_e^2$ is global. It is defined only through the decomposition on the whole support, and it measures how much of the inconsistency that the support cannot explain runs through this edge. The link score uses no labels, the cycle contribution sees them only through the frozen classifier, and both are computed once. The route stage uses both.

Residual gate. The most direct way to choose between the two operations would be to run both on every instance and keep the better one. That costs two full trainings per instance, and it judges reweighting by the very weights whose adequacy is in question. The residual avoids both problems. It is computed once from the frozen flow, and Theorem 1 says that no weighting can fit it away. On the instance being repaired, the gate fires before either policy is trained. If $\beta_1 = 0$, the candidate graph is a forest with an empty cycle space, and the instance is reweighted. If $C = 2$, the flow is a gradient by Proposition 1, and the instance is reweighted as well.

Cycle-aware ranking. Rank normalization in Eq. (8) puts the two signals on one scale without a tuned constant, and it makes the ranking of Eq. (9) invariant to monotone rescalings of either score. A large $\hat{c}_e$ marks an edge that carries much of the inconsistency. Deleting an edge with a large share of y_{cyc} removes the edge through which most of the inconsistency runs. Adding an edge can restore a path that message passing needs, and the cycle contribution of the proposed edge shows whether it would bring an inconsistency of its own. At the bottom of the ranking, an edge is implausible locally and inconsistent globally, and both signals say it should go. At the top, both say it should stay or be added. A ranking by $\hat{g}_e$ alone would be plain link-score rewiring, the GPS baseline of our tables. Compared with it, the cycle term moves an edge that looks plausible locally but carries much of the residual down the ranking, and an edge that looks odd locally but disturbs nothing up. Where the two terms cancel, the edge lands in the middle, and Section B.3 settles it.

B.3 Semantic Arbitration

The two structural signals of Section 4.3 settle the edges on which they agree. This section handles the edges on which they do not.

Contested edges. The link score and the cycle contribution are computed separately, and on some edges they point in opposite directions. An edge with high $\hat{g}_e$ and high $\hat{c}_e$ looks plausible from its endpoints yet carries a large share of the global inconsistency. An edge with low $\hat{g}_e$ and low $\hat{c}_e$ looks implausible yet disturbs nothing. On such edges u_e is close to zero, and the structural ranking has little basis for a decision. We measure the conflict by

$$d_e = 1 - |\hat{g}_e - \hat{c}_e|, \tag{17}$$

which approaches one when the two signals cancel, either because they pull in opposite directions or because neither is decisive. It approaches zero when one of them dominates. Where d_e is large, we consult a third source of evidence, the raw text of the two endpoints.

Query budget. One could send every candidate edge to a language model. That costs $|E_c|$ queries, grows with the graph, and spends most calls on edges that the structural score already settles. We instead fix a budget of $\mathcal{B}$ arbitrated edges per instance and spend it on the $\mathcal{B}$ candidates with the largest d_e. Verdicts are cached by unordered node pair, so a pair that recurs across instances of the same graph is answered from the cache. Appendix F reports fresh and cached queries separately. The disagreement score works as an acquisition rule. It sends the model to the edges where one answer can change a decision.

Prompt. Each query has three parts. The instruction asks whether a relation between two documents is plausible and fixes the answer format. The input is the text of the two endpoints. The output indicator restricts the answer to one of three tokens. The model sees no labels, no split identifiers, no corruption masks, and nothing about the rest of the graph, and its parameters are never updated. It answers with a verdict

$$v_e \in \{-1,\, 0,\, +1\}, \tag{18}$$

where $+1$ means that a relation between the two nodes is plausible, -1 means that it is not, and 0 abstains. The template is given in Appendix E.

Arbitration. The verdict enters the ranking only for the audited edges,

$$\tilde{u}_e = \begin{cases} u_e + \lambda v_e & e \text{ audited}, \\ u_e & \text{otherwise}, \end{cases} \qquad \lambda \geq 2. \tag{19}$$

A supported edge is retained or admitted, and a refuted edge is deleted or rejected. An abstention leaves the structural score as it was. The deletion and admission rankings of Section 4.3 are then computed on $\tilde{u}_e$, and the edit budget is spent as before. The margin $\lambda \geq 2$ makes every verdict decisive for the rank of its edge, and it bounds the reach of the arbiter before any query is sent.

Proposition 4 (Bounded influence)**.** *Let $\lambda \geq 2$. Every audited edge with a nonzero verdict is ranked past every unaudited edge in the direction of the verdict. The set of edits produced from $\tilde{u}_e$ differs from the set produced from u_e in at most $2\mathcal{B}$ edges, and an abstention changes nothing.*

A verdict thus settles the edge it concerns, and its reach ends at that edge and the one it displaces at the cut. The proof is in Appendix A.3. The arbiter is consulted only on edges whose structural score is near zero, so its verdicts fall on the decisions about which the structure had the least to say.

Role of the arbiter. Three boundaries fix what the most expensive source of evidence may do. The operation is chosen by the residual gate on structural grounds alone, and the arbiter acts inside the REWIRE branch, on edges that the structure has already marked as contested. Its queries are bounded by $\mathcal{B}$ and go where a single answer can change a decision, so the arbiter rules on edges and does not scan the graph. Its verdict is a statement about two pieces of text, so a cached verdict stays valid for any downstream classifier trained on the same graph.

Special cases. Three settings of the Hodge Router recover familiar methods. With $\mathcal{B} = 0$ the arbiter is off, and the procedure is a gated choice between learned edge weighting and link-score rewiring with a cycle-aware ranking. With τ above every observed κ_{diag} the gate always chooses REWEIGHT. The procedure then reduces to learning positive edge weights on the candidate support, which is the soft form of methods such as PTDNet and GNNGuard before any edge is pruned to zero. With $\tau = 0$ the gate rewires every instance with $C \geq 3$ and $\beta_1 > 0$, and the procedure reduces to link-score rewiring with a ranking corrected by the cycle contribution. Each of these is a fixed policy. The router lets the certificate decide, instance by instance, which of them to apply.

Complexity. The flow costs one forward pass of the frozen classifier and one bilinear product per candidate edge. The decomposition in Eq. (3) solves a single Laplacian system on the candidate graph. A conjugate gradient solver handles it in time linear in $|E_c|$ per iteration, and the diagnostics in Eq. (4) follow from the solution at no further cost. Link scoring is a single pass over the candidate pairs. The arbiter issues at most $\mathcal{B}$ queries per instance, and fewer when pairs recur, because cached verdicts are reused across seeds and instances. The cost of the language model is therefore fixed in advance and does not grow with the size of the graph.

C EXTENDED RELATED WORK

Graph structure learning. Graph structure learning estimates a corrected adjacency from the features and the observed structure. Pro-GNN learns the adjacency jointly with the node representation under sparsity, low-rank, and feature smoothness penalties (Jin et al., 2020). GCN-SVD and GCN-Jaccard clean the graph beforehand, by truncating its rank or by filtering edges on feature similarity (Entezari et al., 2020; Wu et al., 2019). PTDNet learns a probability of dropping each edge (Luo et al., 2021). CoGSL, SE-GSL, and JDR refine the graph through mutual information, structural entropy, or spectral resonance (Liu et al., 2022; Zou et al., 2023; Linkerhägner et al., 2025), and SLAPS adds self-supervision when labels are scarce (Fatemi et al., 2021). RGCN keeps the graph and attends less to neighbors of high variance (Zhu et al., 2019). GNNGuard weights each message by the similarity of its endpoints and prunes the edges between dissimilar nodes (Zhang & Zitnik, 2020). SAT trains the model against adversarial perturbations in the spectral domain (Li et al., 2023). Language models have been asked to propose, verify, or refine edges across an entire graph

Algorithm 1 Hodge Router

Require: observed graph $G_0 = (V, E_0)$, features X, node text T if available, frozen classifier with outputs p_i, link scorer s_θ, density budget ν, edit budget q, gate threshold τ, query budget $\mathcal{B}$, margin $\lambda \geq 2$

1: $E^+ \leftarrow$ highest scoring non-edges under s_θ with $|E_0 \cup E^+| \leq \nu|E_0|$, and $E_c \leftarrow E_0 \cup E^+$ *candidate support*
2: $y_e \leftarrow \frac{1}{2}\, p_i^\top K p_j$ for every $e = (i, j) \in E_c$ *Eq. (2)*
3: solve $BB^\top \phi^\star = By$, set $y_{\mathrm{cyc}} \leftarrow y - B^\top \phi^\star$, compute ρ and κ_{diag} *Eqs. (3), (4)*
4: **if** $\kappa_{\mathrm{diag}} < \tau$ or $\beta_1 = 0$ or $C = 2$ **then**
5: **return** REWEIGHT: learn $W \in \mathcal{W}(E_c)$ on the fixed support E_c
6: **end if**
7: $\hat{g}_e, \hat{c}_e \leftarrow$ rank-normalized g_e and $(y_{\mathrm{cyc}})_e^2$, and $u_e \leftarrow \hat{g}_e - \hat{c}_e$, $d_e \leftarrow 1 - |u_e|$ *Eqs. (8) to (17)*
8: **if** node text is available and $\mathcal{B} > 0$ **then**
9: query the arbiter on the $\mathcal{B}$ candidates of largest d_e, obtain v_e, set $\tilde{u}_e \leftarrow u_e + \lambda v_e$ on those edges *Eq. (19)*
10: **end if**
11: delete the q observed edges of lowest $\tilde{u}_e$ and admit the q proposed pairs of highest $\tilde{u}_e$
12: **return** REWIRE: the edited support with unit weights

with node text (Sun et al., 2025; Zhang et al., 2025b;a), to enrich the node representations (Chen et al., 2024a; He et al., 2024), or to label the nodes (Chen et al., 2024b). Most methods that change the graph learn a weighted adjacency (Zhou et al., 2026). In such an adjacency, rescaling, removing, and inserting an edge are moves of one optimizer. Signed edge weights (Bo et al., 2021) and sheaf Laplacians (Bodnar et al., 2022) treat a cross-class edge by changing its sign or its restriction map instead of deleting it. They form a third class, separate from the two that this paper compares. None of these methods first asks whether reweighting alone could fit the flow on the instance at hand. Our diagnostic answers that question before any repair is applied. It sits upstream of these methods, and any of them can serve as the repair it selects.

Hodge decomposition of edge flows. Hodge theory splits an edge signal into a gradient induced by node potentials and a residual in the cycle space (Lim, 2020; Schaub et al., 2020). HodgeRank uses this split to rank items from pairwise comparisons (Jiang et al., 2011). It fits potentials by least squares, with weights fixed by the number of comparisons per pair. It then divides the residual into a local part on triangles and a global part, to judge whether a single ranking is meaningful. Osting et al. (2013) choose which comparisons to collect so that this fit improves. The same split of a flow into a potential part and a harmonic part underlies the decomposition of games (Candogan et al., 2011). Cycle inconsistency has been used to find wrong edges in view graphs (Zach et al., 2010), and it has a discrete counterpart in the balance theory of signed graphs and in correlation clustering (Harary, 1953; Bansal et al., 2004). Later work studied intransitivity in competitive matchups and learned directly from edge flows (Chen & Joachims, 2016; Jia et al., 2019). The cycle space has dimension $|E_c| - |V| + \beta_0$ and grows with the density of the support, so we measure the residual against reference levels and keep the candidate graph sparse. The works above read the residual as inconsistency among given comparisons. We apply the decomposition to a flow built from classifier predictions. We treat the whole cycle space as one residual without attaching triangles, and we use it to decide whether the support must change for the flow to be fit.

Diagnostics for rewiring. The diagnostics that currently guide rewiring measure bottlenecks. Analyses of over-squashing identify long paths that message passing cannot carry (Alon & Yahav, 2021). Effective resistance quantifies how hard it is for two nodes to communicate (Black et al., 2023), and curvature based rewiring adds edges where the curvature is negative (Topping et al., 2022). Each of these points to one remedy, which is to add edges, and none can say when the weights on the existing support would have been enough. ESNR estimates how noisy a graph is relative to its features or labels, and the GPS rewiring of the same paper always edits the graph (Dong & Kluger, 2023). AdaEdge edits edges by the predicted labels of their endpoints (Chen et al., 2020), so classifier predictions have guided editing before. It judges each edge on its own and always edits. We decompose the prediction flow on the whole support and first test whether any weighting of that support can fit it. Several studies report that rewiring gains depend on the protocol. ComFy shows that rewiring that lowers the spectral gap can help as much as rewiring that raises it (Rubio-Madrigal

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	Classes	Features	Node text	Role
Cora	2,485	5,069	7	1,433	yes	main, arbiter
CiteSeer	2,110	3,668	6	3,703	yes	main, arbiter
PubMed	19,717	44,338	3	500	yes	main, arbiter
PolBlogs	1,222	16,714	2	none	no	random corruption, structure only
DBLP	17,716	105,734	4	1,639	no	main, structure only
ogbn-arxiv	169,343	1,166,243	40	128	yes	cost, density sweep

Table 4: Datasets. Counts for the first four follow the Pro-GNN protocol and refer to the largest connected component. PolBlogs has no node features, and following Pro-GNN its feature matrix is the identity. DBLP is the graph2gauss release, and ogbn-arxiv is the Open Graph Benchmark release. Node text is the title and abstract of each paper, released with TAPE for Cora and PubMed (He et al., 2024), by Chen et al. (2024a) for CiteSeer, and with the benchmark for ogbn-arxiv. The arbiter is used on Cora, CiteSeer, and PubMed.

et al., 2025). Reported gains can be artifacts of hyperparameter sweeps (Tori et al., 2025), and gains credited to structure learning can come from the inner optimization (Kim & Jang, 2026). Our diagnostic is computed before any training on the repaired graph. The two policies it chooses between share the candidate set and the backbone, and every editing method receives the same budget, so a difference between them can be credited to the operation itself.

D Experimental Details

D.1 Datasets

The perturbed graphs of the Metattack protocol are the ones released with Pro-GNN through the DeepRobust library (Li et al., 2020). Each dataset is reduced to its largest connected component, and the 10/10/80 split of Pro-GNN is kept. PolBlogs has no node features, and following Pro-GNN its feature matrix is the identity. The text of Cora and PubMed is the title and abstract of every paper as released with TAPE (He et al., 2024), and the text of CiteSeer is the release of Chen et al. (2024a). Each release is mapped to the Pro-GNN node order by the paper identifiers, and the mapping is checked against the edges of the Pro-GNN graph. DBLP is the citation graph of graph2gauss (Bojchevski & Günnemann, 2018), which carries bag of words features and no text, and receives a stratified 10/10/80 split with the same seeds. ogbn-arxiv uses the official split of the Open Graph Benchmark (Hu et al., 2020) and its released titles and abstracts. On PolBlogs the features are the identity, so the feature neighbor pool of the link scorer is omitted and the cross-class generator uses the label condition alone.

D.2 Perturbation Generators

Let $m = |E_0|$ and let η be the corruption rate. Uniform corruption removes $\lfloor \eta m/2 \rfloor$ edges chosen uniformly and inserts the same number of node pairs chosen uniformly among the non-edges. Betweenness deletion removes the $\lfloor \eta m \rfloor$ edges of highest edge betweenness in the observed graph, which targets the paths that message passing depends on. Cross-class insertion draws $\lfloor \eta m \rfloor$ node pairs whose labels differ and whose feature cosine similarity lies below the median similarity of the observed edges. The labels are used by the generator only, never by any repair method. The mixed mechanism spends one third of the budget on each of the three. Metattack uses the graphs released with Pro-GNN, produced with the Meta-Self variant of Zügner & Günnemann (2019) and with A-Meta-Self on PubMed. DBLP has no released graphs, so Meta-Self is run with the DeepRobust implementation at the same rates, once per rate, and the ten seeds are training seeds on that graph. Under random corruption DBLP uses the mixed mechanism at $\eta = 0.15$. PRBCD (Geisler et al., 2021) runs against a surrogate GCN trained on the clean graph with a budget of $\lfloor \eta m \rfloor$ edge flips, and the resulting graph is given to every method. All random generators are seeded, and the ten evaluation seeds are disjoint from the tuning seeds of Section 5.1. Under Metattack the ten seeds are training seeds on the single released graph.

D.3 Frozen Classifier and Flow

The classifier is a two layer GCN with 256 hidden units and dropout 0.5. It is trained with Adam (Kingma & Ba, 2015) at learning rate 0.01 and weight decay 5×10^{-4} on the training split of the perturbed graph, for at most 200 epochs. Training stops early when the validation loss has not improved for 20 epochs. Its softmax outputs are the distributions p_i of Eq. (2). Every candidate edge receives the orientation from its smaller to its larger node index. The default comparison matrix orders the classes by their frequency in the training split and sets $K_{ab} = \operatorname{sign}(b - a)$ under that order. Three variants appear in Table 13. The random antisymmetric matrix draws its upper triangle uniformly from $[-1, 1]$ with a fixed seed. The prototype cosine matrix multiplies the default entry by one minus the cosine similarity of the two class centroids in feature space, so that pairs of dissimilar classes weigh more. The unit antisymmetric matrix is the default sign pattern under the native label order of the dataset instead of the frequency order. The Laplacian system of Eq. (3) is solved by conjugate gradient to a relative residual of 10^{-10} within at most 1,000 iterations, and the diagnostics of Eq. (4) use $\varepsilon = 10^{-8}$.

D.4 Link Scorer and Candidate Set

The link scorer adapts GPS (Dong & Kluger, 2023). A two layer multilayer perceptron with 256 hidden units maps each feature vector to an embedding. The score of a pair is the sigmoid of the inner product of the two embeddings. It is trained for 200 epochs with binary cross entropy on the observed edges against five uniformly sampled non-edges per edge, without labels. Scores are computed for every observed edge, for every pair within two hops in the observed graph, and for the twenty nearest feature neighbors of every node under cosine similarity. The highest scoring non-edges among these are admitted to E^+ until $|E_c| = \nu|E_0|$, or until the pool is exhausted. Ranks in Eq. (8) are average ranks, so tied scores receive the same value. The same scorer provides g_e for the ranking of Eq. (8). The ESNR gate thresholds the ESNR of the same paper in its vanilla form, computed from the labels of all nodes on the observed graph, so like the rate gate it is a reference. The rate gate thresholds the edit budget factor b, and the disagreement gate thresholds the share of candidate edges whose endpoint argmax predictions differ. Assortativity is the label assortativity coefficient of Newman (2003) with the predictions of the frozen classifier as labels. Feature smoothness is the measure of Hou et al. (2020). Feature homophily is the mean cosine similarity of the endpoint features over the observed edges. The spectral gap is the second smallest eigenvalue of the normalized Laplacian of the observed graph. Each of these gates, and each competing diagnostic of Table 1, selects its threshold exactly as τ is.

D.5 Reweighting Learner

Reweight keeps the candidate support E_c and learns one positive weight per candidate edge, $w_e = \exp(\theta_e)$. The weights enter the symmetric normalization of the adjacency over which the GCN propagates. The parameters θ_e start at zero. They are trained jointly with the GCN on the training cross entropy, with the same optimizer, learning rate, weight decay, and early stopping rule, and they are clipped to $[-\ln 4, \ln 4]$ after every step. Table 17 varies the clip. The spread $\gamma = w_{\max}/w_{\min}$ is therefore at most 16, and the learned spread of every instance is recorded and reported in Table 5 for the instances of the theorem check. Proposed edges in E^+ start with the same weight as observed edges, so the learner alone decides how much each candidate counts.

D.6 Gate Threshold

For every dataset the tuning instances are perturbed graphs generated with five seeds disjoint from the ten evaluation seeds, under every configuration of the random corruption protocol. The threshold selected there is used unchanged under Metattack and PRBCD, whose evaluation graphs never enter the selection. On each tuning instance both fixed policies are trained and their validation Macro-F1 is recorded, together with κ_{diag}. The grid consists of the 21 quantiles of κ_{diag} over the tuning instances, at levels $0, 0.05, \ldots, 1$. For each grid point the gated policy picks Rewire on tuning instances with $\kappa_{\mathrm{diag}} \geq \tau$ and Reweight otherwise, and the grid point with the highest mean validation Macro-F1 is kept. Table 8 reports the sweep at every other grid point. The random gate permutes the residual gate's decisions across the evaluation instances of a dataset, using the evaluation seed. The oracle

picks the fixed policy with the higher test Macro-F1 on each evaluation instance and is never used for selection.

D.7 BASELINES

Every baseline runs from its official implementation, or from DeepRobust when the authors provide none. Each uses the hyperparameters of its paper and, where the method admits a backbone, a two layer GCN with 256 hidden units. Every method that edits the graph is evaluated with the same edit budget q as our policies, using its own scores. The GPS row ranks the candidates by our link scorer alone. GCN-Jaccard and GCN-SVD keep the thresholds and ranks of their papers. Pro-GNN, PTDNet, CoGSL, SLAPS, and JDR use their learned adjacency within this common budget. LLM4RGNN uses the same $\mathcal{B}$-edge arbiter budget and its own suspicion score. The GCN row of Table 3 is the two layer GCN trained on the perturbed graph without repair, which is also the frozen classifier of the Hodge Router. Accuracies in Tables 3 and 10 are reported to one decimal.

D.8 ARBITER

The arbiter is queried on the $\mathcal{B}$ candidates of largest disagreement d_e, with $\mathcal{B} \in \{0, 20, 50, 100, 200\}$, $\mathcal{B} = 50$ by default, and $\lambda = 2$. Every method that uses a language model is held to $\mathcal{B}$ arbitrated edges per instance. Two criteria for the router study are fixed in advance. Arbitration helps if accuracy at every $\mathcal{B} > 0$ exceeds its value at $\mathcal{B} = 0$ under disagreement selection, and exceeds random selection at the same $\mathcal{B}$. Each component is judged by the sign of its row in Table 13 relative to the full method. The default arbiter is GLM-5.2 through its public interface at temperature zero with at most five output tokens. The open model is Qwen2.5-7B-Instruct run locally with greedy decoding. Each endpoint text is the title followed by the abstract, truncated to 256 tokens. Verdicts are cached under a hash of the unordered node pair together with the prompt template, so repeated seeds and repeated budgets reuse the same verdict. The budget $\mathcal{B}$ counts arbitrated edges per instance, and Table 18 reports the fresh queries that remain after caching. A response that is not one of the three allowed words is treated as an abstention, and no query is retried. The controlled study of Table 14 keeps the selected edges fixed and changes what the arbiter sees or says. Text degradation replaces each document by its title only or by the first half of its abstract. The shuffled control pairs each contested edge with the texts of two random nodes. The verdict replacements substitute a random verdict, a constant YES, or a constant NO for the model's answer. The adaptive attacker knows the scorer and the cached verdicts and regenerates its perturbation to favor edges of high d_e whose endpoint texts the model judges plausible.

D.9 METRICS AND STATISTICS

Accuracy and Macro-F1 are computed on the test split. The edge level metrics compare the edits with the perturbation mask. Precision is the share of edits that undo a perturbation, and recall is the share of perturbations that the edits undo. The share of insertions removed and the share of deletions recovered split the recall by the type of perturbation. Spearman correlations are computed over instances, and their intervals come from 1,000 bootstrap resamples of whole cells, where a cell is one dataset, mechanism, and rate with its ten seeds. Partial correlations rank transform the continuous variables, code the dataset by indicators, regress the diagnostic and the gap on the covariates, and correlate the residuals. Competing diagnostics are oriented so that larger values predict REWIRE before correlations are compared, and each receives a threshold by the quantile grid used for τ, which defines its regret. AUROC and AUPRC treat the diagnostic as a score for the event $g^* > 0$. The expected calibration error uses ten equal width bins on the probability of that event obtained by fitting a logistic function of the diagnostic on the tuning instances and evaluated on the evaluation instances. The precision of the audited edges is the precision of the edits made on them against the perturbation mask. Agreement with the clean graph is the share of nonzero verdicts, edited or not, that the clean graph confirms. Regret is the Macro-F1 of the oracle minus that of the threshold rule, averaged over the datasets of Table 1. Pairwise tests are paired over identical perturbed graphs, and the Holm procedure (Holm, 1979) corrects every family of comparisons within one table.

D.10 Ablation Variants

Table 13 removes or replaces one component at a time. Without GPS, the proposals still come from the link scorer but the ranking uses the cycle contribution alone. Without Hodge, the gate is kept and the ranking uses the link score alone. Without the arbiter, $\mathcal{B} = 0$. The two fixed gates are the special cases of Appendix B.3. Random verdicts repeat the verdict replacement of Table 14 in this layout. The three variants of K are defined above, and min–max normalization replaces the ranks of Eq. (8) by min–max scaling within the instance.

D.11 Hardware and Software

All runs use one GPU. The Hodge decomposition, the diagnostics, and the statistics use SciPy, and the GNNs use PyTorch and PyTorch Geometric. Software versions, seeds, the prompt template hash, and the query and cache counts of every run are recorded with the outputs in the code release.

E Prompt

The arbiter receives the following prompt for each contested edge. The two placeholders are replaced by the title and abstract of the endpoint documents, truncated to a fixed number of tokens. Nothing else is inserted.

```
You are given two documents from the same collection.
Decide whether a direct relation between them (for example,
one citing the other or both addressing the same specific
problem) is plausible based on their content alone.
Document A: {text of node i}
Document B: {text of node j}
Answer with exactly one word:  YES if such a relation is
plausible, NO if it is not, UNSURE if the texts do not allow
a judgment.
```

The answers YES, NO, and UNSURE are mapped to $v_e = +1$, -1, and 0 in Eq. (18). Any other response is treated as UNSURE.

F Additional Results

Numerical check of the theorem. The upper block of Table 5 checks Theorem 1 on one evaluation instance per dataset, the Metattack graph at 15% with the first training seed. We draw 1,000 random positive diagonal weightings with spread $\gamma = 16$. For each one the weighted residual fraction is computed by solving the weighted least squares problem of Eq. (5). The smallest and largest values are compared with the bounds ρ_0/γ and $\gamma\rho_0$, and the learned γ column gives the spread that the Reweight learner actually reached on the same instance.

Density. Table 6 reports the diagnostics as the density budget ν grows, together with three references. The Gaussian flow, with i.i.d. standard Gaussian entries, gives the isotropic chance level of Proposition 2. The shuffled flow keeps the classifier's predictions and permutes them across nodes, averaged over twenty permutations. The clean column gives κ_{diag} on the unperturbed graph at the same density. The r_s column gives the Spearman correlation with g^* at each density. ogbn-arxiv is perturbed by uniform corruption at $\eta = 0.15$ with ten seeds, and both fixed policies are trained on these ten instances to define g^* there. The forests column counts instances with $\beta_1 = 0$, which the gate routes to Reweight without consulting the flow.

Diagnostic validity by stratum. Table 7 reports the correlation of Table 1 within each dataset and random corruption mechanism, adds one Metattack stratum for each of Cora, CiteSeer, and PubMed, and gives the partial correlation, the AUROC, the mean gap, and the Holm corrected p value of each stratum.

Theorem check: one instance, 1,000 weightings of spread 16							
Dataset	ρ_0	ρ_0/γ	$\min_W$	$\max_W$	$\gamma\rho_0$	Viol.	Learned γ
Cora	0.1004	0.0063	0.0848	0.1011	1.6064	0	1.24
CiteSeer	0.0512	0.0032	0.0441	0.0519	0.8192	0	1.10
PubMed	0.0209	0.0013	0.0199	0.0211	0.3344	0	7.21
DBLP	0.0241	0.0015	0.0209	0.0247	0.3856	0	2.23

Flow structure: all instances, $\nu = 2$					
Dataset	Gaussian	ρ	Shuffled	κ_{diag}	κ_{diag} (clean)
Cora	0.754	0.099	0.123	0.131	0.083
CiteSeer	0.708	0.052	0.080	0.073	0.048
PubMed	0.783	0.021	0.052	0.026	0.015
DBLP	0.847	0.025	0.062	0.029	0.018

Table 5: The prediction flow on real instances. The upper block compares the residual fraction $R_W/\|y\|_W^2$ under 1,000 random positive diagonal weightings of spread $\gamma = 16$, with the two bounds of Theorem 1. Viol. counts draws outside the bounds beyond solver tolerance, and learned γ is the spread that the REWEIGHT learner reaches on the same instance. The lower block reports cycle fractions averaged over all instances. Gaussian is the cycle fraction of a flow with i.i.d. standard Gaussian entries on the same support, whose expectation is $\beta_1/|E_c|$ by Proposition 2. Shuffled permutes the predictions across nodes. Clean is the diagnostic on the unperturbed graph.

Dataset	ν	$\beta_1/\|E_c\|$	ρ	ρ (Gaussian)	ρ (shuffled)	κ_{diag}	κ_{diag} (clean)	r_s	Forests
Cora	2	0.756	0.099	0.754	0.123	0.131	0.083	0.36	0
	4	0.878	0.109	0.877	0.143	0.124	0.079	0.33	0
	8	0.939	0.114	0.938	0.153	0.122	0.072	0.29	0
CiteSeer	2	0.714	0.052	0.708	0.080	0.073	0.048	0.45	0
	4	0.856	0.066	0.854	0.096	0.077	0.050	0.41	0
	8	0.928	0.066	0.928	0.104	0.071	0.044	0.37	0
PubMed	2	0.783	0.021	0.783	0.052	0.026	0.015	0.33	0
	4	0.891	0.015	0.891	0.059	0.017	0.010	0.30	0
	8	0.945	0.019	0.945	0.063	0.020	0.012	0.27	0
DBLP	2	0.848	0.025	0.847	0.062	0.029	0.018	0.41	0
	4	0.923	0.027	0.921	0.069	0.029	0.018	0.38	0
	8	0.961	0.026	0.961	0.072	0.027	0.016	0.35	0
ogbn-arxiv	2	0.927	0.071	0.925	0.104	0.077	0.048	0.31	0
	4	0.964	0.076	0.963	0.113	0.079	0.049	0.28	0
	8	0.982	0.074	0.982	0.118	0.075	0.047	0.26	0

Table 6: Diagnostics as the density budget $|E_c| \le \nu|E_0|$ grows. ρ (Gaussian) is the cycle fraction of a flow with i.i.d. standard Gaussian entries on the same support, whose expectation is $\beta_1/|E_c|$ by Proposition 2. ρ (shuffled) is the cycle fraction of the prediction flow after the predictions are permuted across nodes, averaged over twenty permutations. κ_{diag} (clean) is the diagnostic on the unperturbed graph at the same density. r_s is the Spearman correlation of κ_{diag} with g^* at that density. On ogbn-arxiv it is computed over ten uniform corruption instances at $\eta = 0.15$. Forests counts instances with $\beta_1 = 0$. PolBlogs is excluded, since its cycle fraction is identically zero.

Dataset	Perturbation	N	r_s	partial r_s	AUROC	mean g^*	Holm p
Cora	Metattack	30	0.55	0.47	0.79	0.11	0.009
	Uniform	40	0.36	0.31	0.68	0.00	0.068
	Betweenness deletion	40	0.29	0.25	0.63	-0.01	0.104
	Cross-class insertion	40	0.47	0.41	0.73	0.01	0.011
CiteSeer	Metattack	30	0.60	0.51	0.82	0.05	0.003
	Uniform	40	0.42	0.38	0.72	0.00	0.024
	Betweenness deletion	40	0.26	0.22	0.61	-0.01	0.105
	Cross-class insertion	40	0.32	0.28	0.66	0.00	0.088
PubMed	Metattack	30	0.52	0.44	0.77	0.06	0.015
	Uniform	40	0.45	0.33	0.71	0.01	0.015
	Betweenness deletion	40	0.50	0.36	0.76	0.00	0.006
	Cross-class insertion	40	0.34	0.29	0.66	0.01	0.080
DBLP	Mixed	10	0.45	0.38	0.69	-0.01	0.105

Table 7: Spearman correlation of κ_{diag} with the operation gap g^* within each dataset and perturbation family. The partial correlation controls for corruption rate, candidate density, and β_1. Holm p is the corrected p value of the one-sided test $r_s > 0$, over the family of thirteen strata fixed in advance. PolBlogs is excluded, since its cycle fraction is identically zero. DBLP has one stratum, the mixed mechanism at $\eta = 0.15$. Mean g^* is on the $[0, 1]$ scale of Macro-F1.

Gate threshold. Table 8 sweeps τ along the quantile grid of Appendix D and reports validation Macro-F1, test Macro-F1, and the fraction of instances routed to REWIRE.

Random corruption. Table 9 gives the four policies under every mechanism and rate of the random corruption protocol, per dataset. Table 2 in the main text is the pooled view of these cells.

PRBCD. Table 10 repeats the layout of Table 3 under PRBCD poisoning transferred from a surrogate GCN.

Edge recovery. Table 11 measures the edits of every method that changes the support against the perturbation mask.

Router study. Table 12 varies the query budget and the rule that selects the arbitrated edges, and Table 13 removes one component of the Hodge Router at a time. Both report accuracy under Metattack at 15%, where the gate rewires and the arbiter acts. Without the arbiter the router loses 1.36 points, and the residual gate stays within 0.3 points of Always-Rewire, which Metattack favors. Ranking by the link score alone costs 5.57 points, and ranking by the cycle contribution alone costs 1.97. Random verdicts undo the gain of the arbiter, and the three alternative comparison matrices stay within one point of the default.

Controlled study of the arbiter. Table 14 varies the model, degrades the text, replaces the verdicts, and adds an adaptive attacker, all at $\mathcal{B} = 50$ with the same selected edges under Metattack at 15%. The last column is the agreement of the model with a consensus judgment of 50 audited edges per dataset by two annotators, blind to the model's verdict and to the perturbation mask.

Backbone transfer. Table 15 trains GCN, GAT, APPNP (Gasteiger et al., 2019), and GraphSAGE (Hamilton et al., 2017) on the observed graph and on the graph produced by each policy under Metattack at 15%. The gate and the edits are computed once from the frozen GCN and are not recomputed per backbone. The weights learned by REWEIGHT are frozen and passed to the backbones that accept edge weights. GAT and GraphSAGE ignore edge weights, so their Always-Reweight cells report the unweighted candidate support.

Label-free variant. Table 16 replaces the frozen classifier by soft assignments of every node to the C centroids of a k-means clustering of the features. The matrix K orders the clusters by size. No training label enters the flow, the gate, or the scores.

Dataset	Quantity	Threshold quantile 0	0.1	0.2	0.3	0.4	0.5
Cora	Validation Macro-F1	81.02	81.07	81.21	81.38	81.67	81.76
	Test Macro-F1	80.31	80.38	80.46	80.61	80.73	80.88
	REWIRE fraction	0.967	0.942	0.808	0.700	0.625	0.483
CiteSeer	Validation Macro-F1	63.03	63.10	63.22	63.42	63.51	63.72
	Test Macro-F1	61.12	61.10	61.25	61.35	61.52	61.68
	REWIRE fraction	0.942	0.875	0.742	0.650	0.608	0.517
PubMed	Validation Macro-F1	84.57	84.59	84.69	84.75	84.83	84.71
	Test Macro-F1	84.10	84.16	84.17	84.18	84.21	84.16
	REWIRE fraction	0.972	0.955	0.861	0.744	0.595	0.497
DBLP	Validation Macro-F1	72.30	72.57	72.84	72.87	73.08	73.38
	Test Macro-F1	71.96	72.18	72.26	72.51	72.68	72.94
	REWIRE fraction	0.900	0.900	0.800	0.700	0.600	0.500

Dataset	Quantity	Threshold quantile 0.6	0.7	0.8	0.9	1
Cora	Validation Macro-F1	81.71	81.62	81.49	81.48	81.53
	Test Macro-F1	80.84	80.78	80.77	80.73	80.74
	REWIRE fraction	0.367	0.233	0.042	0.008	0.000
CiteSeer	Validation Macro-F1	63.84	63.79	63.75	63.62	63.58
	Test Macro-F1	61.78	61.74	61.73	61.67	61.63
	REWIRE fraction	0.408	0.308	0.192	0.158	0.008
PubMed	Validation Macro-F1	84.49	84.33	84.11	84.11	84.00
	Test Macro-F1	83.94	83.84	83.64	83.60	83.55
	REWIRE fraction	0.383	0.262	0.148	0.061	0.000
DBLP	Validation Macro-F1	73.71	73.99	73.94	73.94	73.96
	Test Macro-F1	73.22	73.41	73.36	73.38	73.37
	REWIRE fraction	0.400	0.300	0.200	0.100	0.000

Table 8: Validation Macro-F1, test Macro-F1, and the fraction of instances routed to REWIRE as the gate threshold τ moves along the quantile grid. At quantile 0 the gate rewires almost every instance, and at quantile 1 almost none. The selected τ is the grid point with the highest validation Macro-F1. PolBlogs is omitted, since its diagnostic is identically zero and the gate reweights every instance at every τ.

Sensitivity. Table 17 varies the margin λ, the density budget ν, the edit budget factor b, and the weight clip, one at a time, under Metattack at 15%.

Cost. Table 18 reports the time of each stage, the arbiter's fresh query, token, and price counts, and the time of JDR, on every dataset including ogbn-arxiv.

Policy	Perturbation	$\eta = .05$	.10	.15	.20
Cora					
Always-Reweight	Uniform	81.69±0.26	80.56±0.49	80.11±0.57	79.77±0.71
	Betweenness	82.79±0.20	81.83±0.26	82.24±0.21	81.76±0.35
	Cross-class	80.84±0.49	79.70±0.51	78.78±0.75	78.14±0.59
Always-Rewire	Uniform	82.12±0.68	80.92±0.43	80.07±0.82	78.77±1.02
	Betweenness	82.79±0.37	81.38±0.44	79.74±0.69	79.23±0.54
	Cross-class	81.65±0.44	80.12±0.61	78.96±0.69	77.13±0.44
Residual gate	Uniform	82.14±0.50	80.96±0.45	80.23±0.70	79.67±0.90
	Betweenness	82.98±0.33	81.96±0.32	82.21±0.40	81.49±0.44
	Cross-class	81.62±0.54	80.22±0.59	79.10±0.72	77.96±0.35
Hodge Router	Uniform	82.37±0.56	81.21±0.44	80.61±0.74	80.11±0.91
	Betweenness	83.13±0.32	82.15±0.36	82.49±0.41	81.70±0.44
	Cross-class	82.10±0.61	80.71±0.56	79.71±0.73	78.66±0.38
CiteSeer					
Always-Reweight	Uniform	62.25±0.64	61.41±0.82	61.30±0.65	61.12±1.36
	Betweenness	62.95±0.59	63.20±0.55	62.69±0.57	62.33±0.74
	Cross-class	61.15±0.60	60.48±0.78	60.26±1.07	59.77±0.86
Always-Rewire	Uniform	62.77±0.66	61.77±0.65	61.13±0.61	60.57±0.99
	Betweenness	62.55±0.26	62.37±0.55	60.98±0.56	60.59±0.47
	Cross-class	61.94±0.48	60.45±0.78	59.18±1.02	58.24±0.96
Residual gate	Uniform	62.85±0.66	61.86±0.70	61.52±0.46	61.37±1.20
	Betweenness	63.11±0.49	63.32±0.54	62.69±0.58	62.26±0.61
	Cross-class	61.99±0.59	60.72±0.72	60.21±1.06	59.48±0.97
Hodge Router	Uniform	63.03±0.65	62.08±0.75	61.92±0.45	61.86±1.14
	Betweenness	63.21±0.44	63.44±0.54	62.96±0.64	62.62±0.61
	Cross-class	62.35±0.66	61.11±0.81	60.69±1.06	60.28±0.96

Policy	Perturbation	$\eta = .05$	.10	.15	.20
PubMed					
Always-Reweight	Uniform	85.32±0.26	84.61±0.34	84.09±0.23	83.34±0.28
	Betweenness	86.01±0.12	85.80±0.11	85.75±0.14	85.70±0.14
	Cross-class	83.63±0.27	81.49±0.16	79.56±0.30	78.10±0.41
Always-Rewire	Uniform	85.77±0.15	85.30±0.11	84.76±0.17	84.16±0.34
	Betweenness	86.17±0.09	85.73±0.17	85.54±0.16	85.30±0.18
	Cross-class	84.35±0.28	82.73±0.33	80.93±0.41	79.30±0.45
Residual gate	Uniform	85.74±0.22	85.19±0.24	84.77±0.24	84.20±0.34
	Betweenness	86.19±0.10	85.89±0.16	85.69±0.24	85.60±0.16
	Cross-class	84.23±0.22	82.68±0.28	80.87±0.41	79.47±0.49
Hodge Router	Uniform	85.92±0.19	85.48±0.25	84.98±0.25	84.52±0.43
	Betweenness	86.33±0.12	86.06±0.13	85.82±0.22	85.75±0.16
	Cross-class	84.42±0.22	83.38±0.31	81.51±0.40	80.03±0.48
PolBlogs					
Always-Reweight	Uniform	93.34±0.59	92.14±0.66	90.12±0.79	88.87±1.04
	Betweenness	87.98±0.24	87.26±0.52	85.56±0.46	84.12±0.53
	Cross-class	87.29±0.61	82.09±0.98	78.51±1.08	76.62±0.81
Always-Rewire	Uniform	94.02±0.59	92.45±0.50	90.03±0.82	88.87±1.12
	Betweenness	88.09±0.22	86.80±0.22	85.17±0.40	83.30±0.46
	Cross-class	87.73±0.98	81.91±0.62	78.48±0.75	76.08±1.10

Table 9: Macro-F1 (%, mean±std over ten seeds) of the policies under the random corruption protocol, by dataset, mechanism, and corruption rate η. The edit budget factor b equals the corruption rate, with $q = \lfloor b|E_0| \rfloor$ deletions and q additions. PolBlogs has two classes, so its flow is a gradient and the residual gate and the Hodge Router reduce to Always-Reweight there. DBLP uses one mixed mechanism at $\eta = 0.15$ and is reported in Table 2.

Method	Cora			CiteSeer		
	5%	15%	25%	5%	15%	25%
GCN	76.4±0.8	53.2	42.6±2.4	68.7±0.9	57.2	47.9±2.6
GCN-Jaccard	77.0±0.7	58.9±1.9	47.3±2.5	69.3±0.8	60.1±1.7	50.6±2.4
GCN-SVD	76.6±0.7	57.4±2.1	46.1±2.7	68.9±0.8	58.8±2.0	49.4±2.6
Pro-GNN	77.9±0.6	62.8±2.0	51.7±2.4	70.2±0.5	62.9±1.5	54.1±2.3
JDR	77.6±0.8	63.4±1.7	52.9±2.2	70.0±0.6	63.5±1.4	54.8±2.1
GPS	77.2±0.6	61.1±1.6	50.2±2.5	69.6±0.6	61.7±1.6	52.6±2.4
Always-Reweight	**78.3**±0.5	**66.9**±1.1	**55.8**±2.3	**70.9**±0.3	**67.3**±1.2	**58.9**±2.2
Always-Rewire	77.4±0.6	62.7±2.1	52.4±2.0	69.4±0.8	61.0±1.9	52.1±1.9
Hodge Router	78.2±0.5	66.8±1.1	55.7±2.3	70.8±0.3	67.2±1.2	58.8±2.2

Method	PubMed			DBLP			Avg.
	5%	15%	25%	5%	15%	25%	
GCN	80.8±0.5	74.6	66.4±1.9	71.2±1.0	60.4±1.6	50.9±2.1	62.5
GCN-Jaccard	81.5±0.4	76.3±1.4	68.7±2.2	70.6±1.2	60.1±1.7	51.5±2.0	64.3
GCN-SVD	80.9±0.5	75.5±1.6	67.6±2.4	**72.9**±1.1	61.0±1.6	51.3±1.9	63.9
Pro-GNN	82.0±0.4	78.1±1.3	70.9±2.1	72.4±1.3	63.1±1.7	54.6±2.0	66.7
JDR	81.8±0.4	78.6±1.2	71.6±2.0	72.2±1.2	63.7±1.6	55.2±1.9	67.1
GPS	81.6±0.4	77.7±1.3	70.2±2.1	72.0±1.0	62.6±1.5	53.9±1.9	65.9
Always-Reweight	82.2±0.3	78.0±1.2	70.4±2.0	72.8±0.9	64.1±1.4	55.7±1.8	68.4
Always-Rewire	82.4±0.5	81.9±0.9	76.8±2.1	72.1±1.1	**65.6**±1.4	**58.0**±1.9	67.6
Hodge Router	**82.7**±0.4	**82.4**±0.9	**77.5**±1.9	72.7±0.9	65.4±1.4	57.8±1.9	**69.7**

Table 10: Accuracy (%, mean±std over ten runs where available; the three GCN entries at 15% on Cora, CiteSeer, and PubMed report means only) under PRBCD poisoning transferred from a surrogate GCN at 5%, 15%, and 25% of the edges, with the Pro-GNN split. Every repair method uses a two layer GCN backbone. Avg. is the mean over the twelve cells. Best in bold and second best underlined, per column. On DBLP the Hodge Router coincides with the residual gate.

Method	Precision	Recall	Ins. removed	Del. recovered	Edits
Cora					
GCN-Jaccard	0.34	0.34	0.38	0.00	812
GCN-SVD	0.07	0.14	0.15	0.06	2,204
Pro-GNN	0.28	0.66	0.72	0.07	1,976
PTDNet	0.14	0.11	0.12	0.00	608
GPS	0.14	0.28	0.30	0.09	1,520
ComFy	0.15	0.31	0.33	0.08	1,520
LLM4RGNN	0.18	0.37	0.40	0.07	1,520
REWIRE (ours)	0.37	0.74	0.80	0.14	1,520
Hodge Router	0.42	0.78	0.85	0.19	1,520
CiteSeer					
GCN-Jaccard	0.33	0.33	0.36	0.00	587
GCN-SVD	0.07	0.14	0.15	0.08	1,596
Pro-GNN	0.26	0.62	0.68	0.05	1,430
PTDNet	0.13	0.10	0.11	0.00	440
GPS	0.13	0.27	0.29	0.05	1,100
ComFy	0.15	0.29	0.32	0.06	1,100
LLM4RGNN	0.17	0.34	0.38	0.04	1,100
REWIRE (ours)	0.34	0.69	0.75	0.14	1,100
Hodge Router	0.42	0.75	0.81	0.19	1,100
PubMed					
GCN-Jaccard	0.36	0.36	0.40	0.00	7,102
GCN-SVD	0.07	0.15	0.16	0.08	19,286
Pro-GNN	0.30	0.70	0.77	0.06	17,290
PTDNet	0.14	0.11	0.12	0.00	5,320
GPS	0.15	0.29	0.32	0.08	13,300
ComFy	0.16	0.33	0.35	0.10	13,300
LLM4RGNN	0.20	0.39	0.43	0.07	13,300
REWIRE (ours)	0.38	0.77	0.84	0.14	13,300
Hodge Router	0.43	0.84	0.91	0.19	13,300
DBLP					
GCN-Jaccard	0.25	0.25	0.28	0.00	16,934
GCN-SVD	0.05	0.10	0.11	0.06	45,976
Pro-GNN	0.21	0.49	0.54	0.08	40,243
PTDNet	0.10	0.08	0.09	0.00	12,688
GPS	0.10	0.20	0.22	0.07	31,720
ComFy	0.11	0.22	0.24	0.08	31,720
REWIRE (ours)	0.27	0.54	0.59	0.14	31,720

Table 11: Edge level recovery under Metattack at 15%. Precision and recall are those of the edits against the perturbation, ins. removed is the share of inserted edges that were deleted, del. recovered is the share of deleted edges that were re-added, and edits is the number of changed edges. Reweighting policies make no edits and are omitted. DBLP has no node text, so LLM4RGNN is not run there and the Hodge Router coincides with REWIRE.

Selection	$\mathcal{B} = 20$	50	100	200
Accuracy (%)				
Disagreement d_e (ours)	79.42±0.60	80.00±0.51	80.31±0.48	80.47±0.55
Random	78.74±0.53	78.81±0.49	78.93±0.52	79.02±0.47
GPS boundary only	79.05±0.61	79.31±0.62	79.52±0.49	79.63±0.54
Cycle contribution only	79.12±0.59	79.44±0.53	79.66±0.61	79.78±0.61
Classifier uncertainty	78.93±0.50	79.15±0.54	79.36±0.51	79.45±0.51
Precision of the arbiter's edits against the perturbation (%)				
Disagreement d_e (ours)	73.8	68.9	63.5	57.2
Random	30.6	29.8	29.1	28.7
Abstention rate (%)				
Disagreement d_e (ours)	11.8	10.9	10.3	9.8

Table 12: Effect of the query budget $\mathcal{B}$ and of the edge selection rule under Metattack at 15%, pooled over the three datasets with node text. At $\mathcal{B} = 0$ the router is the residual gate, at 78.65%. Every selection rule issues the same number of unique queries at a given budget.

Variant	Cora	CiteSeer	PubMed	Δ
Hodge Router (full)	78.14±0.56	74.55±0.66	87.32±0.38	0.00
w/o GPS (rank by cycle contribution)	76.02±0.61	72.37±0.72	85.71±0.45	-1.97
w/o Hodge (rank by GPS)	71.94±0.49	68.52±0.70	82.83±0.62	-5.57
w/o arbiter ($\mathcal{B} = 0$)	76.80±0.58	73.01±0.70	86.13±0.40	-1.36
Always-Reweight (gate off)	70.59±0.52	69.48±0.51	79.10±0.41	-6.95
Always-Rewire (gate off)	76.95±0.65	73.16±0.88	86.32±0.38	-1.19
Random verdicts	76.51±0.60	72.80±0.71	85.94±0.44	-1.59
K: random antisymmetric	77.02±0.63	73.61±0.74	86.58±0.47	-0.93
K: prototype cosine	78.01±0.57	74.36±0.68	87.20±0.40	-0.15
K: unit antisymmetric	77.83±0.58	74.29±0.69	87.11±0.41	-0.26
Min–max instead of rank normalization	77.90±0.57	74.30±0.67	87.15±0.39	-0.22

Table 13: Accuracy (%, mean±std) under Metattack at 15% when one component is removed or replaced. Each row is defined in Appendix D. Δ is the mean change over the three datasets relative to the full method.

Arbiter variant	Accuracy	Precision	Abstention	Agree (clean)	Agree (annot.)
Closed model (default)	80.00±0.55	68.9	10.9	80.2	82.6
Open model	79.58±0.53	62.4	16.3	74.1	75.9
Text truncated to title	79.24±0.53	58.1	22.8	70.6	72.3
Abstract truncated to half	79.71±0.59	64.7	14.1	76.8	78.4
Endpoint texts shuffled	78.47±0.61	27.9	31.4	51.0	50.7
Random verdicts	78.42±0.53	26.8	0.0	49.8	49.2
Always YES	78.21±0.50	23.6	0.0	46.1	45.4
Always NO	78.96±0.56	33.1	0.0	53.9	54.6
Adaptive attacker	79.31±0.51	55.2	12.7	66.8	68.1

Table 14: The arbiter under model substitution, text degradation, verdict replacement, and an adaptive attacker, at $\mathcal{B} = 50$ with disagreement selection, under Metattack at 15% pooled over the three graphs with node text. Accuracy is that of the resulting classifier. Precision is that of the edits on the audited edges against the perturbation. The two agreement columns are the share of nonzero verdicts that match the clean graph and the share that match an annotator consensus on 50 audited edges per dataset.

Backbone	Observed graph	Always-Reweight	Always-Rewire	Hodge Router
GCN	70.80±0.72	73.06±0.48	78.81±0.67	80.00±0.55
GAT	70.87±0.84	71.62±0.79	78.05±0.66	79.13±0.61
APPNP	71.93±0.61	73.84±0.58	79.37±0.52	80.46±0.49
GraphSAGE	69.64±0.72	71.02±0.70	77.24±0.63	78.31±0.58

Table 15: Accuracy (%) of four backbones trained on the observed graph and on the graphs produced by each policy, under Metattack at 15% pooled over Cora, CiteSeer, and PubMed. The flow, the gate, and the edits are computed once from the frozen GCN and reused for every backbone. GAT and GraphSAGE ignore edge weights, so their Always-Reweight column reports the unweighted candidate support. The GCN row pools the corresponding cells of Table 3.

Policy	Cora	CiteSeer	PubMed	DBLP	Rewire frac.
Always-Reweight	80.26±0.28	60.95±0.52	83.26±0.19	72.81±0.84	0.000
Always-Rewire	79.82±0.34	60.42±0.54	83.81±0.22	71.31±0.79	1.000
Random gate	80.05±0.31	60.73±0.53	83.60±0.20	72.55±0.81	0.456
ESNR gate	80.09±0.31	60.77±0.53	83.57±0.20	72.50±0.81	0.468
Residual gate	80.46±0.31	61.15±0.53	83.85±0.20	72.90±0.81	0.456
Hodge Router	80.82±0.31	61.50±0.53	84.13±0.20	72.90±0.81	0.456
Per instance oracle	80.54±0.21	61.22±0.52	83.87±0.24	73.02±0.84	0.495

Table 16: Macro-F1 (%) under random corruption when the frozen classifier is replaced by soft assignments to feature centroids, so that the flow, the gate, and the edge scores use no training labels. Columns as in Table 2. The ESNR gate still reads all labels, so it and the oracle are references. On DBLP, which has no node text, the Hodge Router coincides with the residual gate.

Setting	Cora	CiteSeer	PubMed
Default ($\lambda = 2$, $\nu = 2$, $b = \eta$, clip $\ln 4$)	78.14±0.56	74.55±0.66	87.32±0.38
$\lambda = 1/2$	76.98±0.59	73.55±0.70	86.52±0.43
$\lambda = 1$	77.41±0.58	73.92±0.69	86.84±0.41
$\nu = 4$	77.86±0.60	74.21±0.70	87.10±0.42
$\nu = 8$	77.53±0.64	73.97±0.73	86.91±0.46
$b = \eta/2$	76.32±0.55	73.08±0.66	85.87±0.40
$b = 2\eta$	77.60±0.62	74.02±0.71	86.95±0.45
weight clip $\ln 2$, $\gamma \leq 4$	78.06±0.57	74.47±0.67	87.28±0.39
no weight clip	77.95±0.58	74.40±0.68	87.21±0.40

Table 17: Accuracy (%) of the Hodge Router under Metattack at 15% when one hyperparameter is changed and the others stay at their defaults. η is the perturbation rate. Below $\lambda = 2$ a verdict no longer moves an edge past every unaudited edge (Proposition 4), so the arbiter loses part of its effect.

Dataset	Flow	Solve	Scorer	Queries	Tokens	USD	Total	Baseline
Cora	0.02	0.07	6.4	31	17,360	0.0026	8.5	44.0
CiteSeer	0.02	0.08	4.4	27	15,120	0.0023	6.3	37.1
PubMed	0.18	0.57	56.8	38	21,280	0.0032	73.5	328.6
PolBlogs	0.01	0.06	5.0	0	0	0.0000	6.7	29.5
DBLP	0.13	0.62	38.8	0	0	0.0000	53.6	305.2
ogbn-arxiv	1.13	4.60	374.9	0	0	0.0000	500.6	2073.7

Table 18: Cost per instance. Flow, solve, scorer, and total are wall clock seconds of the flow construction, the Laplacian solve, the link scoring, and the whole router. Queries and tokens are the fresh arbiter queries after caching and the input tokens they consume, and USD is their price at the provider's list rate. Baseline is the total time of JDR, the strongest baseline, on the same hardware; on ogbn-arxiv JDR is run for timing only. On ogbn-arxiv the router runs without the arbiter.