Delaunay Rewiring to Avoid Over-Squashing and Over-Smoothing in Graphs

Edwin Roussin [*1] **Tristan Waddington** [*1]

Abstract

This document reviews the graph rewiring method proposed by (Attali et al., 2024) based on Delaunay triangulation that aims to avoid over-smoothing and over-squashing during prediction tasks. It uses notions of edge curvature to measure the quality of the rewiring. The method is tested on a variety of graph datasets and compared to another one.

1. Delaunay Rewiring

Graph neural networks (GNNs) have emerged as the standard approach for effective learning graph-structured data. GNNs employ an iterative approach, updating node representations through the local aggregation of information from neighboring nodes, known as the message-passing paradigm (Gilmer et al., 2017). However, this iterative process can lead to over-smoothing and over-squashing, where the node representations become too similar to each other, or ineffective to transmit long-range information. In these cases, the model performance can degrade.

1.1. Over-Smoothing and Over-Squashing

Over-Smoothing Message-passing neural networks (MPNN) use an iterative approach, updating node representations through the local aggregation of information from neighboring nodes. The need to stack additional layers to capture non-local interactions tends to make node features more and more similar, leading to over-smoothing. This is particularly problematic when the graph is heterophilic, i.e., when nodes belong to different communities (Zheng et al., 2022).

Over-Squashing The squashing effect occurs when the model is unable to transmit long-range information, leading to a loss of information. MPNN models try to capture exponentially growing information into fixed sized representations. (Alon & Yahav, 2021) have shown the correlation between overs-quashing and bottleneck in the graph structure. Models such as Graph Convolutional Networks (GCN) are known to suffer from this issue because they absorb the information from all edges equally.

1.2. Edge Curvature

The main metrics used to characterize graph structure and measure the quality of the rewiring is the discrete Ricci edge curvature. A complete definition can be found on Appendix A.2. Previous work has shown that:

- Positive curvature edges establish connections between nodes belonging to the same community. Highly positive curved edges cause over-smoothing (Nguyen et al., 2023).
- Negative curvature edges connect nodes from different communities. Highly negative curved edges cause over-squashing (Topping et al., 2022).

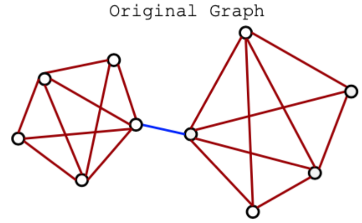

Figure 1. Example graph: in red the edges with positive curvature ($\sim$ 3), in blue with negative curvature (-1.2) (Attali et al., 2024)

Hence, we get a simple metric for further experiments where we expect to see the edge curvature amplitude decrease after the rewiring process.

1.3. Rewiring

1.3.1. Former methods' limitation

Existing methods mitigate over-squashing by rewiring the input graph to minimize structural bottlenecks. First ones rely on the analysis of the graph structure, through local or global features like edge curvature or resistance. However, these methods may not scale well with the number of nodes

[*]Equal contribution [1]Institut Polytechnique de Paris, Palaiseau, France. Correspondence to: <firstname.lastname@polytechnique.edu>.

Final Exam for APM_5DS30_TP, Machine Learning with Graphs, IPP, 2025.

and need depends on the choice of hyperparameters. Moreovoer, they modify the original graph, which is not always available in some applications.

Conversely, over-smoothing is avoided by preventing embedding to become the same through: **Normalization** with PairNorm (Zhao & Akoglu, 2020); or **rewiring** dropping edges, at random (Rong et al., 2019) or in finding the potential good ones (Giraldo et al., 2023)

1.3.2. DELAUNAY TRIANGULATION

Delaunay rewiring Is an extreme **4 steps rewiring** method illustrated bellow.

1. A first GNN[1] constructs **node embeddings** from the original graph.
2. Reduce the embedding with **UMAP** in dim 2.
3. **Rebuilt edges with Delaunay triangulation** from their distance in UMAP embedding space.
4. Second GNN **mix** the Delaunay graph with the original features from the beginning.

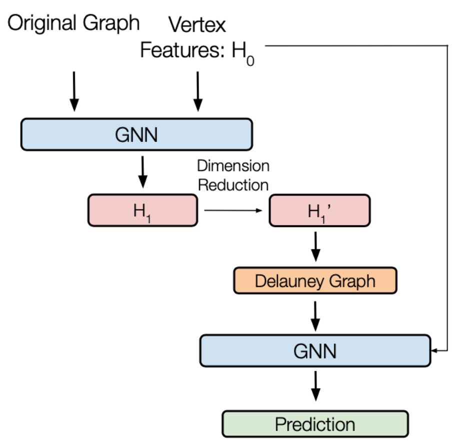

Figure 2: Illustration of the rewiring method using the features obtained by a GNN.

Delauynay Algorithm

GNN embedding

Initial throughts The method is remarkably simple, as it does not require any hyperparameters according to authors, which eliminates the need for a grid search. Its computational complexity is efficient, scaling as $\mathcal{O}(N \log N)$. Additionally, the method constructs a graph directly from the embedding, making it independent of the presence of the original graph.

[1]GCN from (Kipf & Welling, 2017)

The use of UMAP is restricted to two dimensions. This is because performing triangulation in higher dimensions increases computation time and results in denser graphs[2]. This increased density reduced the accuracy in experiments, making the two-dimensional approach more practical and effective.

Authors have added the first GNN late in their research as the Delaunay Graph need quality embedding. However, it raises questions about whether these initial representation are victims of smoothing and squashing effects by this GNN. Does this GNN effectively captures long-range dependencies?

1.4. Experiments

We habve reproduce the rewiring experiment on the **Wisconsin dataset**[3]. The report in Table 5 demonstrate substantial improvements in graph neural network performance. The effect on of the Delaunay rewiring is clearly visible on the degree distribution in the Figure 2 below.

Key Results: GCN accuracy improved from 54.90% to 67.55% $(+12.6\%)$. GAT accuracy improved from 55.88% to 69.12% $(+13.2\%)$. Graph homophily increased by 96% $(0.366 \rightarrow 0.718)$. All improvements are statistically significant $(p < 0.0001)$.

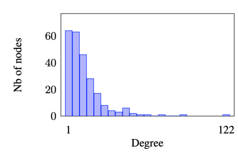

Figure 8: Histogram of the degree distribution for the original Wisconsin graph

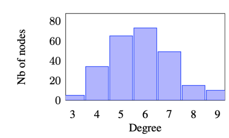

Figure 9: Histogram of the degree distribution for the Delaunay Wisconsin graph

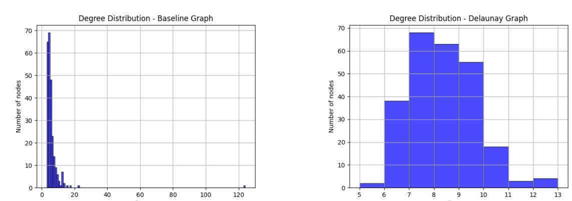

Figure 2. Effect of the Delaunay rewiring on degree distribution. Left: original, Right: after rewiring, Top: [Attali al., 2024] (Attali et al., 2024), Bottom: ours.

Impact: The Delaunay rewiring approach successfully addresses over-squashing and improves graph structure, leading to significant performance gains across different model architectures.

Validation: Results are robust across multiple experiments

[2]Generalized triangles in three dimensions have six edges, while in four dimensions, they have ten edges.

[3]From WebKB dataset, 251 nodes = web pages from Wisconsin connected by edges = hyperlinks, node features = bag-of-words in dim 1703, labels = 5 kind of author.

Table 1. Comparison of Baseline and Delaunay Graph Metrics

Metric	Orignal	Delaunay Graph
Mean Degree	5.59	7.85
Homophily	0.366	0.710 (↑ 96%)
Curvature Range	[-0.475, 0.250]	[-0.214, 0.200]

and statistically significant, with comprehensive analysis of graph properties supporting the improvements.

1.5. Limitations

Dimensionality Reduction We might lose some feature information during the UMAP reduction to 2 dimensions. The quality of Delaunay graph depends on quality of reduced features.

Computational Considerations The method complexity is in $\mathcal{O}\big(N \log N\big)$ only, but the two graphs are fully loaded into memory. Furthermore, the UMAP and curvature computation can cause overhead, but this is way better than the quadratic cost of other methods.

Parameter Sensitivity We do not totally agree with the authors as we find out that the impact of UMAP parameters not fully explored. They could modulate the quality of the embedding we already discussed. Moreover, the potential dependence on feature normalization and the effect of different train/val/test splits is not extensively studied.

1.6. Future Work

If we were to work further in this topic, we could focus on:

- **Algorithmic Improvements**: Investigate higher-dimensional Delaunay triangulation, explore sparse approximations for larger graphs, Develop incremental/streaming versions for large-scale graphs or optimize UMAP parameter selection.
- **Analysis Extensions**: Study impact on different graph properties, investigate relationship between feature space and graph structure, compare with other rewiring methods on the same dataset, or analyze feature importance in graph construction.
- **Ablation Studies**: Compare with other dimensionality reduction method, the effect of different feature preprocessing, the sensitivity to hyperparameters and try different triangulation algorithms.

2. Additional Paper

We have chosen to dig into another rewiring method proposed by (Wilson et al., 2024) based on Cayley graphs, named **Cayley Graph Propagation**. The authors have focus on the creation of bottleneck-free structure to ameliorate the over-squashing effect.

2.1. Choice explanation

We have chosen this paper because it follows the exact same idea of rewiring the initial graph using a mathematical structure. The Delaunay Triangulation on the first side used geometric properties. The Cayley graph on the second side is a mathematical structure that can be used to represent a group. It seems to us that the two methods are competitive and complementary. The two papers are from 2024 and does not refer to each other.

2.2. Paper summary

(Wilson et al., 2024) focuses on the over-squashing effect. Their aim is to precompute bottleneck-free graph structures to alleviate the over-squashing effect. They propose the use of a well-known **expander graph** family: the Cayley graphs of the $\mathrm{SL}(2, \mathbb{Z}_n)$ special linear group as a computational template for GNNs.

They have improved previous work already used this family that required to truncate the graph to align with the input graph. They show that truncation is not necessary and that the complete Cayley graph can be used directly with the direct effect of reducing the diameter of the graph as illustrated in the Figure 3.

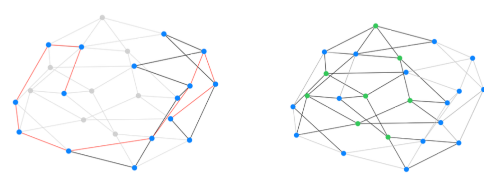

Figure 3. Both Cayley graphs represent $SL(2, \mathbb{Z}_3)$ with $|V| = 24$ nodes using the same construction. Left: A truncated Cayley graph (spectral gap: 0.0751, diameter: 10) aligned to a given input graph. Right: The complete Cayley graph (spectral gap: 1.2679, diameter: 4) structure indicating the additional virtual nodes (in green). Source: (Wilson et al., 2024)

The authors have shown that the spectral gap of the Cayley graph is a good indicator of the quality of the graph structure. A larger spectral gap defines a strong connectivity, or alternatively the global lack of bottlenecks. The spectral gap is the difference between the first and second normalized eigenvalues of the graph Laplacian.

2.3. Comparison

Software and Data

The original code repository and additional work can be found on GitHub[4]. The code is written in Python and uses the PyTorch library. The code is available under the MIT license.

Acknowledgements

We thank our professor Mr. Jhony H. Giraldo for presenting us this article and the theoretical foundations to understand it.

Impact Statement

This paper highlight the promising yet simple method of Delauynay Rewirin to improve the performance of graph-based machine learning models. We hope to see this method adopted.

References

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

Attali, H., Buscaldi, D., and Pernelle, N. Delaunay Graph: Addressing Over-Squashing and Over-Smoothing Using Delaunay Triangulation. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=uyhjKoaIQa.

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. PMLR, 2017.

Giraldo, J. H., Skianis, K., Bouwmans, T., and Malliaros, F. D. On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, pp. 566–576. ACM, October 2023. doi: 10.1145/3583780.3614997. URL http://dx.doi.org/10.1145/3583780.3614997.

Huang, Y., Zeng, Y., Wu, Q., and Lü, L. Higher-order graph convolutional network with flower-petals laplacians on simplicial complexes, 2024. URL https://arxiv.org/abs/2309.12971.

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

Nguyen, K., Hieu, N. M., Nguyen, V. D., Ho, N., Osher, S., and Nguyen, T. M. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *International Conference on Machine Learning*, pp. 25956–25979. PMLR, 2023.

Ni, C.-C., Lin, Y.-Y., Gao, J., Gu, X. D., and Saucan, E. Ricci curvature of the internet topology, 2015. URL https://arxiv.org/abs/1501.04138.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2019.

Topping, J., Giovanni, F. D., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature, 2022. URL https://arxiv.org/abs/2111.14522.

Wilson, J., Bechler-Speicher, M., and Veličković, P. Cayley graph propagation, 2024. URL https://arxiv.org/abs/2410.03424.

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2020.

Zheng, X., Liu, Y., Pan, S., Zhang, M., Jin, D., and Yu, P. S. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, 2022.

[4] https://github.com/waddason/Delaunay-Rewiring

A. Additional details on the Delaunay Rewiring method

A.1. Delaunay Triangulation

The Delaunay triangulation is a method to construct a graph from a set of points in a space. It is a well-known method in computational geometry and has been used in various applications.

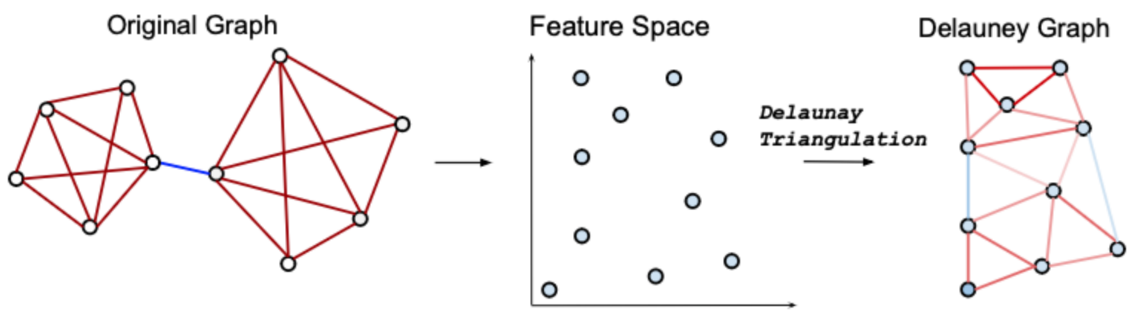

Figure 4. Illustration of the Delaynay rewiring process in a 2-dimension feature space.

Definition A.1. A Delaunay triangulation, denoted as $DT(P)$, for a set P of points in the d-dimensional Euclidean space, is a triangulation where no point in P resides within the circum-hypersphere of any d-simplex in $DT(P)$.

A.2. Edge Curvature

We describe bellow the two main curvature metrics used in the Delaunay rewiring method.

Paper : Balance Forman Curvature (Topping et al., 2022) is computed over cycles of size 4.

Balance Forman Curvature:

$$c_{ij} = \frac{2}{d_i} + \frac{2}{d_j} - 2 + 2\frac{\sharp_\Delta}{\max(d_i, d_j)} + \frac{\sharp_\Delta}{\min(d_i, d_j)} + \frac{\max(\sharp_\square^i, \sharp_\square^j)^{-1}}{\max(d_i, d_j)}(\sharp_\square^i + \sharp_\square^j)$$

where $\sharp_\Delta$ is the number of triangles based at e_{ij}, $\sharp_\square^i$ is the number of 4-cycles based at e_{ij} starting from i without diagonals inside.

Experiment : **Oliver-Ricci Curvature** (Ni et al., 2015)

We used the specific implementation from `GraphRicciCurvature.OllivierRicci`. Node Ricci curvature is defined as the average of all it's adjacency edge. A visual representation of the curvature of edges in a graph is shown in the 5 below from the original paper.

A.3. UMAP

Method presentation Uniform Manifold Approximation and Projection (UMAP) is a dimensionality reduction technique that can be used for visualisation similarly to t-SNE, but also for general non-linear dimension reduction. UMAP constructs a high dimensional graph representation of the data then optimizes a low-dimensional graph to be as structurally similar as possible.

- **Speed**: UMAP is faster than t-SNE.
- **Global structure**: UMAP preserves more of the global structure.
- **Separation**: clearly separate groups of similar categories.

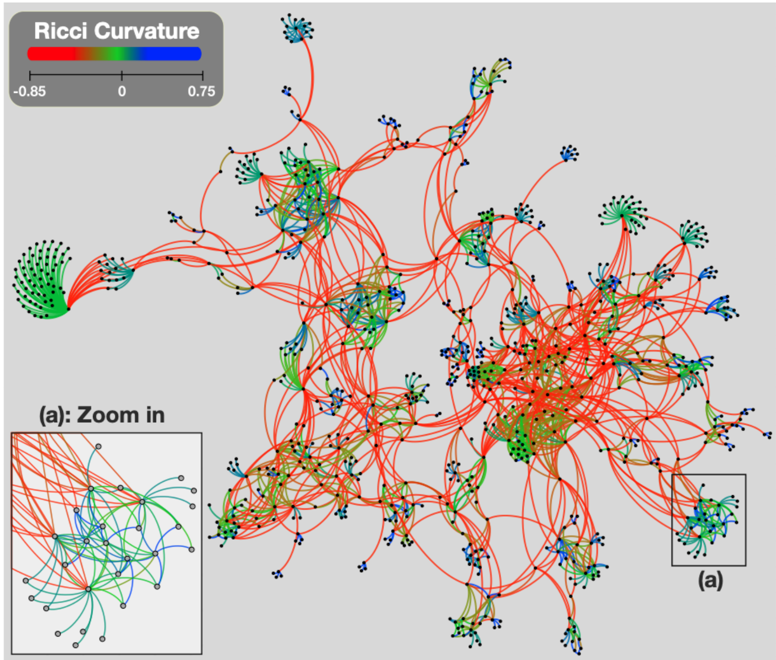

Figure 5. It shows the Ricci curvature of each edge in a router level graph (Exodus(US)) from the Rocketfuel data set with 895 nodes and 2071 edges. Negatively curved edges (in red) behave like "backbones", maintaining the connectivity of clusters that are grouped by zero and positively curved edges (in green and blue). Source (Ni et al., 2015)

Dimensionality reduction technique is not perfect - by necessity, we're distorting the data to fit it into lower dimensions - and UMAP is no exception. But it is a powerful tool to visualize and understand large, high-dimensional datasets.

Hyperparameters choice Most common: `n_neighbors` and `min_dist`, control the balance between local and global structure. They have a significant impact on the resulting embedding, and the choice of these parameters is crucial. A visual example of their effect on the same two linked circles is shown in the Figure 6 below.

- `n_neighbors`: number of neighbors used to construct the high-dimensional graph.
- `min_dist`: minimum distance between points in the low-dimensional space.

B. Experiment Details on the Delaunay Rewiring method

B.1. Wisconsin Dataset

WebKB is a dataset that includes web pages from computer science departments of various universities. It represents $4,518$ web pages that are categorized into 6 imbalanced categories (Student, Faculty, Staff, Department, Course, Project). Additionally there is Other miscellanea category that is not comparable to the rest. We have used the Wisconsin subset of the dataset, which contains 251 web pages from the University of Wisconsin-Madison.

We used the classical dataset splitting from Train/Val/Test Split: $60\%/20\%/20\%$ as proposed by (Attali et al., 2024). We add our own data preprocessing in normalizaing the features. Our experiment modestly reaches a 69.12% accuracy on the Wisconsin dataset, which has been outperformed by other methods since 2019 has shown in the leaderboard bellow. [5]

The results obtained by (Attali et al., 2024) are summarized in the Table 2 below.

[5]Highest accuracy on the Wisconsin dataset is 90.5% by (Huang et al., 2024), using Higher-order Graph Convolutional Network (HiGCN) grounded in Flower Petals Laplacians, capable of discerning intrinsic features across varying topological scales.

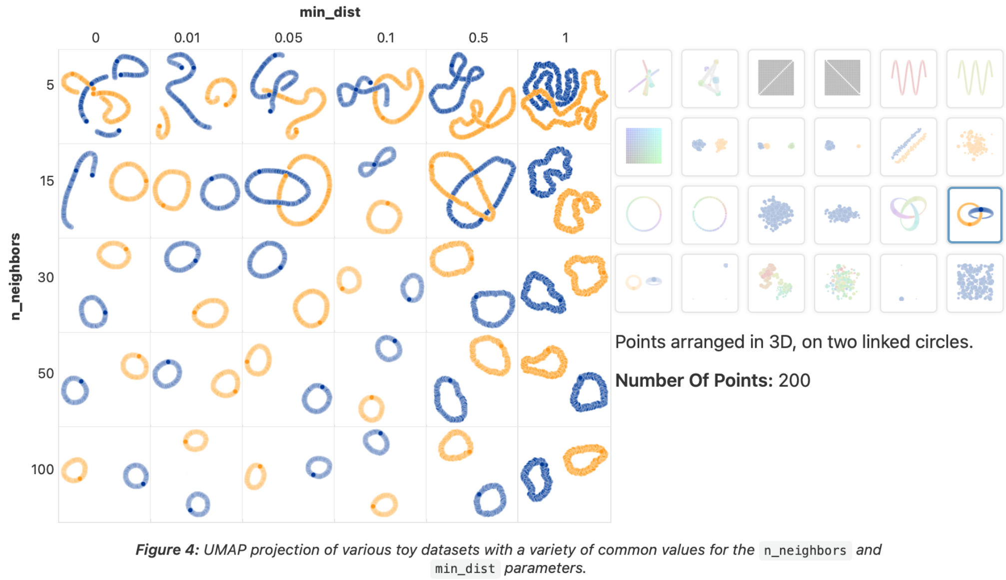

Figure 6. Illustration of UMAP hyperparameters from Google PAIR

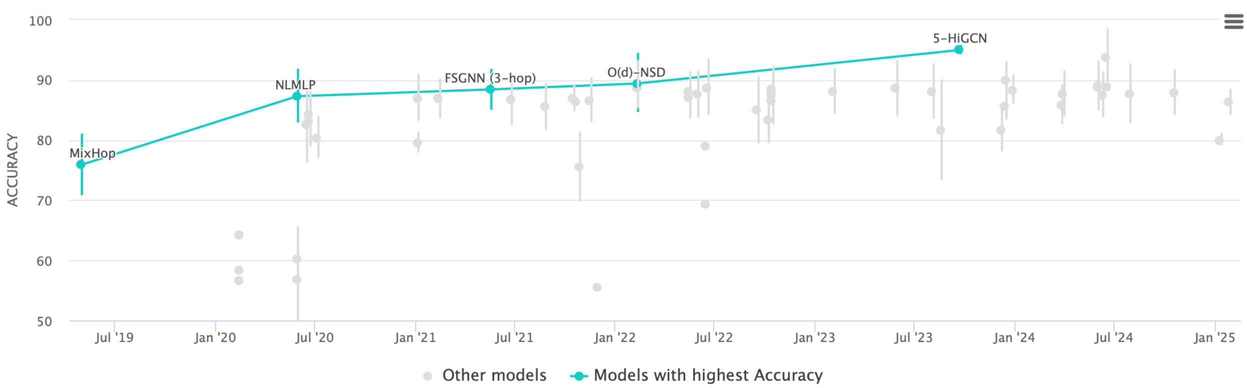

Figure 7. Performance of GNN on the Wisconsin dataset, from Paper With Code.

Table 2. Experimental results with Delaunay Rewiring in different dimensions reduction for Wisconsin dataset

Metric	Original	DR (dim=2)	DR (dim=3)	DR (dim=4)	DR (dim=5)	DR (dim=7)	Ours
Homophily	0.06	0.65	0.63	0.60	0.54	0.44	0.71
Number of edges	499	1470	2534	4064	6266	16148	-
Max degree	24	14	24	42	57	167	14
Mean degree	12	6	12	18	28	66	8
Accuracy GCN (%)	55.12 ± 1.51	**70.98 ± 1.5**	69.45 ± 1.5	68.59 ± 1.5	68.55 ± 1.5	66.42 ± 1.7	67.55
Accuracy GAT (%)	46.05 ± 1.49	**74.33 ± 1.24**	74.23 ± 1.4	70.75 ± 1.4	72.43 ± 1.5	67.16 ± 1.7	69.12
Triangulation time (s)	-	≤ 1	≤ 1	≤ 1	2	50	≤ 1

B.2. Graph Neural Networks

We used two popular GNN architectures: Graph Convolutional Networks (GCN) and Graph Attention Networks (GAT).

Table 3. Hyperparameters for GCN and GAT Models

Hyperparameter	GCN	GAT
Hidden Channels	32	32
Layers	2 (ReLU activation)	2 (8 attention heads in 1st layer, 1 in 2nd)
Dropout	0.5	0.5
Learning Rate	0.005	0.005
Weight Decay	5×10^{-6}	5×10^{-6}
Training Time per Epoch	0.1s (empirical)	0.2s (empirical)

B.3. Experimental setup

Hardware and Software The experiments were conducted on a CUDA-enabled GPU with the following software dependencies:

- **Device**: CUDA-enabled GPU with `PyTorch Geometric, UMAP, NetworkX, GraphRicciCurvature`
- **Preprocessing**: Feature normalization.
- **Runs**: 10 per experiment, max 2000 epochs, early stopping patience 100 epochs.

Preprocessing Time:

- UMAP dimensionality reduction: 1-2 seconds.
- Delaunay triangulation: < 1 second.
- Curvature calculation: $\sim 3 - 5$ seconds per graph.
- Total preprocessing overhead: $\sim 5 - 8$ seconds.

Table 4. Training Metrics Comparison

Average Epochs Until Convergence	Baseline	Delaunay	s/epoch
GCN	$\sim$ 150 epochs	$\sim$ 130 epochs	0.1s
GAT	$\sim$ 180 epochs	$\sim$ 160 epochs	0.2s

Memory Usage Peak memory during preprocessing: $\sim 2GB$, Training memory footprint: Baseline: $\sim$ $1GB$; Delaunay: $\sim 1.2GB$. Additional storage for results: < 100MB.

B.4. Results

Our experiment modestly reaches a 69.12% accuracy on the Wisconsin dataset, which has been outperformed by other methods since 2019 has shown in the leaderboard Figure 7. The results are summarized in Table 5.

Table 5. Comparison of Baseline and Delaunay Graph Metrics

Metric	Orignal	Delaunay Graph
Mean Degree	5.59	7.85
Homophily	0.366	0.710 (↑ 96%)
Curvature Range	[-0.475, 0.250]	[-0.214, 0.200]
GCN accuracy	54.90%	67.55% (↑ 12.6%)
GAT accuracy	55.88%	69.12% (↑ 13.2%)

Conclusion The Delaunay rewiring approach demonstrates significant and consistent improvements on the Wisconsin dataset. The improvements are not only substantial in magnitude $(12.6 - 13.2\%)$ but also statistically significant, with both GCN and GAT models benefiting from the rewiring. The enhanced graph properties (improved homophily and reduced negative curvature) provide structural evidence for why the approach works well. While there are some limitations and areas for future investigation, the current results strongly support the effectiveness of this approach for improving graph neural network performance.