Probabilistic Task-Adaptive Graph Rewiring

Chendi Qian [*1] **Andrei Manolache** [*2 3] **Kareem Ahmed** [4] **Zhe Zeng** [4] **Guy Van den Broeck** [4] **Mathias Niepert** [2] **Christopher Morris** [1]

Abstract

Message-passing graph neural networks (MPNNs) emerged as powerful tools for processing graph-structured input. However, they operate on a fixed graph structure, ignoring potential noise and missing information. In addition, due to their purely local aggregation mechanism, they are susceptible to phenomena such as over-smoothing, over-squashing, or under-reaching. Hence, devising principled approaches for learning to focus on graph structure relevant to the given prediction task remains an open challenge. In this work, leveraging recent progress in differentiable k-subset sampling, we devise a novel task-adaptive graph rewiring approach, which learns to add relevant edges while omitting less beneficial ones. We empirically demonstrate on synthetic datasets that our approach effectively alleviates the issues of over-squashing and under-reaching. In addition, on established real-world datasets, we demonstrate that our method is competitive or superior to conventional MPNN models and graph transformer architectures regarding predictive performance and computational efficiency.

1. Introduction

Graph-structured data is widespread across several application domains, including chemo- and bioinformatics (Barabasi & Oltvai, 2004; Reiser et al., 2022), combinatorial optimization (Cappart et al., 2023), and social-network analysis (Easley & Kleinberg, 2010), underlining the importance of machine learning methods tailored to graphs. In recent years, *message-passing graph neural networks* (MPNNs) (Gilmer et al., 2017; Scarselli et al., 2009) emerged as the dominant paradigm in this area. However, due to their purely local aggregation mechanism, MPNNs are inherently biased towards encoding local structure and unable to capture global or long-range information, often linked to phenomena such as *under-reaching* (Barceló et al., 2020) or *over-squashing* (Alon & Yahav, 2021), with the latter being heavily investigated in recent works.

Concretely, the issue of over-squashing, as detailed in Alon & Yahav (2021), refers to the excessive compression of information from distant nodes due to a source node's extensive receptive field, occurring when too many layers are stacked. Recent works aim to alleviate over-squashing by resorting to *graph rewiring*, i.e., adding edges between distant nodes to make the exchange of information more accessible.

Theoretically, Topping et al. (2021); Bober et al. (2022) investigated over-squashing through the lens of Ricci and Forman curvature. Refining Topping et al. (2021), Di Giovanni et al. (2023) analyzed how the architectures' width and graph structure contribute to the over-squashing problem, showing that over-squashing happens among nodes with high commute time, stressing the importance of rewiring techniques. In addition, Deac et al. (2022) utilized expander graphs to enhance message passing and connectivity, while Karhadkar et al. (2022) resort to spectral techniques, and Banerjee et al. (2022) proposed a greedy random edge flip approach to overcome over-squashing. Many studies have suggested different versions of multi-hop-neighbor-based message passing to maintain long-range dependency (Abboud et al., 2022; Abu-El-Haija et al., 2019; Gasteiger et al., 2019; Gutteridge et al., 2023; Xue et al., 2023), which can as well be interpreted as a heuristic rewiring scheme. The above works indicate that graph rewiring is an effective strategy to mitigate over-squashing.

Different from the above, graph transformers (Dwivedi et al., 2022b; He et al., 2022; Müller et al., 2023; Rampášek et al., 2022; Chen et al., 2022) and similar global attention mechanisms (Liu et al., 2021; Wu et al., 2022) marked a shift from local to global message passing, aggregating over all nodes. While not understood in a principled way, empirical studies indicate that graph transformers possibly alleviate over-squashing; see, e.g., Müller et al. (2023). However, due

[*]Equal contribution [1]RWTH Aachen, Germany [2]University of Stuttgart, Germany [3]Bitdefender, Romania [4]University of California, Los Angeles, USA. Correspondence to: Chendi Qian <chendi.qian@log.rwth-aachen.de>, Andrei Manolache <andrei.manolache@ipvs.uni-stuttgart.de>.

Published at the Differentiable Almost Everything Workshop of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. July 2023.

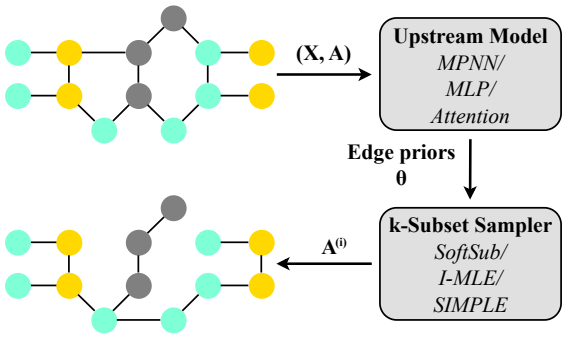

Figure 1. Our sampling scheme. The upstream model extracts edge priors θ from the initial graph, which are forwarded to a sampling module. After sampling, we obtain N rewired graphs processed by a downstream model. The pipeline is end-to-end trainable.

to their global aggregation mode, computing an attention matrix with n^2 entries for an n-order graph makes them applicable only to small or mid-sized graphs. Further, to capture non-trivial graph structure, they need to resort to hand-engineered positional or structural encodings.

In summary, most approaches for circumventing over-squashing either resort to heuristic rewiring approaches, possibly not adapting to the given prediction task, or computational heavy global attention mechanisms.

Present Work. By leveraging recent progress in differentiable k-subset sampling, we derive *task-adaptive graph rewiring*. Concretely, we utilize an *upstream model* to learn prior scores for candidate edges. We then utilize the scores to parameterize a probability distribution constraint by so-called k-subset constraints. Subsequently, we sample multiple k-edge adjacency matrices from this distribution and process them using a *downstream model*, typically a GNN, for the final predictions task. To make this pipeline trainable via gradient descent, we adapt recently proposed discrete gradient estimation and tractable sampling techniques (Xie & Ermon, 2019; Niepert et al., 2021; Ahmed et al., 2023). We empirically demonstrate on synthetic datasets that our approach effectively alleviates the issues of over-squashing and under-reaching. In addition, on established real-world datasets, we establish that our method is competitive or superior to conventional MPNN models and graph transformer architectures regarding predictive performance and computational efficiency.

2. Probabilistic Data-driven Graph Rewiring

Here, we outline our task-adaptive graph rewiring approach based on recent advancements in discrete gradient estimation and tractable sampling techniques (Ahmed et al., 2023; Niepert et al., 2021; Xie & Ermon, 2019).

Let $\mathfrak{A}_n$ denote the set of adjacency matrices of graphs on n nodes. Further, let G be a graph with $V(G) \coloneqq \{1, \ldots, n\}$, an adjacency matrix $\mathbf{A}(G) \in \mathfrak{A}_n$, and node attribute matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ for $d > 0$. That is, each row in the matrix $\mathbf{X}$ corresponds to a node's initial feature in the graph G. Our task-adaptive graph rewiring maintains a (parameterized) *upstream model* $h_{\boldsymbol{v}} \colon \mathfrak{A}_n \times \mathbb{R}^{n \times d} \to \Theta$, typically a neural network, parameterized by $\boldsymbol{v}$, mapping an adjacency matrix and corresponding node attributes to unnormalized edge priors $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^{n \times n}$.

In the following, we use the priors $\boldsymbol{\theta}$ as parameters of a probability distribution,

$$p_{\boldsymbol{\theta}}(\mathbf{A}(G)) \coloneqq \prod_{i,j=1}^{n} p_{\theta_{ij}}(\mathbf{A}(G)_{ij}),$$

where $p_{\theta_{ij}}(\mathbf{A}(G)_{ij} = 1) = \operatorname{sigmoid}(\theta_{ij})$ and $p_{\theta_{ij}}(\mathbf{A}(G)_{ij} = 0) = 1 - \operatorname{sigmoid}(\theta_{ij})$. Unlike prior probabilistic rewiring approaches (Franceschi et al., 2019), we introduce dependencies between the graph's edges by conditioning the distribution $p_{\theta_{ij}}(\mathbf{A}(G))$ on a *k-subset constraint*. That is, the probability of sampling any given k-edge adjacency matrix $\mathbf{A}(G)$, becomes

$$p_{(\boldsymbol{\theta},k)}(\mathbf{A}(G)) \coloneqq \begin{cases} p_{\boldsymbol{\theta}}(\mathbf{A}(G))/Z & \text{if } \|\mathbf{A}(G)\|_1 = k, \\ 0 & \text{otherwise,} \end{cases}$$

where

$$Z \coloneqq \sum_{\mathbf{B} \in \mathfrak{A}_n \colon \|\mathbf{B}\|_1 = k} p_{\boldsymbol{\theta}}(\mathbf{B}).$$

The original graph is now rewired into a new adjacency matrix $\bar{\mathbf{A}}$ by combining N samples $\mathbf{A}^{(i)} \sim p_{(\boldsymbol{\theta},k)}(\mathbf{A}(G))$ for $i \in [N]$ together with the original adjacency matrix $\mathbf{A}(G)$ using a differentiable aggregation function $g \colon \mathfrak{A}_n^{(N+1)} \to \mathfrak{A}_n$, i.e., $\bar{\mathbf{A}} \coloneqq g(\mathbf{A}(G), \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}) \in \mathfrak{A}_n$. In practice, we rewire our graphs by sampling two adjacency matrices for deleting edges and adding new edges, i.e., $g(\mathbf{A}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}) \coloneqq (\mathbf{A}(G) - \mathbf{A}^{(1)}) + \mathbf{A}^{(2)}$ where $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ are two sampled adjacency matrices with a possibly different number of edges, respectively. Finally, the rewired adjacency matrix is used in a *downstream model* $f_{\boldsymbol{u}} \colon \mathfrak{A}_n \times \mathbb{R}^{n \times d} \to \mathcal{Y}$, typically an MPNN, with parameters $\boldsymbol{u}$ and $\mathcal{Y}$ the prediction target set.

Learning to Sample. Given a graph G with adjacency matrix $\mathbf{A}(G) \in \mathfrak{A}_n$ and attribute matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, we are now concerned with learning the parameters $\boldsymbol{\omega} = (\boldsymbol{v}, \boldsymbol{u})$ of the architecture through minimizing the expected loss

$$L(\mathbf{A}(G), \mathbf{X}, y; \boldsymbol{\omega}) \coloneqq \mathbb{E}_{\mathbf{A}^{(i)} \sim p_{(\boldsymbol{\theta},k)}(\mathbf{A}(G))}\left[L\right],$$

where

$$L = \ell\left(f_{\boldsymbol{u}}\left(g\left(\mathbf{A}(G), \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}\right), \mathbf{X}\right), y\right)$$

with $y \in \mathcal{Y}$, ℓ a point-wise loss such as the cross-entropy or MSE, and $\boldsymbol{\theta} = h_{\boldsymbol{v}}(\mathbf{A}(G), \mathbf{X})$.

For minimizing the above expectation using gradient descent and backpropagation, we need to efficiently sample from $p_{(\boldsymbol{\theta},k)}$ and estimate the gradients regarding the parameters $\boldsymbol{\theta}$. Here, we adapt recently proposed discrete gradient estimation and tractable sampling techniques (Xie & Ermon, 2019; Niepert et al., 2021; Ahmed et al., 2023).

Sampling. To sample an adjacency matrix $\mathbf{A}(G)$ from the distribution $p_{(\boldsymbol{\theta},k)}(\mathbf{A}(G))$ conditioning on the k-edge constraint, and further to allow the sampling to be trained end-to-end, we investigate the use of three recently proposed gradient estimators in our pipeline, SoftSub (Xie & Ermon, 2019), I-MLE (Niepert et al., 2021), and Simple (Ahmed et al., 2023). Concretely, SoftSub extends the Gumbel-Softmax trick to sample a relaxed k-subset amenable to auto-differentiation on the backward pass, with its complexity being $\mathcal{O}(n^2 k)$ due to the relaxed top-k operation and a vectorized complexity being $\mathcal{O}(k \log n)$. Alternatively, I-MLE performs approximate sampling using perturb-and-map (PAM) on the forward pass. On the backward pass, it proposes to use the marginal distribution as a proxy to the sampled matrix for the derivative computation, that is,

$$\nabla_{\boldsymbol{\theta}} L \approx \partial_{\boldsymbol{\theta}} \mu(\boldsymbol{\theta}) \nabla_{\mathbf{A}} \ell \text{ with } \mu(\boldsymbol{\theta}) \coloneqq \{p_{(\boldsymbol{\theta},k)}(\mathbf{A}(G)_{ij})\}_{i,j=1}^{n},$$

and further approximates the marginals using PAM samples. Instead of resorting to approximations, the Simple gradient estimator derives an exact sampling algorithm to sample from the k-edge adjacency matrix distribution $p_{(\boldsymbol{\theta},k)}(\mathbf{A}(G))$ on the forward pass, and couples it with an exact and efficient computation of the marginals $\mu(\boldsymbol{\theta})$ that is differentiable on the backward pass, to achieve lower bias and variance via the exact computations, whose vectorized complexity is $\mathcal{O}(\log k \log n)$.

Upstream Model. For the upstream model $h_{\boldsymbol{v}}(\mathbf{A}, \mathbf{X})$, we use either an MPNN or a transformer followed by an attention layer. In the case of MPNNs, we use the GIN layer (Xu et al., 2019). That is, given a graph G, in each layer $t \in [T]$, we compute a feature

$$\mathbf{h}_i^{(t)} = \gamma^{\ell}\Big(\big(1 + \epsilon^{(t)}\big) \cdot \mathbf{h}_i^{(t)} + \sum_{j \in \mathcal{N}(i)} \mathbf{h}_j^{(t-1)}\Big),$$

for node $i \in V(G)$, with $\gamma^{(t)}$ a multi-layer perceptron (MLP), ϵ a learnable parameter, $\mathbf{h}_i^0 = \mathbf{X}_i$, and $\mathcal{N}(i)$ is the neighborhood of node i. For an edge $(i,j) \in E(G)$, we compute $\boldsymbol{\theta}_{ij} = \phi([\mathbf{h}_i^T || \mathbf{h}_j^T]) \in \mathbb{R}$, where $[\cdot||\cdot]$ is the concatenation operator and ϕ is an MLP.

Alternatively, to be less dependent on local structure, we utilize a *self-attention* (SA) block (Vaswani et al., 2017) for the calculation of prior scores, i.e.,

$$\boldsymbol{\theta} = \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} \in \mathbb{R}^{n \times n},$$

where d_k denotes the number of columns of the matrices $\mathbf{Q}$ and $\mathbf{K}$. The matrices $\mathbf{Q}$ and $\mathbf{K}$ are the result of projecting $\mathbf{X}$ linearly, i.e., $\mathbf{Q} \coloneqq \mathbf{X}\mathbf{W}^Q$ and $\mathbf{K} \coloneqq \mathbf{X}\mathbf{W}^K$, where $\mathbf{W}^Q$ and $\mathbf{W}^K \in \mathbb{R}^{d \times d_k}$. In addition, to capture graph structure, i.e., information regarding $\mathbf{A}(G)$, we column-wise concatenate structural or position encoding (Dwivedi & Bresson, 2020; Müller et al., 2023; Min et al., 2022; Kreuzer et al., 2021) to the initial attribute matrix $\mathbf{X}$. In addition, to circumvent the quadratic complexity, we compute priors for only a subset of node pairs by utilizing a heuristic, e.g., based on shortest-path distances.

Downstream Model. For all of our sampling and base experiments, we use an MPNN with GIN layers (Xu et al., 2019). For the instances where we have access to edge features, we employ the GINE variant (Hu et al., 2020) for edge feature processing. For graph-level tasks, we employ a pooling function, such as mean or sum pooling, while for node-level tasks, we take the node embedding $\mathbf{h}_i^T$. The final embeddings are then processed and projected to the target space by an MLP.

3. Experimental Evaluation

Here, we aim to investigate to what extent our task-adaptive graph rewiring leads to improved predictive performance on synthetic and real-world datasets. Concretely, we answer the following questions.

Q1 Does task-adaptive graph rewiring alleviate over-squashing and under-reaching on synthetic datasets?

Q2 Does task-adaptive graph rewiring translate to boosted predictive performance on (a) graph-level molecular prediction tasks and (b) heterophilic node-level prediction tasks?

The source code of all methods and evaluation procedures will be made available in a public repository. We refer to Appendix B for details on the experimental protocol and model configurations.

Datasets. To answer **Q1**, we utilized the Trees-NeighborsMatch dataset, as in (Alon & Yahav, 2021). Additionally, we created the Trees-LeafCount dataset to investigate whether our method could mitigate under-reaching issues; see Appendix A for details. To answer **Q2** (a), we used the established molecular graph-level regression datasets Alchemy (Chen et al., 2019) and Zinc (Jin et al., 2017; Dwivedi et al., 2020). To answer **Q2** (b), we used the established Cornell, Wisconsin, Texas node-level classification datasets (Pei et al., 2020).

Baseline and Model Configurations. We compare our rewiring approaches with the base downstream model, both with and without positional embeddings; see Appendix B).

Table 1. Quantitative results on the ALCHEMY dataset and the heterophilic and transductive WEBKB datasets. **Best overall**; **Second best**; **Best Non-GT**. Rewiring outperforms the base models on all of the datasets. Graph transformers have an advantage over both the base models and the ones employing rewiring.

		ALCHEMY + EDGE ↓	HETEROPHILIC & TRANSDUCTIVE CORNELL ↑	TEXAS ↑	WISCONSIN ↑
OURS	BASE	11.12 ± 0.690	0.574 ± 0.006	0.674 ± 0.010	0.697 ± 0.013
	BASE W. PE	7.197 ± 0.094	0.540 ± 0.043	0.654 ± 0.010	0.649 ± 0.018
	HALF TRF.	**7.135 ± 0.171**	0.501 ± 0.014	0.597 ± 0.023	0.630 ± 0.016
	REWIRE$_{Sim}$	**6.447 ± 0.057**	**0.623 ± 0.029**	**0.706 ± 0.014**	**0.750 ± 0.015**
GTS	GPS (LAPPE)	-	0.662 ± 0.038	**0.778 ± 0.010**	0.747 ± 0.029
	GPS (RWSE)	-	**0.708 ± 0.020**	**0.775 ± 0.012**	**0.802 ± 0.022**
	GPS (DEG)	-	**0.718 ± 0.024**	0.773 ± 0.013	**0.798 ± 0.090**
	GRAPHORMER (DEG)	-	0.683 ± 0.017	0.767 ± 0.017	0.770 ± 0.019
	GRAPHORMER (DEG + ATTN BIAS)	-	0.683 ± 0.017	0.767 ± 0.017	0.770 ± 0.019

Further, we compare to standard graph transformer baseline (Müller et al., 2023), GPS (Rampášek et al., 2022), and SAT (Chen et al., 2022), two state-of-the-art graph transformer models. We utilize the same upstream $h_{\mathbf{v}}$ and downstream $f_{\mathbf{u}}$ architectures for the rewiring experiments. To rewire the graphs, for ALCHEMY, we learn to add 10 new edges while removing 40 existing edges. For ZINC, we add 80 edges and remove 20 edges. Additionally, in the case of the ZINC dataset, we compare multiple sampling schemes, using either GUMBEL SOFTSUB-ST (Maddison et al., 2017; Jang et al., 2017; Xie & Ermon, 2019), I-MLE (Niepert et al., 2021), or SIMPLE (Ahmed et al., 2023) in terms of predictive power and computation time.

Experimental Results and Discussion. Concerning **Q1**, our rewiring method achieves perfect test accuracy up to a problem radius of 6 on both the TREES-NEIGHBORSMATCH and the TREES-LEAFCOUNT datasets, see Figure 3. For the TREES-LEAFCOUNT dataset, our model can create connections directly from the leaves to the root, achieving perfect accuracy with a downstream model containing a single MPNN layer. We provide a qualitative result in Figure 2 and a detailed discussion in Appendix A. Concerning **Q2** (a), the results in Table 1 and Table 2 show that our rewiring methods consistently outperform the base models on both ZINC and ALCHEMY, and are competitive or better than the state-of-the-art GPS and SAT graph transformer methods. Hence, our results indicate that task-adaptive graph rewiring can improve performance for molecular prediction tasks. Concerning **Q2** (b), Table 1 showcases the performance gains of our rewiring method over the base models, indicating that data-driven rewiring has the potential of alleviating the *effects* of over-smoothing by removing undesirable edges and making new ones between nodes with similar features. The graph transformer methods outperform both the rewiring approach and the base models. We speculate that GIN's aggregation mechanism for the downstream models is a limiting factor on heterophilic data and leave the analysis of combining task-adaptive graph rewiring with downstream models that address over-smoothing for future investigations.

Table 2. Quantitative results on the ZINC dataset. The two columns refer to using edge features or not. **Best overall**; **Best with GIN backbone**; **Second best**. All of the rewiring approaches (GUMBEL, I-MLE, SIMPLE) are using the same hyperparameters. For the Structure-Aware Transformer (SAT), we report the results by (Chen et al., 2022) for both the GIN backbone and the best overall MPNN backbone. For GPS, we report the results by (Rampášek et al., 2022). The REWIRE$_{\text{SIM}}$ rewiring method uses the SIMPLE gradient estimator and outperforms all of the models using GIN layers while being competitive with state-of-the-art graph transformers.

		ZINC - EDGE ↓	ZINC + EDGE ↓
GIN	K-ST SAT	0.166 ± 0.007	0.115 ± 0.005
	K-SG SAT	0.162 ± 0.013	**0.095 ± 0.002**
	BASE	0.258 ± 0.006	0.207 ± 0.006
	BASE W. PE	0.162 ± 0.001	0.101 ± 0.004
	HALF TRF.	0.154 ± 0.005	0.109 ± 0.005
	REWIRE$_{\text{GMB}}$	0.153 ± 0.003	0.103 ± 0.008
	REWIRE$_{\text{IMLE}}$	**0.151 ± 0.001**	0.104 ± 0.008
	REWIRE$_{\text{SIM}}$	**0.139 ± 0.001**	**0.092 ± 0.004**
BEST	GPS	-	**0.070 ± 0.004**
	K-ST SAT	0.164 ± 0.007	0.102 ± 0.005
	K-SG SAT	**0.131 ± 0.002**	**0.094 ± 0.008**

4. Conclusions

Here, we utilized recent advances in differentiable k-subset sampling to devise a novel task-adaptive graph rewiring approach, which learns to add relevant edges while omitting less beneficial ones. On synthetic datasets, we demonstrated that our approach effectively alleviates the issues of over-squashing and under-reaching. In addition, on established real-world datasets, we showed that our method is competitive or superior to conventional MPNN models and graph transformer architectures regarding predictive performance and computational efficiency.

5. Acknowledgements

AM and MN acknowledge funding by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 – 390740016, the support by the Stuttgart Center for Simulation Science (SimTech), and the International Max Planck Research School for Intelligent Systems (IMPRS-IS). CQ and CM are partially funded by a DFG Emmy Noether grant (468502433) and RWTH Junior Principal Investigator Fellowship under Germany's Excellence Strategy. ZZ is supported by an Amazon Doctoral Student Fellowship.

References

Abboud, R., Dimitrov, R., and Ceylan, İ. İ. Shortest path networks for graph property prediction. *ArXiv preprint*, 2022.

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Steeg, G. V., and Galstyan, A. MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. *ArXiv preprint*, 2019.

Ahmed, K., Zeng, Z., Niepert, M., and Van den Broeck, G. Simple: A gradient estimator for k-subset sampling. In *International Conference on Learning Representations*, 2023.

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

Banerjee, P. K., Karhadkar, K., Wang, Y. G., Alon, U., and Montúfar, G. Oversquashing in GNNs through the lens of information contraction and graph expansion. *ArXiv preprint*, 2022.

Barabasi, A.-L. and Oltvai, Z. N. Network biology: Understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

Barceló, P., Kostylev, E. V., Monet, M., Pérez, J., Reutter, J., and Silva, J. P. The logical expressiveness of graph neural networks. In *International Conference on Learning Representations*, 2020.

Bober, J., Monod, A., Saucan, E., and Webster, K. N. Rewiring networks for graph neural network training using discrete geometry. *ArXiv preprint*, 2022.

Cappart, Q., Chételat, D., Khalil, E. B., Lodi, A., Morris, C., and Veličković, P. Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24(130):1–61, 2023.

Chen, D., O'Bray, L., and Borgwardt, K. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2022.

Chen, G., Chen, P., Hsieh, C.-Y., Lee, C.-K., Liao, B., Liao, R., Liu, W., Qiu, J., Sun, Q., Tang, J., Zemel, R., and Zhang, S. Alchemy: A quantum chemistry dataset for benchmarking ai models. *ArXiv preprint*, 2019.

Deac, A., Lackenby, M., and Veličković, P. Expander graph propagation. *ArXiv preprint*, 2022.

Di Giovanni, F., Giusti, L., Barbero, F., Luise, G., Lio', P., and Bronstein, M. On Over-Squashing in message passing neural networks: The impact of width, depth, and topology. *ArXiv preprint*, 2023.

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs. *ArXiv preprint*, 2020.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *ArXiv preprint*, 2020.

Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2022a.

Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., and Beaini, D. Long range graph benchmark. *ArXiv preprint*, 2022b.

Easley, D. and Kleinberg, J. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

Franceschi, L., Niepert, M., Pontil, M., and He, X. Learning discrete structures for graph neural networks. In *International conference on machine learning*, pp. 1972–1982, 2019.

Gasteiger, J., Weißenberger, S., and Günnemann, S. Diffusion improves graph learning. *ArXiv preprint*, 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272, 2017.

Gutteridge, B., Dong, X., Bronstein, M., and Di Giovanni, F. DRew: Dynamically rewired message passing with delay. *ArXiv preprint*, 2023.

He, X., Hooi, B., Laurent, T., Perold, A., LeCun, Y., and Bresson, X. A generalization of ViT/MLP-Mixer to graphs. *ArXiv preprint*, 2022.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Jin, W., Coley, C. W., Barzilay, R., and Jaakkola, T. Predicting organic reaction outcomes with weisfeiler-lehman network. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 2604–2613, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Karhadkar, K., Banerjee, P. K., and Montúfar, G. FoSR: First-order spectral rewiring for addressing oversquashing in GNNs. *ArXiv preprint*, 2022.

Kreuzer, D., Beaini, D., Hamilton, W. L., Létourneau, V., and Tossou, P. Rethinking graph transformers with spectral attention. *ArXiv preprint*, 2021.

Liu, M., Wang, Z., and Ji, S. Non-Local graph neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, PP, 2021.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

Min, E., Chen, R., Bian, Y., Xu, T., Zhao, K., Huang, W., Zhao, P., Huang, J., Ananiadou, S., and Rong, Y. Transformer for graphs: An overview from architecture perspective. *ArXiv preprint*, 2022.

Müller, L., Galkin, M., Morris, C., and Rampásek, L. Attending to graph transformers. *ArXiv preprint*, 2023.

Niepert, M., Minervini, P., and Franceschi, L. Implicit MLE: Backpropagating through discrete exponential family distributions. *ArXiv preprint*, 2021.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer. *ArXiv preprint*, 2022.

Reiser, P., Neubert, M., Eberhard, A., Torresi, L., Zhou, C., Shao, C., Metni, H., van Hoesel, C., Schopmans, H., Sommer, T., et al. Graph neural networks for materials science and chemistry. *Communications Materials*, 3(1): 93, 2022.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. *ArXiv preprint*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *ArXiv preprint*, 2017.

Wu, Z., Jain, P., Wright, M. A., Mirhoseini, A., Gonzalez, J. E., and Stoica, I. Representing Long-Range context for graph neural networks with global attention. *ArXiv preprint*, 2022.

Xie, S. M. and Ermon, S. Reparameterizable subset sampling via continuous relaxations. *International Joint Conference on Artificial Intelligence*, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Xue, R., Han, H., Torkamani, M., Pei, J., and Liu, X. LazyGNN: Large-Scale graph neural networks via lazy propagation. *ArXiv preprint*, 2023.

Table 3. Overview of used hyperparameters.

DATASET	HIDDEN SIZE	GIN LAYERS	BATCH SIZE	MLP LAYERS
ZINC	256	4	128	3
ALCHEMY	256	4	128	3
HETEROPHILIC	128	2	-	3
SYNTHETIC	32	1	1024	3

A. Datasets

Here, we give additional information regarding the datasets.

Synthetic Dataset. For the TREES-LEAFCOUNT dataset, we fix a problem radius $R > 0$ and retrieve the binary representation of all numbers fitting into 2^R bits. This construction allows us to create 2^R unique binary trees by labeling the leaves with "0" and "1" corresponding to the binary equivalents of the numbers. A label is then assigned to the root node, reflecting the count of leaves tagged with "1". From the resulting graphs, we sample to ensure an equal class distribution. The task requires a model to predict the root label, thereby requiring a strategy capable of conveying information from the leaves to the root.

We aim to have a controlled environment to observe if our upstream model h_v can sample meaningful edges for the new graph configuration. Conventionally, a minimum of R message-passing layers is required to accomplish both tasks (Barceló et al., 2020; Alon & Yahav, 2021). However, a single-layer upstream MPNN could trivially resolve both datasets, provided the rewired graphs embed direct pathways from the root node to the leaf nodes containing the label information. To circumvent any potential bias within the sampling procedure, we utilize the self-attention mechanism described in Section 2 as our upstream model h_v, along with a single-layer GIN architecture serving as the downstream model f_u. For each problem radius, we sample exactly $k = 2^D$ edges. Indeed, our method consistently succeeded in correctly rewiring the graphs in all tested scenarios, extending up to a problem radius of $R = 6$, and achieved perfect test accuracy on both datasets. Figure 2 presents a qualitative result from the TREES-LEAFCOUNT dataset, further illustrating the capabilities of our approach.

B. Hyperparameter and Training Details

Experimental Protocol. Table 3 details our base models' hyperparameters. The HALF TRANSFORMER model contains two transformer layers with a hidden size of 64, followed by the BASE model. For the upstream model, we do a hyperparameter search for the number of added and deleted edges, hidden size, and network depth for each rewiring experiment. For all our experiments, we use early stopping with an initial learning rate of 0.001 that we decay by half on a plateau.

We compute each experiment's mean and standard deviation with different random seeds over three runs. We take the best results from the literature for the graph transformer models. We evaluate test predictive performance based on validation performance. In the case of the WEBKB datasets, we employ a 10-fold cross-validation with the official data splits. Except for the BASE model, our models use positional and structural embeddings concatenated to the initial node features. Specifically, we add both RWSE and LAPPE (Dwivedi et al., 2022a). We use the same downstream model as the base model for the rewiring models.

C. Additional Experimental Results

Here, we report on the computation times of different variants of our task-adaptive graph rewiring schemes and results on synthetic datasets.

Training Times. We report the average training time per epoch in Tab. 4. The RANDOM entry refers to using random adjacency matrices as rewired graphs.

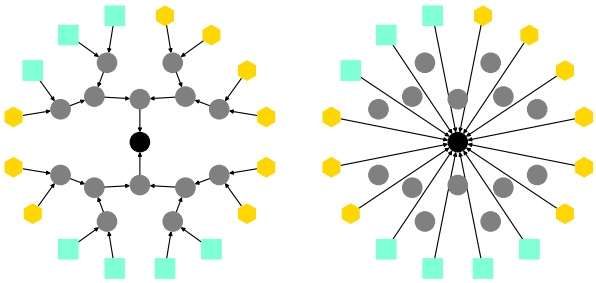

Figure 2. A sample from the TREES-LEAFCOUNT dataset with the problem radius $R = 4$, after 100 training epochs. Left: original graph, right: rewired graph. After just one round of message-passing, the root node can obtain the label information from the leaves.

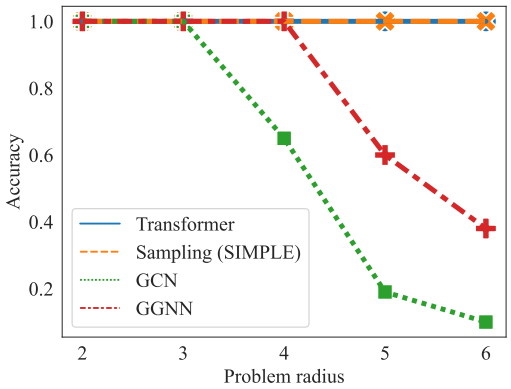

Figure 3. Test accuracy of our rewiring method on the TREES-NEIGHBORSMATCH dataset, compared to the reported accuracies from (Müller et al., 2023).

Table 4. Computations time for rewiring approaches on ZINC. Average over five epochs. Number of added edges: 80. Number of deleted edges: 20. Experiments performed on a machine with a single Nvidia RTX A5000 GPU and a Intel i9-11900K CPU.

SAMPLER	TIME/EPOCH (S)
RANDOM	8.54 ± 0.04
GUMBEL SOFTSUB-ST	11.87 ± 0.05
I-MLE	11.24 ± 0.11
SIMPLE	11.53 ± 0.07