# 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)[Attali et al, 2024] based on Delauynay 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 B. 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).

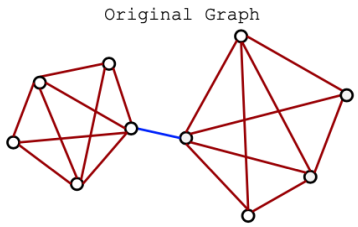


*Figure 1.* Example graph: in red the edges with positive curvature ($\sim 3$), in blue with negative curvature (-1.2) (Attali et al., 2024)

**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: <first-name.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.

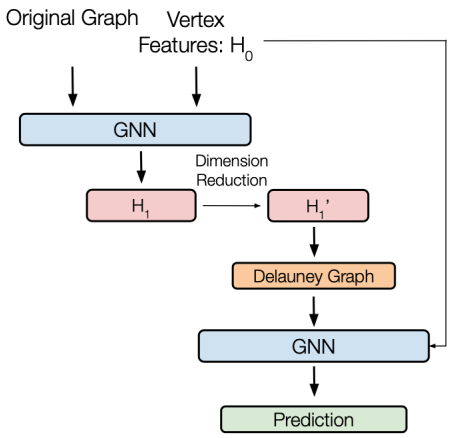


Figure 2: Illustration of the rewiring method using the features obtained by a GNN.

**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 $\mathcal{O}\big(N \log N\big)$. Additionally, the method constructs a graph directly from the embedding, making it independent of the presence of the original graph.

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]. This report presents a comprehensive evaluation of the Delaunay rewiring approach, demonstrating substantial improvements in graph neural network performance:

**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 ($p < 0.0001$)

*Table 1.* Comparison of Baseline and Delaunay Graph Metrics

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

**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 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.

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

[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.

The quality of Delaunay graph depends on quality of reduced features.

**Computational Considerations** The method complexity is in $\mathcal{O}(N \log N)$ 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

### 2.1. Choice explanation

### 2.2. Paper summary

### 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.

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

## 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.

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.

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.

# A. Additional details on the Delaunay Rewiring

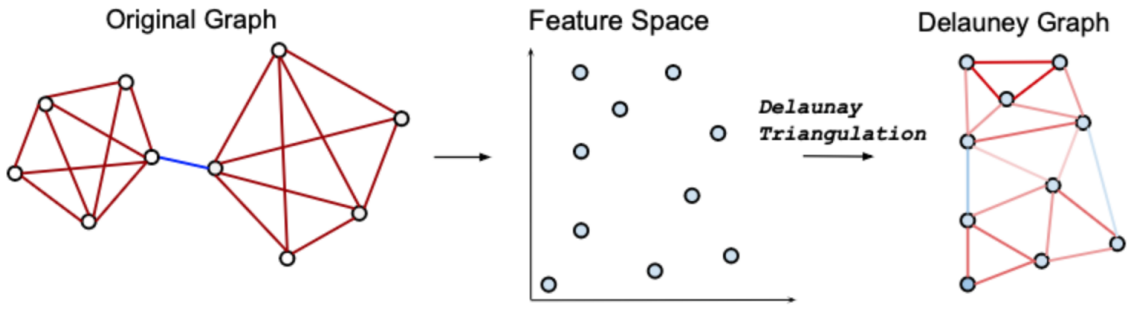


*Figure 2.* Illustration of the Delaynay rewiring process in a 2-dimension feature space.

# B. Edge Curvature

Paper: Balance Forman Curvature (Topping et al., 2022) is computed over cycles of size 4.
*Experiment: Oliver-Ricci Curvature (Ni et al., 2015)* `GraphRicciCurvature.OllivierRicci`.

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

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

## B.1. Figures

You may want to include figures in the paper to illustrate your approach and results. Such artwork should be centered, legible, and separated from the text. Lines should be dark and at least 0.5 points thick for purposes of reproduction, and text should not appear on a gray background.

Label all distinct components of each figure. If the figure takes the form of a graph, then give a name for each axis and include a legend that briefly describes each curve. Do not include a title inside the figure; instead, the caption should serve this function.

Number figures sequentially, placing the figure number and caption *after* the graphics, with at least 0.1 inches of space before the caption and 0.1 inches after it, as in Figure 3. The figure caption should be set in 9 point type and centered unless it runs two or more lines, in which case it should be flush left. You may float figures to the top or bottom of a column, and you may set wide figures across both columns (use the environment `figure*` in LaTeX). Always place two-column figures at the top or bottom of the page.

## B.2. Algorithms

If you are using LaTeX, please use the "algorithm" and "algorithmic" environments to format pseudocode. These require the corresponding stylefiles, algorithm.sty and algorithmic.sty, which are supplied with this package. Algorithm 1 shows an example.

## B.3. Tables

You may also want to include tables that summarize material. Like figures, these should be centered, legible, and numbered consecutively. However, place the title *above* the table with at least 0.1 inches of space before the title and the same after it, as in Table 2. The table title should be set in 9 point type and centered unless it runs two or more lines, in which case it should be flush left.

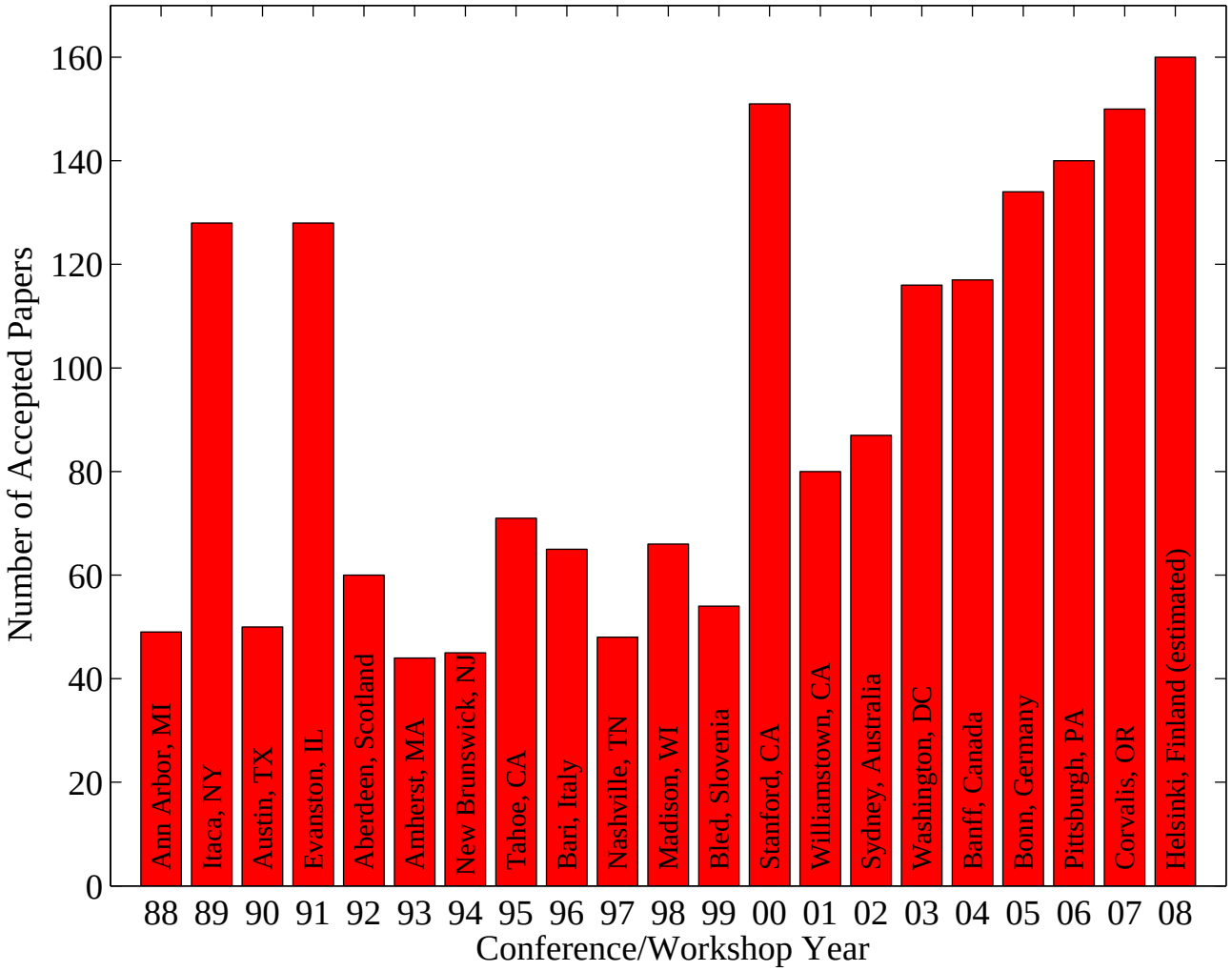


*Figure 3.* Historical locations and number of accepted papers for International Machine Learning Conferences (ICML 1993 – ICML 2008) and International Workshops on Machine Learning (ML 1988 – ML 1992). At the time this figure was produced, the number of accepted papers for ICML 2008 was unknown and instead estimated.

Tables contain textual material, whereas figures contain graphical material. Specify the contents of each row and column in the table's topmost row. Again, you may float tables to a column's top or bottom, and set wide tables across both columns. Place two-column tables at the top or bottom of the page.

### B.4. Theorems and such

The preferred way is to number definitions, propositions, lemmas, etc. consecutively, within sections, as shown below.

**Definition B.1.** A function $f : X \to Y$ is injective if for any $x, y \in X$ different, $f(x) \neq f(y)$.

Using Definition B.1 we immediate get the following result:

**Proposition B.2.** *If $f$ is injective mapping a set $X$ to another set $Y$, the cardinality of $Y$ is at least as large as that of $X$*

*Proof.* Left as an exercise to the reader. □

Lemma B.3 stated next will prove to be useful.

**Algorithm 1** Bubble Sort

**Input:** data $x_i$, size $m$
**repeat**
  Initialize $noChange = true$.
  **for** $i = 1$ **to** $m - 1$ **do**
    **if** $x_i > x_{i+1}$ **then**
      Swap $x_i$ and $x_{i+1}$
      $noChange = false$
    **end if**
  **end for**
**until** $noChange$ is $true$

*Table 2.* Classification accuracies for naive Bayes and flexible Bayes on various data sets.

| DATA SET | NAIVE | FLEXIBLE | BETTER? |
|---|---|---|---|
| BREAST | 95.9± 0.2 | 96.7± 0.2 | √ |
| CLEVELAND | 83.3± 0.6 | 80.0± 0.6 | × |
| GLASS2 | 61.9± 1.4 | 83.8± 0.7 | √ |
| CREDIT | 74.8± 0.5 | 78.3± 0.6 | |
| HORSE | 73.3± 0.9 | 69.7± 1.0 | × |
| META | 67.1± 0.6 | 76.5± 0.5 | √ |
| PIMA | 75.1± 0.6 | 73.9± 0.5 | |
| VEHICLE | 44.9± 0.6 | 61.5± 0.4 | √ |

**Lemma B.3.** *For any $f : X \to Y$ and $g : Y \to Z$ injective functions, $f \circ g$ is injective.*

**Theorem B.4.** *If $f : X \to Y$ is bijective, the cardinality of $X$ and $Y$ are the same.*

An easy corollary of Theorem B.4 is the following:

**Corollary B.5.** *If $f : X \to Y$ is bijective, the cardinality of $X$ is at least as large as that of $Y$.*

**Assumption B.6.** The set $X$ is finite.

*Remark* B.7. According to some, it is only the finite case (cf. Assumption B.6) that is interesting.