Graph Rewiring: From theory to Applications in Fairness

Tutorial on the 1st Learning on Graphs Conference 2022

Adrián Arnaiz Rodríguez  
ELLIS Alicante

Francisco Escolano  
ELLIS Alicante

Nuria Oliver  
ELLIS Alicante

Edwin Hancock  
University of York

adrian@ellisalicante.org  
@arnaiztech

@Fescolano  
@nuriaoliver
Resources

Tutorial Webpage
https://ellisalicante.org/tutorials/GraphRewiring

Slides
https://ellisalicante.org/tutorials/GraphRewiring

Video
https://ellisalicante.org/tutorials/GraphRewiring

Code
https://github.com/ellisalicante/GraphRewiring-Tutorial
Outline

1. Motivation
   - Graph Classification and Expressiveness
   - Node Classification and Over-smoothing
   - Desiderates

2. Graph Spectral Theory
   - Average Cut Problem
   - Fiedler Vector
   - Laplacian and Dirichlet Energies
   - Laplacian Eigenfunctions and Spectrum
   - Spectral Theorem
   - Spectral Commute times

3. Transductive Graph Rewiring
   - Diffusive Rewiring
   - Cheeger constant
   - Curvature Rewiring

4. Inductive Graph Rewiring
   - CT and Lovász Bound
   - CT and Sparsification
   - CT and Directional Graph Networks
   - CT-Layer
     - Loss function and CT-Layer
     - Learned CTE and CT distance
     - Experiments in Graph and Node Classification
     - CT-Layer as Differentiable Curvature
     - CT and Cheeger Constant
   - GAP Layer
     - Network derivatives / Spectral Derivatives
     - Approximation of fielder vector

5. Rewiring in Graph Fairness
   - Algorithmic Fairness
   - Structure: a New Dimension
   - Cause of Graph Bias
   - Taxonomy of Definitions
   - Graph Rewiring Methods for Fairness
Motivation and Challenges

Introduction to Spectral Theory

Transductive Rewiring

Inductive Rewiring

Graph Fairness

Panel Discussion
Motivation
Motivation and Challenges

- Introduction to Spectral Theory
- Transductive Rewiring
- Inductive Rewiring
- Graph Fairness
- Panel Discussion
Motivation
What is (or should be) graph rewiring?

Graph Rewiring pursues the optimal graph structure for the downstream task

In GRAPH CLASSIFICATION, graph rewiring skeletonizes the graph so that the structure becomes more informative

- Given an input graph (left), bottleneck-preserving rewiring (center) discriminates graphs whose differences are in the bottlenecks themselves since intra-class edges are often removed or down-weighted.
- **Gap-minimization** rewiring (right) however, discriminates graphs whose differences are in the communities.
- **Example**: Web networks such as COLLAB are better discriminated by ‘bottleneck-preserving rewiring but SBM-like networks with large bottlenecks are better discriminated by gap-minimization rewiring.
Motivation

Graph Classification and Isomorphism

Most GNNs are as powerful as 1-WL test

Distance Encodings (DE) or Positional Encodings (PE) make GNNs more powerful than 1-WL

- **PE**: Random walk measures (e.g. shortest path, diameter, commute times), Spectral metrics (e.g. eigenvectors)
- **Expressiveness**: DE or PE provides strictly more expressive power than 1-WL test
  - [Li, P. et al. 2020] [Velingker, A. et al. 2022]
- **Invariance**: Spectral GCN are permutation and sign equivariant [Lim, D. et al. 2022]
- **Usage**: Usually used as an extra node feature or to control message aggregation

Use Spectral metrics to perform Graph Rewiring

Bronstein, M. GNNs through the lens of differential geometry and algebraic topology. Blog Post, 2021. [Link]
Motivation
What is (or should be) graph rewiring?

Graph Rewiring pursues the optimal graph structure for the downstream task

In NODE CLASSIFICATION, graph rewiring enables/disables information flow between nodes.
- **Homophilic** networks (where structure is correlated with class labels) are easy to rewire (e.g. reduce the gap).
- **Heterophilic** networks often require to increase the flow between heterophilic nodes.
- In the figure above (Cornell): distant **green** nodes can access the periphery of the hub while the gap is preserved.
- **Result**: classes with high heterophilic index are better classified.
Motivation

Node Classification. Heterophily and Over-squashing.

GNNs were originally designed based on the smoothness principle

\[ h_{\text{edges}} = \frac{|\{(u, v) \in E : y_u = y_v\}|}{|E|} \]

\[ H_{ij}(E) = \frac{|\{(u, v) \in E : y_u i \land y_v = j\}|}{|\{(u, v) \in E : y_u = i\}|} \]

\[ h_{\text{nodes}} = \frac{1}{|V|} \sum_{v \in V} \frac{|\{u \in N(v) : y_u = y_v\}|}{|N(v)|} \]

\[ h_{\text{class}} = \frac{1}{|C| - 1} \sum_{c \in C} \left( h_c - \frac{|C_c|}{n} \right)_+, h_c = \frac{\sum_{v \in c} |\{u \in N(v) : y_u = y_v\}|}{\sum_{v \in C} |N(v)|} \]

i.e. Correlation between structure and labels

\[ h_{\text{smooth}} = \mathcal{E}(y) = Tr[y^T Ly] \]

\[ h = r = \text{Pearson correlation coefficient} \]

[Newman, M., 2002]

Motivation

Node Classification. Heterophily and Over-squashing.

Heterophily → Problem with higher radius

Under-reaching
\[ k < r \]

\[ \uparrow k \]

Over-smoothing
\[ k > r \approx d \]

Over-squashing
node’s receptive field increases exponentially

Graph Rewiring as a solution
- Connect distant nodes to overcome the three problems.
- E.g. increase bottleneck

Original graph
Over-squashing

Dominant class in each community absorbs the other

Alon, U. and Yahav, E. “On the bottleneck of graph neural networks and its practical implications”. In ICLR 2021
Motivation

Key Challenges – Desiderates of Graph Rewiring

- Principled
- Expressive
- Interpretable
- Task-aligned
- Parameter-Free

Global Propagation
Remove bottlenecks
Low Complexity
No dedicated preprocessing
Preserve the original structure

[Deac, A. et al., 2019]

Deac, A., Lackenby, M. and Velicković, P. “Expander Graph Propagation”. In Learning on Graphs Conference (LoG), 2022.
\[
\begin{pmatrix}
\Phi & \Lambda & \Phi^T
\end{pmatrix} =
\begin{pmatrix}
\cdots & \cdots & \cdots
\end{pmatrix}
\begin{pmatrix}
\lambda_0 & 0 & 0 \\
0 & \ddots & 0 \\
0 & 0 & \lambda_n
\end{pmatrix}
\begin{pmatrix}
\cdots & \cdots & \cdots
\end{pmatrix}
\begin{pmatrix}
\cdots & \cdots & \cdots \\
v_1 & \cdots & v_n
\end{pmatrix}
\]
Graphs as Combinatorial Objects

Understanding the Graph Laplacian

• Undirected Graph
  
  \[ G = (V, E), V = \{1,2, ..., n\} e_{ij} \in E \subseteq V \times V \]

• Adjacency Matrix:
  
  \[
  \begin{bmatrix}
  0 & 1 & 1 & 0 \\
  1 & 0 & 1 & 0 \\
  1 & 1 & 0 & 1 \\
  0 & 0 & 1 & 0 \\
  \end{bmatrix}
  \]

  \[ A_{ij} \quad \text{As a variable} \]

  \[ a_{ij}, w_{ij} \quad \text{As a weight} \]

  \( G \) as a Combinatorial Object: \( 2^n \) functions \( f \)

• Function over the nodes: \( f : V \rightarrow \mathbb{R} \)

• Example: \( f : V \rightarrow \{-1,1\} \)
The Average Cut Problem
Understanding the Graph Laplacian

• What $f$'s are more informative about $G$?

$f(i) = f(j) \forall e_{ij}$

$\begin{array}{c}
\begin{array}{c}
2 \\
3 \\
4 \\
5 \\
6
\end{array}
\end{array}$

Harmonic

$f(i) = \frac{1}{|N_i|} \sum_j a_{ij} f(i) = \frac{1}{d_i} \sum_j a_{ij} f(i)$

$\begin{array}{c}
\begin{array}{c}
2 \\
3 \\
4 \\
5 \\
6
\end{array}
\end{array}$

Piecewise Harmonic

Vertex Partition
$A \cup B = V$
$A \cap B = \emptyset$

Cut

$f = \min_{f \in \Omega} \text{Acut}(A, B)$

$\text{Acut}(A, B) = \frac{\text{cut}(A, B)}{|A|} + \frac{\text{cut}(A, B)}{|B|}$

NP-Hard!
Fiedler’s Theorem: Measures the variability of the optimal solution

\[ x \in \{-1,1\}^n \quad x_i = +1 \rightarrow i \in A, x_i = -1 \rightarrow i \in B \]

\[ x \perp 1 \quad \text{Minimal variability is } \lambda_1 = 0, \text{ i.e. that of the harmonic function} \]

The variability \( \lambda_2 \) of the optimal partition minimizes the ratio between the variability imposed by the structure of the graph and the unconstrained one!

\[
\lambda_2 = n \cdot \min \left\{ \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} (x_i - x_j)^2}{\sum_{i=1}^{n} \sum_{j=1}^{n} (x_i - x_j)^2} : x \neq c \cdot 1, c \in \mathbb{R} \right\}.
\]
The Fiedler Vector
Understanding the Graph Laplacian

\[ G \in SBM(P) \]
\[ P = \begin{bmatrix} p_1 & q_{12} \\ q_{12} & p_2 \end{bmatrix} \]

\[ \lambda_1 = 0 \]

Fiedler vector and its mapping over \( G \)

\[ x = \begin{bmatrix} 0.25351771 \\ 0.25653022 \\ 0.25387927 \\ 0.16959223 \\ 0.17525303 \\ 0.24266911 \\ 0.24454324 \\ 0.1106077 \\ 0.2855154 \\ 0.19132828 \\ -0.25326518 \\ -0.24898363 \\ -0.15099328 \\ -0.17707374 \\ -0.1441682 \\ -0.2553871 \\ -0.24018788 \\ -0.25043498 \\ -0.19223463 \\ -0.2707059 \end{bmatrix} \]

\[ \lambda_2 = 8.37042078 \times 10^{-1} \]
The Combinatorial Laplacian
Understanding the Graph Laplacian

The combinatorial Laplacian
\( L = D - A, D = \text{diag}(d_1, \ldots, d_n) \)

\[
L = \begin{bmatrix}
d_1 & -a_{12} & \cdots & -a_{1n} \\
-a_{21} & d_2 & \cdots & -a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
-a_{n1} & -a_{n2} & \cdots & d_n
\end{bmatrix}
\]

\( \forall i : \sum_j L_{ij} = 0 \)

Semidefinite Positive
\( Tr(f^T L f) \geq 0, \forall f \in \mathbb{R}^n \)

The trace of \( L \) is \( \propto \) to the variability imposed by the structure of the graph (Fiedler’s Thm)

\[
Tr(f^T L f) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} (f_i - f_j)^2 = \frac{1}{2} \sum_{i \sim j} (f_i - f_j)^2
\]

Dirichlet Energies!

Harmonicity
\( Lf = f - D^{-1}Af \rightarrow f(i) = \frac{1}{d_i} \sum_j a_{ij} f(i) \)
The spectrum and eigenfunctions of $L$

### Courant-Fisher Theorem:

$$\lambda_i = \max_{x^i} \left( \min_{x \perp x^i, x \neq 0} R(x) \right)$$

### Rayleigh Quotient:

$$R(x) = \frac{x^T L x}{x^T x} \in \mathbb{R}$$

### Unconstrained Variability:

$$x^T x = \sum_{i=1}^{n} \sum_{j=1}^{n} (x_i - x_j)^2$$

- $L f_1 = 0 \rightarrow \lambda_1 = 0$
- $L f_2 = \lambda_2 f_2$ \text{ Fiedler vector}
- $\vdots$
- $L f_n = \lambda_n f_n$

$$f_2 \perp f_1, f_3 \perp \{f_1, f_2\}, \ldots \rightarrow \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_n$$

For connected graphs
Eigenfunctions of the Laplacian
Understanding the Graph Laplacian

\[ \lambda_1 = 0 \]
\[ \lambda_2 = 0.837 \]
\[ \lambda_3 = 3.389 \]
\[ \lambda_4 = 3.635 \]

\[ \lambda_5 = 4.514 \]
\[ \lambda_n = 11.547 \]

Spectrum
Spectral Theorem and Heat Kernels

Understanding the Graph Laplacian

Diffusion through Heat Kernels

\[ L = \Phi \Lambda \Phi^T, \quad \Phi = [f_1, f_2, ..., f_n], \quad \Lambda = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_n) \rightarrow L = \sum_{i=1}^{n} \lambda_i f_i f_i^T \]

Solution of heat equation and measures information flow across edges of graph with time:

\[ \frac{\partial h_t}{\partial t} = -L h_t \]

Matricial Exponential: Solution found by exponentiating Laplacian eigensystem

\[ K_t = \exp(-tL) \rightarrow \Phi \exp(-t\Lambda)\Phi^T \rightarrow \Phi \begin{bmatrix} e^{-t\lambda_1} & 0 & \cdots & 0 \\ 0 & e^{-t\lambda_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e^{-t\lambda_n} \end{bmatrix} \Phi^T \]

\[ \exp(-tL) = I - tL + \frac{t^2}{2!}L^2 - \frac{t^3}{3!}L^3 + \ldots \]

\[ t \approx 0 : K_t \approx I - tL \]

\[ t \rightarrow \infty : K_t \approx e^{-t\lambda_2}f_2 f_2^T \]
Diffusion through Heat Kernels

Motivation

- At time $t = 0$, each node has a unit of heat.
- The heat diffuses as $t \to \infty$ driven by $L$ (actually by its harmonization behavior).
- The Heat Kernel Signature (HKS) of a node is its heat trace over time.
- Heat $H_t(i,j)$ is the probability that a lazy random walk starting at node $i$ hits node $j$ at time $t$. 
Commuting Time Times
Understanding the Graph Laplacian

Commuting Time and its Embedding

\[ CT(i, j) = H(i, j) + H(j, i) \]

\[ R(i, j) = \frac{CT(i, j)}{vol(G)} \]

\[ CT(u, v) = vol(G) \sum_{i=2}^{n} \frac{1}{\lambda_i} (f_i(u) - f_i(v))^2 \]

Motivation

Time needed by a random walk to hit \( j \) (Hitting time) and return. More respectful with \( G \)'s structure than SP!

Sum of divergences between eigenfunctions pinpointed at \( u \) and \( v \) but downweighed by the eigenvalue

Smoothest eigenfunctions contribute less (btw their \( \lambda_i \) is smaller) whereas the contribution of high variance eigenfunctions is reduced by their large inverse \( \lambda_i \)
Commute Time and its Embedding

\[
CT(u, v) = \text{vol}(G) \sum_{i=2}^{n} \frac{1}{\lambda_i} (f_i(u) - f_i(v))^2
\]

\[
\Theta = \sqrt{\text{vol}(G)}\Lambda^{-1/2} \Phi^T
\]

\[
CT(u, v) = \text{vol}(G) \sum_{i=2}^{n} \frac{1}{\lambda_i} \left( \frac{g_i(u)}{\sqrt{d_u}} - \frac{g_i(v)}{\sqrt{d_v}} \right)^2
\]

\[
\Theta = \sqrt{\text{vol}(G)}\Lambda'^{-1/2} \Phi'^T \mathbf{D}^{1/2}
\]

CT matrix:

CT Embedding:

CTEmbedding in the cols of \( \Theta \)
Commute Times
Understanding the Graph Laplacian

Commute Time and its Embedding

\[ \Theta^* = \min_{\Theta \in \mathbb{R}^{n \times d} : \Theta \Theta^T = I} \text{Tr}[\Theta^T \mathcal{L} \Theta] \rightarrow \min L = \text{Tr}[\Theta^T \mathcal{L} \Theta] + \lambda_{\text{reg}} \parallel \Theta \Theta^T - I \parallel^2 \]

\[ \frac{\partial L}{\partial \Theta} = 2 \mathcal{L} \Theta + 4 \lambda_{\text{reg}} \Theta (\Theta \Theta^T - I) \]

Distance matrix:
Transductivistic Graph Rewiring
Diffusive Rewiring
Motivation and basic equations

Diffusion processes provide principled methods for linking distant nodes [Klicpera et al. 2019]

- **Improving Message Passing**: Spatial MPNNs need deep layers to leverage high-order (distant) neighborhoods.
- **Structural Noise**: Edges in real graphs are often noisy or not correlated with the distribution of nodal features.
- **Spectral principles**: Spectral GNNs allow high-order neighborhoods but are not inductive for unseen graphs.
- **GDC/DIGL**: Diffuse (PageRank/RW with restart, Heat Kernels) + sparsify + threshold as an alternative message passing.

**Parameterized**

**PPR**: \( S = \alpha (I_n + (\alpha - 1)A)^{-1} \)
- **Alpha**
  - Top-K or epsilon for thresholding edges

**Heat**: \( S = e^{t(A - I_n)} \)
- **t**
  - Top-K or epsilon for thresholding edges

**Powers to the transition matrix**

\[
S = \sum_{k=0}^{\infty} \theta_k T^k
\]
\[
S = \alpha \sum_{k=0}^{\infty} ((1 - \alpha)T)^k
\]
\[
\sum_{k=0}^{\infty} T^k = (I - T)^{-1}
\]
\[
\theta_k = \alpha (1 - \alpha)^k
\]
\[
S = \alpha (I - (1 - \alpha)T)^{-1}
\]

Row-stochastic matrix
Diffusive Rewiring
Analysis

Diffusion works as a low—pass filter of structural noise [Klicpera et al. 2019]

- Trivial choice of $T$ (random walker): $T \equiv T_{rw} = D^{-1}A$
- Interpretation of: $T^k(i,j)$ probability of hitting $j$ from $i$ in $k$-steps. Hop aggregation: $\theta_1 T + \theta_2 T^2 + \theta_3 T^3 + ...$
- $k \to \infty$: Hitting probability is proportional to degree. But more distant modes can be reached -> Structural Smoothing

$$T \equiv T_{sym} = (I + D)^{-1/2}(I + A)(I + D)^{-1/2}$$

$$S = \alpha(I - (1 - \alpha)T)^{-1}$$
Diffusive Rewiring
Analysis

Sparsification and thresholding after diffusion [Nassar et al. 2015]

- Sparsification and thresholding: \( \tilde{S} = S \times (S \geq \varepsilon) \)
- Why \( \tilde{S} \)? Limit distribution of \( S \) is somewhat sparse (some nodes maybe not visited). This is “localization”.
- Sparsification is enabled by localization! Perturbation mostly affects to highest and lowest eigenvalues.

After sparsification

\[ T_{\text{sym}}^{\tilde{S}} \equiv D_{S}^{-1/2} \tilde{S} D_{S}^{-1/2} \]

Edge magnitude

\[ S(i, j), (i \in V_a, j \in V_b) \] vs \( S(i, j), (i \in V_a, j \in V_a) \)

Final thresholding

\[ T_{S \_th} = \frac{T_{S \_zeroD}}{0.008} \]


GITHUB: https://github.com/gasteigerjo/gdc and also recently incorporated to Pytorch Geometric.
The Cheeger Constant is a separator problem

- Given a graph $G$, remove as few edges as possible to disconnect the graph into two parts of almost equal size.
- Solving this problem implies exploring the $2^{|V|}$ subsets $S \subseteq V$ of the graph.
- Each one induces a partition $S \cup \bar{S} = V$, $S \cap \bar{S} = \emptyset$.

$$h_G = \min_{S \subseteq V} h_S,$$

$$h_S = \frac{\text{cut}(S, \bar{S})}{\min(\text{vol}(S), \text{vol}(\bar{S}))}$$

$\text{cut}(S, \bar{S}) = \{|(u, v): u \in S, v \in \bar{S}\}$

Number of edges in the bottleneck

Minimal edge density in the partition

Volume of the community

However, this quantity can be spectrally bounded (and it bounds the spectra)

$$\frac{\lambda_2}{2} \leq h_G < \sqrt{2\lambda_2} \quad \text{and} \quad 2h_G \leq \lambda_2 < \frac{h_G^2}{2}$$

$\lambda_2$ is the first non-trivial eigenvalue of the normalized Laplacian of $G$.
Curvature
The Cheeger Constant

Since graphs encode manifolds, curvature (positive, negative or zero) quantifies the dispersion of geodesics (e.g. shortest paths): [Devrient and Lambiotte. 2022]

- Zero: geodesics remain parallel (e.g. grid)
- Positive: geodesics converge (e.g. clique)
- Negative: geodesics diverge (e.g. trees)

Edge curvature: [Topping el al., 2022]
- \( \#\Delta(i, j)\): Triangles based at (i,j)
- \(\#^i\mathcal{M}(i, j)\): Neighbors of i forming a 4-cycle based on (i,j) without diagonals inside.
- \(\gamma_{\text{max}}(i, j)\): Maximal number of 4-cycles based at (i,j) traversing a common node


**Curvature**

**Intuition**

**Balanced forman curvature**

\[ Ric(i, j) = 0 \text{ if } \min\{d_i, d_j\} = 1 \]

\[ Ric(i, j) = \frac{2}{d_i} + \frac{2}{d_j} - 2 + 2 \frac{|\#(i, j)|}{\max\{d_i, d_j\}} + \frac{|\#(i, j)|}{\min\{d_i, d_j\}} + \frac{(\gamma_{\max})^{-1}}{\max\{d_i, d_j\}}(|i| + |j|) \]

\[ d_0=5, \ d_1=3 \]

\[ |\#(0,1)| = 1 \text{ given by triangle } \{1, 6, 0\} \]

\[ \#^0(0,1) = \{2, 3\} \text{ without } 4,6 \text{ because triangle } \{1, 6, 0\} \]

\[ \#^1(0,1) = \{5\} \text{ without } 4,6 \text{ because triangle } \{1, 6, 0\} \]

\[ \gamma_{\max}(0,1) = 2 \text{ from the two } 4\text{-cycles passing through node } 5. \]

\[ Ric(0,1) = \frac{2}{5} + \frac{2}{3} - 2 + 2 \frac{1}{5} + \frac{1}{5} + \frac{6+10}{15} - 2 + \frac{6+5}{15} + \frac{5}{10} = -2 + \frac{22}{15} + \frac{3}{10} = -2 + \frac{44+9}{30} = -2 + \frac{51}{30} = -0.23 < 0 \]

Balanced forman curvature

\[ Ric(i, j) = 0 \text{ if } \min\{d_i, d_j\} = 1 \]

\[
Ric(i, j) = \frac{2}{d_i} + \frac{2}{d_j} - 2 + 2 \frac{|\Delta(i,j)|}{\max\{d_i, d_j\}} + \frac{|\Delta(i,j)|}{\min\{d_i, d_j\}} + \frac{(\gamma_{\max})^{-1}}{\max\{d_i, d_j\}} (|\#i| + |\#j|) 
\]

\[ d_0 = 5, \ d_1 = 2 \]

\[ Ric(0,1) = \frac{2}{5} + \frac{2}{2} - 2 + \frac{1}{5} + \frac{(1)^{-1}}{5}(1 + 0) = \frac{4+10}{10} - 2 + \frac{4+5}{10} + \frac{1}{5} = -2 + \frac{23}{10} + \frac{1}{5} = -2 + \frac{25}{10} = 2.5 > 0 \]
Balanced forman curvature

\[ Ric(i, j) = 0 \text{ if } \min\{d_i, d_j\} = 1 \]

\[ Ric(i, j) = \frac{2}{d_i} + \frac{2}{d_j} - 2 + 2 \frac{|\triangledown (i,j)|}{\max\{d_i, d_j\}} + \frac{|\triangledown (i,j)|}{\min\{d_i, d_j\}} + \frac{(\gamma_{\max})^{-1}}{\max\{d_i, d_j\}} (|\#i| + |\#j|) \]

\[ |\triangledown (0,1)| = 0 \text{ no triangle based at } (0,1) \]
\[ \#\triangledown (0,1) = \emptyset \text{ no 4-cycle based at } (0,1) \]
\[ \#\triangledown 1 (0,1) = \emptyset \text{ no 4-cycle based at } (0,1) \]
\[ \gamma_{\max} (0,1) = 0 \text{ no 4-cycle} \]

\[ Ric (0,1) = \frac{2}{4} + \frac{2}{3} - 2 + 2 \frac{0}{4} + 2 \frac{0}{3} + (0)^{-1} (0 + 0) = \frac{6 + 8}{12} - 2 + 0 + 0 = -2 + \frac{14}{12} = -2 + \frac{14}{12} = -0.83 < 0 \]

Jake Topping, et al. “Understanding over-squashing and bottlenecks on graphs via curvature”. In ICLR, 2022. [URL]
Curvature

Balanced forman curvature
• Edges with very negative curvature (> -2) create bottlenecks and thus over-squashing

Original graph

Over-squashing

Leads to

Dominant class in each community absorbs the other

Jake Topping, et al. “Understanding over-squashing and bottlenecks on graphs via curvature”. In ICLR, 2022. [URL]
Curvature
Intuition

Balanced forman curvature
• Enlarging the bottlenecks reduces over-squashing

Original graph

Relaxed
Over-squashing

Leads to

Dominant class in each community may NOT absorb the other

Jake Topping, et al. “Understanding over-squashing and bottlenecks on graphs via curvature”. In ICLR, 2022. [URL]
The SRDF ALGORITHM

**Stochastic Discrete Ricci Flow (SDRF)**

**Algorithm 1:** Stochastic Discrete Ricci Flow (SDRF)

**Input:** graph $G$, temperature $\tau > 0$, max number of iterations, optional Ric upper-bound $C^+$

**Repeat**

1. For edge $i \sim j$ with minimal Ricci curvature $\text{Ric}(i, j)$:
   - Calculate vector $x$ where $x_{kl} = \text{Ric}_{kl}(i, j) - \text{Ric}(i, j)$, the improvement to $\text{Ric}(i, j)$ from adding edge $k \sim l$ where $k \in B_1(i), l \in B_1(j)$;
   - Sample index $k, l$ with probability $\text{softmax}(\tau x)_{kl}$ and add edge $k \sim l$ to $G$.
2. Remove edge $i \sim j$ with maximal Ricci curvature $\text{Ric}(i, j)$ if $\text{Ric}(i, j) > C^+$.

**Until** convergence, or max iterations reached;

**SURGICAL REWIRING:**

**Minimal Ricci curvature:** Best candidate edge to improve.

Sample neighboring edges with probability proportional to improvement.

Remove Edge with maximal Ricci curvature

$\text{Ric}(i, j) > k > 0 \ \forall (i, j) \Rightarrow \frac{\lambda_2}{\tau} \geq h_G \geq \frac{k}{\tau}$

Jake Topping, et al. “Understanding over-squashing and bottlenecks on graphs via curvature”. In ICLR, 2022. [URL]
Diffusion works as a low—pass filter of structural noise [Klicpera et al. 2019]
SDRF is quirurgical on behalf of a structural test for each edge [Topping et al., 2022]

- The Cheeger constant of SGD/DIGL is controlled by that of SDRF:
  \[ h_{S,\alpha} \leq \frac{(1-\alpha)}{\alpha} \frac{d_{\text{avg}}(S)}{d_{\text{min}}(S)} h_S \]
- SDRF preserves more the structure than SGD/DIGL (which may remove the cut)

After SDRF

Removes intra & Adds inter

Degree Distributions

Jake Topping, et al. “Understanding over-squashing and bottlenecks on graphs via curvature”. In ICLR, 2022. URL.
Curvature vs Diffusive Rewiring

Analysis

Diffusion works better in homophilic graphs [Klicpera et al. 2019] Needs parameters $\alpha$ (or $t$) and $\epsilon$

SDRF works better in heterophilic graphs [Topping et al., 2022] Needs parameters $\tau$ and $C^+$

<table>
<thead>
<tr>
<th>$\mathcal{H}(G)$</th>
<th>Cornell</th>
<th>Texas</th>
<th>Wisconsin</th>
<th>Chameleon</th>
<th>Squirrel</th>
<th>Actor</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0.11</td>
<td>0.06</td>
<td>0.16</td>
<td>0.25</td>
<td>0.22</td>
<td>0.24</td>
<td>0.83</td>
<td>0.71</td>
<td>0.79</td>
</tr>
<tr>
<td>Undirected</td>
<td>53.20 ± 0.53</td>
<td>63.38 ± 0.87</td>
<td>51.37 ± 1.15</td>
<td>42.02 ± 0.30</td>
<td>35.53 ± 0.78</td>
<td>21.45 ± 0.47</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>+FA</td>
<td>58.29 ± 0.49</td>
<td>64.82 ± 0.29</td>
<td>55.48 ± 0.62</td>
<td>42.67 ± 0.17</td>
<td>36.86 ± 0.44</td>
<td>24.14 ± 0.43</td>
<td>81.65 ± 0.18</td>
<td>70.47 ± 0.18</td>
<td>79.48 ± 0.12</td>
</tr>
<tr>
<td>DIGL (PPR)</td>
<td>58.26 ± 0.50</td>
<td>62.03 ± 0.43</td>
<td>49.53 ± 0.27</td>
<td>42.02 ± 0.13</td>
<td>33.22 ± 0.14</td>
<td>24.77 ± 0.32</td>
<td>83.21 ± 0.27</td>
<td>73.29 ± 0.17</td>
<td>78.84 ± 0.08</td>
</tr>
<tr>
<td>DIGL + Undirected</td>
<td>59.54 ± 0.64</td>
<td>63.54 ± 0.38</td>
<td>52.23 ± 0.54</td>
<td>42.68 ± 0.12</td>
<td>32.48 ± 0.23</td>
<td>25.45 ± 0.30</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SDRF</td>
<td>54.60 ± 0.39</td>
<td>64.46 ± 0.38</td>
<td>55.51 ± 0.27</td>
<td>42.73 ± 0.15</td>
<td>37.05 ± 0.17</td>
<td>28.42 ± 0.75</td>
<td>82.76 ± 0.23</td>
<td>72.58 ± 0.20</td>
<td>79.10 ± 0.11</td>
</tr>
<tr>
<td>SDRF + Undirected</td>
<td>57.54 ± 0.34</td>
<td>70.35 ± 0.60</td>
<td>61.55 ± 0.86</td>
<td>44.46 ± 0.17</td>
<td>37.67 ± 0.23</td>
<td>28.35 ± 0.06</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>


Inductive Graph Rewiring
The Lovász Bound
Motivation and basic equations

The Lovász bound explains the expressiveness of commute times [Lovász, 1993]

\[
\frac{|CT(u, v)|}{\text{vol}(G)} - \left( \frac{1}{d_u} + \frac{1}{d_v} \right) \leq \frac{2}{\lambda_2 d_{\min}}
\]

Deviation from Local resistance: The global effective resistance should be far from its local estimation to be informative.

Inverse of the bottleneck: High spectral gaps induce uninformative effective resistances. (Link to Cirvature)

High probability of getting lost in (some) large graphs [von Luxburg et al., 2014]

Some facts:

- Effective resistances are also given by the Laplacian’s pseudoinverse or Green’s function
  \[
  R(u, v) = (e_u - e_v)^T L^+ (e_u - e_v), \quad L^+ = \sum_{i \geq 2} \lambda_i^{-1} f_i f_i^T
  \]

- Effective resistances are upper bounded by shortest paths
  (and they are by far more informative about the role of the Edge (u,v) in the graph since all paths are considered)
The Lovász Bound
Impact of the bound

Consider two SBMs with small and large gap respectively:

Map the Fiedler vector as node attributes!

Bottleneck of $G$ is $0.027295784924703657$

Bottleneck of $H$ is $0.7588701310820082$


The spectral gap (i.e. the Dirichlet energy of the Fiedler vector) controls the variance of $f_2$ and consequently the scatter in the latent space:

Latent spaces: Nodes and KDEs

Effective resistances (when informative) $R(u,v)$ reveal the impact of each Edge $(u,v)$ in the topology of the Graph. Therefore, sampling edges with a probability proportional to the effective resistance results in a sparse version of the graph. [Spielman and Srivastava, 2011]

$$O\left(\frac{n \log n}{\epsilon}\right)$$ samples suffice to satisfy

$$\forall x \in \mathbb{R}^n: (1 - \epsilon)x^T L_G x \leq x^T L_{G'} x \leq (1 + \epsilon)x^T L_G x$$

KDE of probabilities

The Lovász Bound

Commute Times embeddings rely on **down-scaled versions** of the eigenvectors \( F \) and the scale factor is the corresponding eigenvalue.

\[
\text{CT Embedding } \rightarrow CT_{uv} = \|z_u - z_v\|^2 \quad \text{comes from } \quad Z = \sqrt{\text{vol}(G)} \Lambda^{-1/2} F^T
\]

\[
z_u = \sqrt{\text{vol}(G)} \begin{pmatrix} 0 & f_2(u) & f_3(u) & \cdots & f_n(u) \\ \sqrt{\lambda_2} & \sqrt{\lambda_3} & \cdots & \sqrt{\lambda_n} \end{pmatrix}^T
\]

and consequently \( Z^T \) gets the scaled non-trivial eigenvectors.

\[
Z^T \quad |Z^T - F| \quad \text{KDE of } |Z^T - F|
\]
# Directional Graph Networks

## Pre-computed steps $O(kE)$

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>Input graph</td>
</tr>
<tr>
<td>(b)</td>
<td>Compute the first $k$ eigenvectors</td>
</tr>
<tr>
<td>(c)</td>
<td>Compute the gradient</td>
</tr>
<tr>
<td>(d)</td>
<td>Create the aggregation matrices $B$</td>
</tr>
</tbody>
</table>

- The eigenvectors $\phi$ of $L$ are computed and sorted such that $\phi_1$ has the lowest non-zero eigenvalue and $\phi_k$ has the $k$-th lowest.
- We compute the $k$ first eigenvectors with a complexity of $O(kE)$.
- The gradient of $\phi$ is a function of the edges (a matrix) such that $\nabla \phi_i = \phi_i - \phi_j$ if the nodes $i, j$ are connected, or $\nabla \phi_i = 0$ otherwise.
- If the graph has a known direction, it can be encoded as field $F$.

## Graph neural network steps $O(kE + kN)$

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>Input graph</td>
</tr>
<tr>
<td>(f)</td>
<td>Feature aggregation</td>
</tr>
<tr>
<td>(e)</td>
<td>MLP</td>
</tr>
</tbody>
</table>

- A graph with the node features is given, $X^{(0)}$ is the feature matrix of the graph at the 0-th GNN layer, of size $N \times n_0$.
- The aggregation matrices $B^{\text{dir}}_d, B^{\text{ate}}_d$ are taken from the pre-computed steps.
- The aggregation matrices $B^{\text{dir}}_d, B^{\text{ate}}_d$ are used to aggregate the features $X^{(0)}$ via the matrix product $BX$. For $B_{\text{ate}}$, we take the absolute value due to the sign ambiguity of $\phi$.
- $Y^{(0)}$ is the column-concatenation of all directional and a directional aggregations.
- The complexity is $O(kE)$, or $O(E)$ if the aggregations are parallelized.

## Diagram

[Diagram showing the process of directional graph networks]

---

**References**

**CT-Layer**

Why commute times for rewiring? Quick Recap!

\[ CT_{uv} \propto R_{uv} = H_{uv} + H_{vu} \]

→ Expected time to from \( u \) to \( v \) and come back to \( u \)

**CT Embedding** → \[ CT_{uv} = \| z_u - z_v \|_2^2 \]

→ Node embedding which pairwise Euclidean distance is \( CT_{uv} \)

**Direct relationship with**

- **Eigenvectors**
  \[ R_{uv} = \frac{CT_{uv}}{\text{vol}(G)} = \sum_{i=2}^{n} \frac{1}{\lambda_i} (f_i(u) - f_i(v))^2 \]

- **Dirichlet Energies**
  \[ E_G(x) = x^T L_G x = \sum_{(u,v) \in E} (x_u - x_v)^2 = Tr[X^T L_G X] \]

- **Expanders and Sparsifiers**
  \[ \forall x \in \mathbb{R}^n: (1 - \epsilon) x^T L_G x \leq x^T L_G^* x \leq (1 + \epsilon) x^T L_G x \]

- **Cheeger Constant**
  \[ h_G = \min_{S \subseteq V} h_S, h_S = \frac{\{(u,v) : u \in S, v \in \bar{S}\}}{\text{min}(\text{vol}(S), \text{vol}(\bar{S}))} \]

- **Curvature**
  \[ \kappa_{uv} := \frac{2(p_u + p_v)}{R_{uv}} \]

---

**Spectral computation**

\[ CT_{uv} = \sum_{i=2}^{n} \frac{1}{\lambda_i} (f_i(u) - f_i(v))^2 \]

\[ Z = \sqrt{\text{vol}(G)} A^{-1/2} F^T \text{ given } L = FA F^T \]

or

\[ R_{uv} = (e_u - e_v) L^+ (e_u - e_v) \]

\[ L^+ = \sum_{i=2}^{n} \frac{1}{\lambda_i} f_i f_i^T \]

**Optimization problem**

\[ Z = \arg \min_{s.t. Z^T Z = 1} Tr[Z^T L_G Z] \]

\[ CT_{uv} = \| z_u - z_v \|_2^2 \]
CT-Layer
From Spectral CT to CT-Layer

\[ Z = \sqrt{\text{vol}(G)} \Lambda^{-1/2} F^T \]
\[ Z = \arg \min_{s.t. \ Z^T Z = I} \frac{\text{Tr}[Z^T L G Z]}{\text{Tr}[Z^T D G Z]} \]

\[ L_{CT} = \frac{\text{Tr}[Z^T L Z]}{\text{Tr}[Z^T D Z]} + \left\| \frac{Z^T Z}{\|Z^T Z\|_F} - I_N \right\|_F \]

Use Effective Resistances matrix (commute times) as the input adjacency matrix for new layers.

CT-layer can be added as the first layer or as the # desired layer.

CT-Layer
From Spectral CT to CT-Layer

\[ L_{CT} = \frac{Tr[Z^T L Z]}{Tr[Z^T D Z]} + \frac{Z^T Z}{\|Z^T Z\|_F} - I_n \]_F

CT as diffusion (A)
(DiffWire – actual rewiring – previous slide)

CT as edge feat
(Affinity-Aware approach)

CTE as node feat
(preliminary work in both papers)

Simplified version of the previous slide

CT-Layer
From Spectral CT to CT-Layer

\[ L_{CT} = \frac{Tr[Z^T L Z]}{Tr[Z^T D Z]} + \left\| \frac{Z^T Z}{\|Z^T Z\|_F} - I_N \right\|_F \]

# Pooling for CT embedding
num_of_centers1 = k_centers # k1 #order of number of nodes
self.pool1 = Linear(hidden_channels, num_of_centers1)

# CT REWRITING
s = self.pool1(x)

https://github.com/AdrianArnaiz/DiffWire/blob/main/layers/CT_layer.py
CT-Layer
From Spectral CT to CT-Layer

\[
\begin{align*}
L_{CT} &= \frac{Tr[Z^T LZ]}{Tr[Z^T DZ]} + \left\| \frac{Z^T Z}{\|Z^T Z\|_F} - I_N \right\|_F \\
T^C &\in \mathbb{R}^{n \times n} = \frac{\text{cdist}(Z)}{\text{vol}(G)} \odot A \\
T^C &= \text{CT} \\
Z &\in \mathbb{R}^{n \times O(n)} \xrightarrow{\text{MLP \, tanh}}
\]

### Code Example
```python
# Calculate CT_dist as cdist(s,s)/vol(G)
CT_dist = torch.cdist(s, s)  # [b, N, k], [b, N, k] -> [2b, N, N]

# Calculate degree d.flat and degree matrix d
d_flat = torch.einsum('ijk\rightarrow ij', adj)  # torch.Size([b, N])
d = _rank3_diag(d_flat) + EPS  # torch.Size([b, N, N])

# Calculate Vol (volumes): one per graph
vol = _rank3_trace(d)  # torch.Size([b])

# Calculate out_adj as CT_dist*(N-1)/vol(G)
N = adj.size(0)
CT_dist = (CT_dist / vol.unsqueeze(1).unsqueeze(1))

# Mask with adjacency
adj = CT_dist*adj
```

https://github.com/AdrianArnaiz/DiffWire/blob/main/layers/CT_layer.py
CT-Layer
From Spectral CT to CT-Layer

\[ L_{CT} = \frac{\text{Tr}[Z^TLZ]}{\text{Tr}[Z^TDZ]} + \frac{Z^TZ}{\|Z^TZ\|_F} - I_N \|_{F} \]

**From Spectral CT to CT-Layer**

**MLP**
\[ Z \in \mathbb{R}^{n \times O(n)} \]

\[ T^{CT} \in \mathbb{R}^{n \times n} = \frac{\text{cdist}(Z)}{\text{vol}(G)} \odot A \]

\[ T^{CT} \]

\[ X \rightarrow \text{MLP} \rightarrow \text{tanh} \rightarrow Z \in \mathbb{R}^{n \times O(n)} \rightarrow T^{CT} \in \mathbb{R}^{n \times n} = \frac{\text{cdist}(Z)}{\text{vol}(G)} \odot A \rightarrow T^{CT} \]

28  # Losses
29  ## Loss cut
30  ### Calculate Laplacian \( L = D - A \)
31  \( L = d - \text{adj} \)
32  ### Calculate loss num as \( \text{Tr}[S(T \cdot L \cdot S)] \)
33  num = torch.matmul(torch.matmul(s.transpose(1, 2), L), s)
34  CT_num = _rank3_trace(num) # without torch.Size([B]) one sum over each graph
35  ### Calculate CT_den
36  CT_den = _rank3_trace(
37      torch.matmul(torch.matmul(s.transpose(1, 2), d), s)) + eps
38  ### Calculate loss cut
39  CT_loss = CT_num / CT_den
40  CT_loss = torch.mean(CT_loss) # Mean over batch!

https://github.com/AdrianArnaiz/DiffWire/blob/main/layers/CT_layer.py

CT-Layer
From Spectral CT to CT-Layer

\[ L_{CT} = \frac{Tr[Z^T L Z]}{Tr[Z^T D Z]} + \left\| \frac{Z^T Z}{\|Z^T Z\|_F} - I_N \right\|_F \]

\[ T^{CT} = \frac{\text{cdist}(Z)}{\text{vol}(G)} \odot A \]

MLP tanh

\[ Z \in \mathbb{R}^{n \times O(n)} \rightarrow T^{CT} \in \mathbb{R}^{n \times n} \]

\[ X \quad \text{A} \]

https://github.com/AdrianArnaiz/DiffWire/blob/main/layers/CT_layer.py

CT-Layer
From Spectral CT to CT-Layer

\[ X \xrightarrow{\text{MLP tanh}} Z \in \mathbb{R}^{n\times O(n)} \]

\[ L_{CT} = \frac{Tr[Z^T LZ]}{Tr[Z^T D Z]} + \|Z^T Z\|_F - I_N \|_F \]

CT-Layer
From Spectral CT to CT-Layer

\[ Z \in \mathbb{R}^{n \times O(n)} \]
\[ \text{cdist}(Z) / \text{vol}(G) \]
\[ T^{CT} \in \mathbb{R}^{n \times n} \]

\[ L_{CT} = \frac{Tr[Z^T LZ]}{Tr[Z^T D Z]} + \left\| \frac{Z^T Z}{\|Z^T Z\|_F} - I_N \right\|_F \]

CT-Layer
From Spectral CT to CT-Layer

\[
X \xrightarrow{\text{MLP \, tanh}} Z \in \mathbb{R}^{n \times O(n)} \xrightarrow{\text{cdist}(Z) \middle/ \text{vol}(G)} A \xrightarrow{\text{A} \circ \text{CT} \in \mathbb{R}^{n \times n}} \]

\[
L_{CT} = \frac{\text{Tr}[Z^T LZ]}{\text{Tr}[Z^T D Z]} + \left\| \frac{Z^T Z}{\|Z^T Z\|_F} - I_N \right\|_F
\]

Graph from COLLAB Test Set
CT-Layer
Experiments on Graph Classification

<table>
<thead>
<tr>
<th></th>
<th>MinCutPool</th>
<th>$k$-NN</th>
<th>DIGL</th>
<th>SDRF</th>
<th>CT-LAYER</th>
</tr>
</thead>
<tbody>
<tr>
<td>REDDIT-B*</td>
<td>66.53±4.4</td>
<td>64.40±3.8</td>
<td>76.02±4.3</td>
<td>65.3±7.7</td>
<td><strong>78.45±4.5</strong></td>
</tr>
<tr>
<td>IMDB-B*</td>
<td>60.75±7.0</td>
<td>55.20±4.3</td>
<td>59.35±7.7</td>
<td>59.2±6.9</td>
<td><strong>69.84±4.6</strong></td>
</tr>
<tr>
<td>COLLAB*</td>
<td>58.00±6.2</td>
<td>58.33±11</td>
<td>57.51±5.9</td>
<td>56.60±10</td>
<td><strong>60.87±2.4</strong></td>
</tr>
<tr>
<td>MUTAG</td>
<td>84.21±6.3</td>
<td><strong>87.58±4.1</strong></td>
<td>85.00±5.6</td>
<td>82.4±6.8</td>
<td><strong>87.58±4.4</strong></td>
</tr>
<tr>
<td>PROTEINS</td>
<td>74.84±2.3</td>
<td><strong>76.76±2.5</strong></td>
<td>74.49±2.8</td>
<td>74.4±2.7</td>
<td><strong>75.38±2.9</strong></td>
</tr>
<tr>
<td>SBM*</td>
<td>53.00±9.9</td>
<td>50.00±0.0</td>
<td>56.93±12</td>
<td>54.1±7.1</td>
<td>81.40±11</td>
</tr>
<tr>
<td>Erdős-Rényi*</td>
<td>81.86±6.2</td>
<td>63.40±3.9</td>
<td><strong>81.93±6.3</strong></td>
<td>73.6±9.1</td>
<td>79.06±9.8</td>
</tr>
</tbody>
</table>

**EXPRESSIVENESS**
More sparse Graph Readouts $\rightarrow$ greater ability to detect differences between graphs
DE or PE provides strictly more expressive power than $1$-WL test [Li, P. et al. 2020] [Velingker, A. et al. 2022]

CT-Layer
Implications in Cheeger constant

CT prioritizes edges in the bottleneck while it sparsifies the communities

\[ h_G = \min_{S \subseteq V} h_S, \quad h_S = \frac{\text{# edges in the bottleneck}}{\min(\text{vol}(S), \text{vol}(\overline{S}))} \]

Giving priority to the edges in the bottleneck maintains this
Community sparsification minimizes this

In the rewired graph \( G' \): bottleneck is wider \( \rightarrow \) resistances are lower in \( G' \)

[Allev et al., 2018]

\[ R_{diam} := \max_{u,v} R_{uv} \]

Edges in the bottleneck

\[ h_G \leq \frac{\alpha^\epsilon}{\sqrt{R_{diam}} \cdot \epsilon^{\frac{1}{2}}} \text{vol}(S)^{\epsilon - \frac{1}{2}} \]

Prioritizing edges in the bottleneck maintains upper bound (at least)

Rewiring using CT dist
Community sparsification
Preserve bottleneck
Preserve \( R_{diam} \) in \( G' \)
No reduction of \( h_G \) upper bound

Alev, VL., et al. “Graph Clustering using Effective Resistance”. In ITCS, 2018.
CT-Layer
Relationship with Curvature


\[ R_{uv} = T_{uv}^{CT} \]

[Devriendt. et al., 2022]

Node Curvature

\[ p_u := 1 - \frac{1}{2} \sum_{v \in N(u)} R_{uv} \]

Edge Curvature

\[ \kappa_{uv} := \frac{2(p_u + p_v)}{R_{uv}} \]

CT-Layer as differentiable curvature
CT-Layer
Relationship with Curvature

CT-Layer

Node Classification. CT-Diffusions vs CT as Positional Encoding.

- **CTE as structural feature (PE) reinforces performance in homophily tasks**

- **CT Distance for diffusion helps in heterophilic tasks**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>GCN (baseline)</th>
<th>model 1: ( X \parallel Z )</th>
<th>model 2: ( A = T^{CT} )</th>
<th>Homophily</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>82.01 ± 0.8</td>
<td>83.66 ± 0.6</td>
<td>67.96 ± 0.8</td>
<td>81.0%</td>
</tr>
<tr>
<td>Pubmed</td>
<td>81.61 ± 0.3</td>
<td>86.07 ± 0.1</td>
<td>68.19 ± 0.7</td>
<td>80.0%</td>
</tr>
<tr>
<td>Citeser</td>
<td>70.81 ± 0.5</td>
<td>72.26 ± 0.5</td>
<td>66.71 ± 0.6</td>
<td>73.6%</td>
</tr>
<tr>
<td>Cornell</td>
<td>59.19 ± 3.5</td>
<td>58.02 ± 3.7</td>
<td>69.04 ± 2.2</td>
<td>30.5%</td>
</tr>
<tr>
<td>Actor</td>
<td>29.59 ± 0.4</td>
<td>29.35 ± 0.4</td>
<td>31.98 ± 0.3</td>
<td>21.9%</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>68.05 ± 6.2</td>
<td>69.25 ± 5.1</td>
<td>79.05 ± 2.1</td>
<td>19.6%</td>
</tr>
</tbody>
</table>

Homophily vs Accuracy

CT-Layer

Node Classification. CT-Diffusions vs CT as Positional Encoding.

- **CTE** as structural feature (PE) reinforces performance in **homophily** tasks
- **CT Distance** for diffusion helps in **heterophilic** tasks

<table>
<thead>
<tr>
<th>Dataset</th>
<th>GCN (baseline)</th>
<th>model 1: ( X \parallel Z )</th>
<th>model 2: ( A = T^{CT} )</th>
<th>Homophily</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>82.01±0.8</td>
<td>83.66±0.6</td>
<td>67.96±0.8</td>
<td>81.0%</td>
</tr>
<tr>
<td>Pubmed</td>
<td>81.61±0.3</td>
<td>86.07±0.1</td>
<td>68.19±0.7</td>
<td>80.0%</td>
</tr>
<tr>
<td>Citeseer</td>
<td>70.81±0.5</td>
<td>72.26±0.5</td>
<td>66.71±0.6</td>
<td>73.6%</td>
</tr>
<tr>
<td>Cornell</td>
<td>59.19±3.5</td>
<td>58.02±3.7</td>
<td>69.04±2.2</td>
<td>30.5%</td>
</tr>
<tr>
<td>Actor</td>
<td>29.59±0.4</td>
<td>29.35±0.4</td>
<td>31.98±0.3</td>
<td>21.9%</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>68.05±6.2</td>
<td>69.25±5.1</td>
<td>79.05±2.1</td>
<td>19.6%</td>
</tr>
</tbody>
</table>

**Homophily vs Accuracy**

---


**Goal**: Optimize bottleneck width

\[ \lambda_2 := \text{spectral gap or bottleneck size} \]

- Search \( \tilde{A} \) as similar as \( A \) but **minimizing bottleneck size**
  - Spectral derivatives

\[
L_{\text{Fiedler}} = \| \tilde{A} - A \|_F + \alpha (\lambda_2)^2 \\
\nabla_{\tilde{A}} \lambda_2 := Tr [(\nabla L \lambda_2)^T \nabla_{\tilde{A}} \tilde{L}] = \text{diag}(f_2 f_2^T) 11^T - f_2 f_2^T
\]

- \( f_2 \in \mathbb{R}^n \) := Fiedler vector
  - \( f_2 \): Node membership to each of the 2 clusters
  - \( \lambda_2 \): Eigenvalue of \( f_2 \) (Dirichlet energies of \( f_2 \))

- Main problem: \( \lambda_2 \) and \( f_2 \) are **usually spectrally computed**

---

Kang, J. and Tong, H. “N2n: Network derivative mining”. In CIKM, 2019.
**GAP-Layer**

Gap-Layer: Approximating the Fiedler vector

\[ L_{cut} = \frac{\text{Tr}[S^T L S]}{\text{Tr}[S^T D S]} + \left\| \frac{S^T S}{\|S^T S\|_F} - \frac{I_N}{\sqrt{2}} \right\|_F \]

\[ L_{fiedler} = \|\tilde{A} - A\|_F + \alpha(\lambda_2)^2 \]

\[ \nabla_{\tilde{A}} \lambda_2 = [2(\tilde{A} - A) + (\text{diag}(f_2^T f_2^T))11^T - f_2^T f_2^T] \times \lambda_2 \]

**How does GAP-Layer learn \( f_2 \)?**

\[ f_2(S) = \begin{cases} +1/\sqrt{n} & \text{if } u \text{ belongs to cluster #1} \\ -1/\sqrt{n} & \text{if } u \text{ belongs to cluster #2} \end{cases} \]

[Hoang et al., 2020]

**How does GAP-Layer learn \( \lambda_2 \)?**

\[ \lambda_2 = \mathcal{E}_G (f_2) = f_2^T L_G f_2 \]

Dirichlet energies of the approximated \( f_2 \)

---


**GAP-Layer Experiments**

### Original vs. CT-Layer vs. GAP-Layer

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MinCutPool</th>
<th>k-NN</th>
<th>DIGL</th>
<th>SDRF</th>
<th>CT-LAYER</th>
<th>GAP-LAYER (R)</th>
<th>GAP-LAYER (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>REDDIT-B†</td>
<td>66.53±4.4</td>
<td>64.40±3.8</td>
<td>76.02±4.3</td>
<td>65.3±7.7</td>
<td><strong>78.45±4.5</strong></td>
<td><strong>77.63±4.9</strong></td>
<td>76.00±5.3</td>
</tr>
<tr>
<td>IMDB-B†</td>
<td>60.75±7.0</td>
<td>55.20±4.3</td>
<td>59.35±7.7</td>
<td>59.2±6.9</td>
<td><strong>69.84±4.6</strong></td>
<td><strong>69.93±3.3</strong></td>
<td>68.80±3.1</td>
</tr>
<tr>
<td>COLLAB†</td>
<td>58.00±6.2</td>
<td>58.33±11</td>
<td>57.51±5.9</td>
<td>56.60±10</td>
<td><strong>69.87±2.4</strong></td>
<td>64.47±4.0</td>
<td><strong>65.89±4.9</strong></td>
</tr>
<tr>
<td>MUTAG</td>
<td>84.21±6.3</td>
<td><strong>87.58±4.1</strong></td>
<td>85.00±5.6</td>
<td>82.4±6.8</td>
<td><strong>87.58±4.4</strong></td>
<td><strong>86.90±4.0</strong></td>
<td><strong>86.90±4.0</strong></td>
</tr>
<tr>
<td>PROTEINS</td>
<td>74.84±2.3</td>
<td><strong>76.76±2.5</strong></td>
<td>74.49±2.8</td>
<td>74.4±2.7</td>
<td><strong>75.38±2.9</strong></td>
<td>75.03±3.0</td>
<td><strong>75.34±2.1</strong></td>
</tr>
<tr>
<td>SBM*</td>
<td>53.00±9.9</td>
<td>50.00±0.0</td>
<td>56.93±12</td>
<td>54.1±7.1</td>
<td><strong>81.40±11</strong></td>
<td><strong>90.80±7.0</strong></td>
<td><strong>92.26±2.9</strong></td>
</tr>
<tr>
<td>Erdős-Rényi*</td>
<td>81.86±6.2</td>
<td>63.40±3.9</td>
<td><strong>81.93±6.3</strong></td>
<td>73.6±9.1</td>
<td>79.06±9.8</td>
<td>79.26±10</td>
<td><strong>82.26±3.2</strong></td>
</tr>
</tbody>
</table>

Future work

Rewiring
- Dynamic Rewiring wrt structure, homophily-heterophily and utility
  - Reduce or enforce over-squashing when needed (merge only util information)
- Rewiring with Interpretability

DiffWire
- Use of learned CT for different objectives
- Code to sparse $\rightarrow$ efficient computation
- Code to PyG $\rightarrow$ Easy use (even more)
Graph Fairness

Algorithmic Fairness with Graph Rewiring

Algorithmic Fairness
ML for Critical Decision Making

✓ Privacy
✓ Transparency
✓ Accountability
✓ Reliability
✓ Autonomy
✓ Fairness

Social Biased decisions leads to

<table>
<thead>
<tr>
<th>INDIVIDUAL HARDS</th>
<th>COLLECTIVE SOCIAL HARDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILLEGAL DISCRIMINATION</td>
<td>UNFAIR PRACTICES</td>
</tr>
<tr>
<td>HIRING</td>
<td>LOSS OF OPPORTUNITY</td>
</tr>
<tr>
<td>EMPLOYMENT</td>
<td></td>
</tr>
<tr>
<td>INSURANCE &amp; SOCIAL BENEFITS</td>
<td></td>
</tr>
<tr>
<td>HOUSING</td>
<td></td>
</tr>
<tr>
<td>EDUCATION</td>
<td></td>
</tr>
<tr>
<td>CREDIT</td>
<td>ECONOMIC LOSS</td>
</tr>
<tr>
<td>DIFFERENTIAL PRICES OF GOODS</td>
<td></td>
</tr>
<tr>
<td>LOSS OF LIBERTY</td>
<td></td>
</tr>
<tr>
<td>INCREASED SURVEILLANCE</td>
<td></td>
</tr>
<tr>
<td>STEREOTYPE REINFORCEMENT</td>
<td></td>
</tr>
<tr>
<td>DIGNATORY HARMS</td>
<td>SOCIAL STIGMATIZATION</td>
</tr>
</tbody>
</table>

Buolamwini, J., et al. “Gender shades: Intersectional accuracy disparities in commercial gender classification”. In FAccT, 2021
Algorithmic Fairness

Independence on the Protected Attributes

Ensure that the outputs of a model DO NOT depend on sensitive attributes

\[ F(X) = R, \quad S \in X \rightarrow R \perp S \]

Group Fairness

Groups (defined by sensitive attributes) are treated equally

| \( P(R|S) \) | \( P(R|Y, S) \) | \( P(Y|R, S) \) |
|----------------|----------------|----------------|
| **Independence** | **Separation** | **Sufficiency** |
| \( R \perp S \) | \( R \perp S \mid Y \) | \( S \perp Y \mid R \) |

Demographic parity

\[ P(R=1|S=a) = P(R=1|S=b) \]

Positive Predicted Ratio: Equal acceptance rate

\[ TPR - FPR \]

Equalized odds

\[ P(R=1|Y=i, S=a) = P(R=1|Y=i, S=b), \quad i \in 0,1 \]

Predictive Parity

\[ P(Y=1|R=1, S=a) = P(Y=1|R=1, S=b) \]

Equal error rates

\[ PPV - NPV \]

Equal success rate

Individual Fairness

Treat similar individuals in a similar way

Our Dataset: \( D = \{(x_i, y_i)\}_i^N \)

Distance between \( x_i \) pairs: \( k: V \times V \rightarrow R \).

Mapping from \( x_i \) to outcomes probability distribution \( M: V \rightarrow \alpha S \)

Distance between distributions of outputs \( D \)

\[ D(M(x), M(y)) = k(x, y) \]


Why Graph Fairness?
The Graph Structure: a New Biased Element

Topology of the graph (A) can be biased \(\rightarrow\) correlated with sensitive attributes

- **Over-representing homophilic edges** (social stratification, fraudulent links, social homophily [McPherson, 2001])
- **Missing heterophilic edges** that would have been present in more fair settings

Friendship among students in a Dutch School [Masrour, 2020]

**Why Graph Fairness?**

Consequences on the Real world

### Decisions on the nodes

- Protected attributes
- Labels

#### Fair Topology

$$X, S$$

#### Biased Topology

$$X, S$$

* Also applies to community detection and Link prediction

### Recommendations

- Biased recommendations
  - (Products, jobs, content…)

### Influence Maximization

- Echo Chambers and Filter bubbles

---

Hampson, M. “Smart Algorithm Bursts Social Networks’ ‘Filter Bubbles’”. 2011. [Link](https://example.com)

Wang, S., et al. “Graph learning based recommender systems”. In IJCAI, 2021
# Graph Fairness

## Causes

<table>
<thead>
<tr>
<th>Structure Type</th>
<th>Unbiased Attributes</th>
<th>Biased Attributes [Y-S]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unbiased</td>
<td><img src="image1" alt="Graph" /></td>
<td><img src="image2" alt="Graph" /></td>
</tr>
<tr>
<td>Structure</td>
<td><img src="image3" alt="Graph" /></td>
<td><img src="image4" alt="Graph" /></td>
</tr>
<tr>
<td>Biased Structure [A-S]</td>
<td><img src="image5" alt="Graph" /></td>
<td><img src="image6" alt="Graph" /></td>
</tr>
</tbody>
</table>

### Unbiased Attributes
- **Structure correlates with the protected attribute**
  - $A - S$
- **Graph generation process is Biased**
  - People built relationships in correlation only with $A$
  - Model will have **bad accuracy and biased decision**
  - Assortativity in protected attribute

### Biased Attributes [Y-S]
- **Label correlates with the protected attribute**
  - $Y - S$
- **Individuals are biased, their relationships are not**
- **Both correlate with the protected attribute**
  - $S - A - Y$
- **Society is Biased**
  - People built relationships in correlation with $Y$ and/or $A$
  - Model will have **good accuracy but biased decision**
  - Assortativity in protected attribute ($S-A$) and label homophily ($A-Y$)

---

**S**: Sensitive attribute  
**A**: Adjacency, i.e. Matrix Structure  
**Y**: Node Label
## Dimensions of taxonomy

Perspectives to analyze graph fairness

<table>
<thead>
<tr>
<th>Causes</th>
<th>Fairness definitions</th>
<th>Tasks</th>
<th>Techniques</th>
</tr>
</thead>
</table>
| ▪ **A-S** correlation | ▪ **Node-level decision**  
  - Group  
  - Individual  
  - ...  
  ▪ **Structure segregation**  
  - Group  
  - Individual  
  - ... | ▪ Topology analysis  
  ▪ Representation Learning  
  ▪ Classification/Regression  
  ▪ Link prediction  
  ▪ Community detection  
  ▪ Application specific  
  ▪ Recommender systems  
  ▪ Influence maximization  
  ▪ Ranking | ▪ Constrained optimization  
  ▪ Adversarial/orthogonal  
  ▪ Rebalancing  
  ▪ **Graph Rewiring** |
| ▪ **Y-S** correlation | | | |
| ▪ **Y-A** correlation  
  (homophily) | | | |
Graph Fairness Definitions

Definitions and Metrics

Graph Fairness

Node Fairness

- Focused on node-level decisions
- Group
  - Node clf
  - Influence maximization
  - Embedding
- Individual
  - Node clf
  - Degree
  - Counterfactual
  - Embedding
  - Ranking or distances

- Similar* nodes should lead to similar predictions.
  *Similarity can be n be given based on X, A...

- Node embedding \( \perp S \)
- Node prediction \( \perp S \)

Structural Fairness

- Focused on structure segregation
- Group
  - Clustering
  - Link pred
  - Original Structure
- Individual
  - Clustering
  - Link pred
  - Original Structure

- Structure \( \perp S \)
- No Assortativity in S
- No communities based on S
- Link prediction \( \perp S \)

\[
P((u, v) \in E) = d(x_u, x_v)
\]

or they should have highly overlapped neighboring node sets

Similar nodes should be connected.
Graph Fairness Definitions
Definitions and metrics from a Pipeline Point of View

- **Model**
  - **Embeddings**
    - **Group**
      - Distributional Bias
      - Representation Bias (AUC predicting A)
    - **Individual**
      - Pair Distance $\text{Tr}(Y^T L_s Y)$
      - Ranking similarity
  - **Clustering**
    - **Group**
      - Balance (SC)
      - ModRed
    - **Individual**
      - SC
  - **Topology**
    - **Group**
      - Assortativity
      - Attribute Homophily
  - **Link Prediction**
    - **Group**
      - Dyadic Fairness
      - DI / EOd / EOp
    - **Individual**
      - $\text{Tr}(A^3 \phi^T L_k \phi A)$

- **Classification**
  - **Group**
    - DI/EOD/Eop
    - Counterfactual (% change $f(X)$ when change A)
  - **Individual**
    - Consistency
    - Degree
Graph Fairness Definitions
Definitions and metrics from a Pipeline Point of View

<table>
<thead>
<tr>
<th>Topology</th>
<th>Link Prediction</th>
<th>Clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Group</strong></td>
<td><strong>Group – Dyadic Fairness</strong></td>
<td><strong>Group</strong></td>
</tr>
<tr>
<td>▪ Assortativity (Newman, 2003)</td>
<td>▪ Statistical Parity or Disparate Impact</td>
<td>▪ Same proportion of each group in each cluster as in the population as a whole</td>
</tr>
<tr>
<td>▪ Modularity: modred (Masrour, 2020)</td>
<td>[Laclau, 2020; Rahman, 2019; Buyl, 2020; Li, 2021; Spinelli, 2021]</td>
<td></td>
</tr>
<tr>
<td>[ Q = \frac{1}{2</td>
<td>E</td>
<td>} \sum_{ij} (A_{ij} - \frac{d_i d_j}{2</td>
</tr>
<tr>
<td>▪ Attribute Homophily</td>
<td>▪ Equal opportunity (Buyl, 2020; Li, 2021)</td>
<td>▪ For each node, each cluster must contain an adequate number of members similar* to the individual. *similar defined by some graph R built using sensitive attributes.</td>
</tr>
<tr>
<td>[ h_{edges} = \frac{1}{</td>
<td>E</td>
<td>} \left</td>
</tr>
<tr>
<td>▪ Information Unfairness Score (Jalali 2020)</td>
<td>▪ Equalized odds [Li, 2021]</td>
<td></td>
</tr>
<tr>
<td>[ M = \sum \theta A^k ; \max \left(\min \left(\sum_{v \in V} d(u, v)</td>
<td>\left({v \in V : S_u = S_v } \right) \right) \right) \forall i \in {0,1} ] [ P((u, v)</td>
<td>y_{uv} = i, S_u = S_v) = P((u, v)</td>
</tr>
<tr>
<td>▪ Dirichlet energies wrt node features X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ \mathcal{E}(\gamma) = \text{Tr} [X^{T} L X] ]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Individual</strong></td>
<td><strong>Individual</strong></td>
<td><strong>Individual</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[ \frac{1}{</td>
</tr>
</tbody>
</table>

* More group metrics like ARP (Rahman, 2019) or DI or EO based on group and subgroup dyadic-specific (Spinelli, 2021) (above metrics are mixed dyadic)
Graph Fairness Definitions
Definitions and metrics from a Pipeline Point of View

- **Link Prediction**
  - Group
    - Dyadic Fairness
    - DI / EOd / EOp
  - Individual
    - \( \text{Tr}(A^T \phi^T L \phi A) \)

- **Topology**
  - Group
    - Assortativity
    - Attribute Homophily

- **Clustering**
  - Group
    - Balance (SC)
    - ModRes
  - Individual
    - SC

- **Embeddings**
  - Group
    - Distributional Bias
    - Representation Bias (AUC predicting A)
  - Individual
    - Pair Distance \( \text{Tr}(Y^T L_s Y) \)
    - Ranking similarity

- **Classification**
  - Group
    - DI/EOd/Eop
    - Counterfactual (% change \( f(X) \) when change A)
  - Individual
    - Consistency
    - Degree
Graph Fairness Definitions

Definitions and metrics from a Pipeline Point of View

Graph Rewiring
- Benefits all tasks in the pipeline
- Provides a strong interpretability
- Lot of theory behind
- Aligned with other problems in GNNs
  - Homophily/Heterophily
  - Expressiveness
Rewiring for Topology Debiasing
On the Information Unfairness of Social Networks

Information Unfairness
Maximum difference between distribution of intra and inter edge weights

\[ D_{fg} = \{ A_{uv} : S_u = f, S_v = g \} \]

Assortativity = 0.66

IU: 0.19
Assortativity = 0.66

IU: 0.32
*same intra-inter edges
Rewiring for Topology Debiasing
On the Information Unfairness of Social Networks

MaxFair
Find $b$ edges such that the IU of $G' = (V, E \cup B)$ is minimized

1. Calculate Node-Attribute centrality: Quantify how well a node spreads information into a group
   - $vec_s \in \mathbb{R}^{1 \times n} = \sum_k p^k \times vec_{s,k} :$ each node’s centrality with respect to sensitive group. One for each $s \in S$.
   - $vec_{s,0} :$ vector of node membership to sensitive group. i.e. $vec_{s,0}(u) = 1$ if $S_u = f$ else 0
   - $vec_{s,k}(u) = \text{sum} \left( [vec_{s,k-1}(v)]_{v \in N(u)} \right) :$ message passing using $vec_{s,0}$ as initial feature

2. Score unconnected pair of nodes using $vec_s$
   - $A = \sum pM^k \rightarrow D_{fg} = \{A_{uv} : S_u = f, S_v = g\}, \forall f, g \in S$, i.e.
   - $s_{fg} = \text{mean}(A) - \text{mean}(D_{fg})$. How each distribution deviate from the mean of all edges.
   - score$(u, v) = \sum_{f, g \in S} s_{fg} \ast (vec_f(u) \ast vec_g(v) + vec_g(u) \ast vec_f(v))$

3. Select the highest scoring edge

Rewiring for Fair Link Prediction
Bursting the Filter Bubble: Fairness-aware network link prediction

Evaluate Structural Fairness by change in modularity after link prediction

Greedy-FLIP
Greedy rewiring at post-processing

\[
Q = \frac{1}{2|E|} \sum_{ij} \left( A_{ij} - \frac{d_i d_j}{2|E|} \right) (S_u \otimes S_v)
\]

\[
\text{modred} = \frac{Q - Q'}{Q}
\]

How flipping an edge prediction change the modularity?
Flip edge with the lowest score and repeat

\[
\text{score}(\hat{e}_{xy}) = \frac{(-1)^{\delta(e_{xy})}}{2m} \left( -1 + \frac{d_x + d_y - 1}{2m} \right) \delta(X_x^{(p)}, X_y^{(p)})
\]

\[
+ \left( \sum_{v \in V, X_v^{(p)} \neq X_x^{(p)}} d_v + \sum_{v \in V, X_v^{(p)} \neq X_y^{(p)}} d_v \right) / 4m^2
\]

Masrour, F., et al. “Bursting the filter bubble: Fairness-aware network link prediction”. In AAAI, 2020
Adversarial Learning for Fair Link Prediction

Bursting the Filter Bubble: Fairness-aware network link prediction

Evaluate Structural Fairness by change in modularity after link prediction

\[ Q = \frac{1}{2|E|} \sum_{ij} \left( A_{ij} - \frac{d_i d_j}{2|E|} \right) (S_u \otimes S_v) \]
modred = \frac{Q - Q'}{Q}

Masrour, F., et al. "Bursting the filter bubble: Fairness-aware network link prediction". In AAAI, 2020
Rewiring for Fair Link Prediction

On dyadic fairness: Exploring and mitigating bias in graph connections

Dyadic Fairness: \( P((u, v) | S_u = S_v) = P((u, v) | S_u \neq S_v) \) \( \rightarrow \) predict equal number of

FairAdj
Rewire the graph topology to get fair embeddings to perform fair link prediction using projected gradient descent \( \rightarrow \) maintain \( \mathbf{A} \) nature

- They prove that their rewiring reduces an upper bound of a constant that, if low, is a sufficient condition for Dyadic Fairness
  - It reduces the disparity of representation between nodes of different groups after message passing

* Also, same TPR, TNR, FPR and FNR

\[
\begin{align*}
\max_\mathbf{A} \mathcal{L}_{VGAE} := & \mathbb{E} \left[ \log p(\mathbf{A}|\mathbf{Z}) \right] - K \left[ \text{GNN}(\mathbf{Z}|\mathbf{X},\mathbf{A}) \right] \| N(0,1) \\
\min_\mathbf{A} \mathcal{L}_{\text{fair}} := & \left\| \mathbb{E}_{u,v \sim U \times U} [h(u,v)|S_u = S_v] - \mathbb{E}_{u,v \sim U \times U} [h(u,v)|S_u \neq S_v] \right\|^2 \\
& \text{s.t } [\mathbf{A}]_{uv} = 0 \text{ if } [\mathbf{A}]_{uv} = 0 \text{ and } \mathbf{A}^1 = 1
\end{align*}
\]

\( \mathbf{A} = \mathbf{A} - (\eta \nabla_\mathbf{A} \mathcal{L}_{\text{fair}}) \)

Project \( \mathbf{A} - (\eta \nabla_\mathbf{A} \mathcal{L}_{\text{fair}}) \) to the feasible space \( \mathbf{A} - (\eta \nabla_\mathbf{A} \mathcal{L}_{\text{fair}}) \mathbf{1} = \mathbf{1} \)

\( \mathbf{A} \) is row stochastic

Li, P., et al. “On dyadic fairness: Exploring and mitigating bias in graph connections”. In ICLR, 2021
Rewiring for Fair Representation Learning

FairDrop: Biased edge dropout for enhancing fairness in Graph Representation Learning

Fairness: AUC predicting $S$ *they also perform link prediction evaluated with dyadic fairness

**FairDrop**

Fair edge dropout

- Dropout homophilic edges with prob $\frac{1}{2} + \delta$
- Dropout heterophilic edges with prob $\frac{1}{2} - \delta$

Spinelli, L., et al. “FairDrop: Biased edge dropout for enhancing fairness in GRL”. In TAI 2021

$\delta = 0$

$\delta = 0.1$

$\delta = 0.35$

$\delta = 0.5$
That’s not all Folks!
More Graph Rewiring Methods for Graph Fairness

RW for topology debiasing
- MaxFair
  Jalali Z. S., et al. “On the information unfairness of social networks”. In SDM, 2020

RW first for link prediction
- Greedy-FLIP
- FairAdj
- FairDrop
  Spinelli, I., et al. “FairDrop: Biased edge dropout for enhancing fairness in GRL”. In TAI 2021
- OT: Individual Fairness

RW for fair representation learning
- InForm: Individual Fairness
- FairDrop – [Spinelli, I., 2021]
- FairAdj – [Li, P., 2021]

RW for node classification
- OT - [Laclau, C., 2020]
- EDITS
- FairEdit

RW for specific applications
- Recommender systems
What can we do now?

- Normalization of benchmarks, evaluation metrics and pipelines
- Formalization of Graph Fairness as happens in Algorithmic Fairness
- Beyond Dyadic fairness
- Accuracy-fairness tradeoff in Graph Fairness?
- More efficient and Interpretable Rewiring Methods
- Causality Aware GNNs for fairness
- Ethical challenges:
  - Different values and philosophical fairness definitions
  - Human-in-the-loop
  - Robustness, XAI, privacy...
  - Go beyond known, measurable, discrete and static sensitive attributes*
Acknowledgments

The following authors...

Adrián Arnaiz Rodríguez
ELLIS Alicante
Speaker, panel moderator and content creator

Francisco Escolano
University of Alicante
ELLIS Alicante
Speaker and content creator

Nuria Oliver
ELLIS Alicante
Panel moderator and content creator

Edwin Hancock
University of York
Speaker and content creator

Ahmed Begga
University of Valencia
Content creator

... want to thanks to LoG reviewers of DiffWire and other researches that we got feedback from!

This work is partially funded by:
Bibliography

Graph Rewiring


Bibliography
Algorithmic Fairness

Rewiring Methods
❖ Jalali Z. S., et al. "On the information unfairness of social networks". In SDM, 2020
❖ Spinelli, I., et al. "FairDrop: Biased edge dropout for enhancing fairness in GRL", in TAI 2021
❖ More Methods

❖ Dong, Y. et al. GUIDE: Group Equality Informed Individual Fairness in Graph Neural Networks. In KDD 2022.

Topology Bias
❖ Dong, Y., et al. "On Structural Explanation of Bias in Graph Neural Networks". In KDD 2022.
❖ Algorithmic Fairness and Network Science
❖ Hampson, M. "Smart Algorithm Bursts Social Networks' 'Filter Bubbles'. 2011.
❖ Surveys on Graph Fairness
❖ Hampson, M. "Smart Algorithm Bursts Social Networks' 'Filter Bubbles'. 2011.
❖ Tutorials
❖ Recommender Systems
❖ Influence Maximization
Motivation and Challenges

Introduction to Spectral Theory

Transductive Rewiring

Inductive Rewiring

Graph Fairness

Panel Discussion
Panel
The Institute for Humanity-Centric Artificial Intelligence

https://ellisalicante.org/tutorials/GraphRewiring

https://github.com/ellisalicante/GraphRewiring-Tutorial

@arnaiztech
@Fescolano
@nuriaoliver
@ahmexbeg115

Thanks

adrian@ellisalicante.org