Locality-Aware Graph Rewiring in GNNs

Fabian Bosshard

November 27, 2025

Contents

List of Acronyms

GNN	Graph Neural Network
MPNN	Message Passing Neural Network

1 Background

Let $G = (V, E)$ be an undirected graph with the adjacency matrix $\underline{\boldsymbol{A}} \in \mathbb{R}^{n \times n}$

$$\underline{\boldsymbol{A}}_{uv} = \begin{cases} 1 & (u, v) \in E \\ 0 & (u, v) \notin E \end{cases} \tag{1}$$

The diagonal degree matrix $\underline{\boldsymbol{D}} \in \mathbb{R}^{n \times n}$ is defined by

$$\underline{\boldsymbol{D}}_{uv} = \begin{cases} d_u & u = v \\ 0 & u \neq v \end{cases} \tag{2}$$

i.e. $\underline{\boldsymbol{D}}$ simply places all node degrees on the diagonal.

1.1 normalized adjacency and multi-hop propagation

Definition 1. The symmetrically normalized adjacency matrix is

$$\boxed{\hat{\underline{\boldsymbol{A}}} = \underline{\boldsymbol{D}}^{-1/2}\underline{\boldsymbol{A}}\underline{\boldsymbol{D}}^{-1/2}} \tag{3}$$

or, entrywise,

$$\hat{\underline{\boldsymbol{A}}}_{uv} = \begin{cases} \frac{1}{\sqrt{d_u d_v}} & (u,v) \in E \\ 0 & (u,v) \notin E \end{cases}$$

✍

Proposition 1 (multi-hop propagation)**.** The entry $(\hat{\underline{\boldsymbol{A}}}^k)_{vu}$ can be computed explicitly as follows:

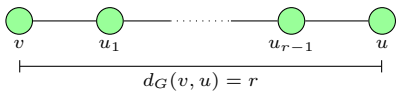
$$(\hat{\underline{\boldsymbol{A}}}^k)_{vu} = \sum_{\pi} \prod_{(x,y)\in\pi} \frac{1}{\sqrt{d_x d_y}} \tag{4}$$

where the sum is over all walks $\pi = (v, \ldots, u)$ of length k from v to u. ◁

Corollary 2. Let $v, u \in V$ with $r = d_G(v,u)$, where $d_G(\cdot,\cdot)$ denotes the shortest-path distance. Assume there is exactly one path

$$(v, u_1, \ldots, u_{r-1}, u)$$

of length r between v and u:



Then

$$\begin{aligned} (\hat{\underline{\boldsymbol{A}}}^r)_{vu} &= \frac{1}{\sqrt{d_v d_{u_1}}} \cdot \prod_{i=1}^{r-2} \frac{1}{\sqrt{d_{u_i} d_{u_{i+1}}}} \cdot \frac{1}{\sqrt{d_{u_{r-1}} d_u}} \\ &= \frac{1}{\sqrt{d_v d_u}} \prod_{i=1}^{r-1} \frac{1}{d_{u_i}} \end{aligned} \tag{5}$$

◁

1.1.1 distance layers and layer degrees

Definition 2. For $\ell \in \mathbb{N}_0$, we define the distance-$(\ell+1)$ adjacency matrix $\underline{\boldsymbol{A}}_\ell \in \mathbb{R}^{n\times n}$ by

$$(\underline{\boldsymbol{A}}_\ell)_{uv} = \begin{cases} 1 & d_G(u,v) = \ell + 1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $d_G(u,v)$ is the shortest-path distance. The corresponding layer degree of a node v at distance level ℓ is

$$d_{v,\ell} = \sum_{u\in V} (\underline{\boldsymbol{A}}_\ell)_{vu}, \tag{7}$$

i.e. the number of nodes at graph distance $\ell+1$ from v. Let $\underline{\boldsymbol{D}}_\ell$ be the diagonal matrix with $(\underline{\boldsymbol{D}}_\ell)_{vv} = d_{v,\ell}$. The normalized distance-$(\ell+1)$ adjacency is

$$\boxed{\hat{\underline{\boldsymbol{A}}}_\ell = \underline{\boldsymbol{D}}_\ell^{-1/2}\underline{\boldsymbol{A}}_\ell\underline{\boldsymbol{D}}_\ell^{-1/2}} \tag{8}$$

so that

$$(\hat{\underline{\boldsymbol{A}}}_\ell)_{uv} = \begin{cases} \frac{1}{\sqrt{d_{u,\ell} d_{v,\ell}}} & d_G(u,v) = \ell + 1 \\ 0 & \text{otherwise} \end{cases}$$

✍

Finally, we denote by

$$d_{\min} = \min_{v\in V} d_v \tag{9}$$

the minimum node degree in the graph.

1.2 graph Laplacian

Definition 3. The combinatorial graph Laplacian is

$$\underline{\boldsymbol{L}} = \underline{\boldsymbol{D}} - \underline{\boldsymbol{A}} \tag{10}$$

and the normalized graph Laplacian is

$$\boxed{\hat{\underline{\boldsymbol{L}}} = \underline{\boldsymbol{D}}^{-1/2}\underline{\boldsymbol{L}}\underline{\boldsymbol{D}}^{-1/2} \overset{(10)}{=} \underline{\boldsymbol{D}}^{-1/2}(\underline{\boldsymbol{D}} - \underline{\boldsymbol{A}})\underline{\boldsymbol{D}}^{-1/2} \overset{(3)}{=} \underline{\boldsymbol{I}}_n - \hat{\underline{\boldsymbol{A}}}} \tag{11}$$

It is symmetric and positive semidefinite, and its eigenvalues satisfy

$$0 = \lambda_0 \le \lambda_1 \le \cdots \le \lambda_{n-1}$$

λ_1 is called the spectral gap. The number of zero eigenvalues (i.e., the multiplicity of the 0 eigenvalue) equals the number of connected components of the graph. ✍

To understand Definition 3, consider a function $f\colon V \to \mathbb{R}$. Denote by $\vec{\boldsymbol{f}} \in \mathbb{R}^n$ the vector whose v-th entry is $f(v)$. Then

$$(\hat{\underline{\boldsymbol{L}}}\vec{\boldsymbol{f}})_v = f(v) - \frac{1}{\sqrt{d_v}} \sum_{(u,v)\in E} \frac{f(u)}{\sqrt{d_u}} \tag{12}$$

i.e., $(\hat{\underline{\boldsymbol{L}}}\vec{\boldsymbol{f}})_v$ is the value at v minus a degree-normalized average of the neighbors. This is why the Laplacian is often viewed as a discrete second derivative on the graph: it measures how much f at v deviates from its neighborhood. Another important identity is the quadratic form

$$\vec{\boldsymbol{f}}^{\top}\underline{\boldsymbol{L}}\vec{\boldsymbol{f}} = \frac{1}{2}\sum_{(u,v)\in E} \big(f(u) - f(v)\big)^2 \tag{13}$$

which shows that $\underline{\boldsymbol{L}}$ (and hence also $\hat{\underline{\boldsymbol{L}}}$) is positive semidefinite, since the right-hand side is always nonnegative. Moreover, (13) is small exactly when f varies slowly across edges, so the Laplacian encodes the smoothness of functions on the graph.

1.3 Cheeger inequality

The Cheeger inequality relates the spectral gap λ_1 to the Cheeger constant $h(G)$, which measures how difficult it is to separate the graph into two large pieces. It states, in particular, that

$$\tfrac{1}{2}h(G)^2 \le \lambda_1 \le 2h(G),$$

so a larger spectral gap implies that the graph is more "well-connected".

1.4 effective resistance

Definition 4 (effective resistance)**.** View each edge $(u, v) \in E$ as an electrical resistor of resistance $1\,\Omega$. The resulting network has a well-defined resistance between any two nodes.

For two nodes $s, t \in V$, the effective resistance $R(s, t)$ is defined as the voltage difference needed to send one unit of electrical current from s to t. It can be computed as

$$\boxed{R(s,t) = (\vec{\boldsymbol{e}}_s - \vec{\boldsymbol{e}}_t)^{\top}\underline{\boldsymbol{L}}^{\dagger}(\vec{\boldsymbol{e}}_s - \vec{\boldsymbol{e}}_t)} \tag{14}$$

where $\underline{\boldsymbol{L}}^{\dagger}$ is the Moore–Penrose pseudoinverse of $\underline{\boldsymbol{L}}$ and $\vec{\boldsymbol{e}}_v$ is the standard basis vector of vertex v. ✍

1.4.1 Interpretation

If the graph offers many short, parallel paths between s and t, then current can flow easily, so $R(s, t)$ is small. If there are few or long paths, the current is "bottlenecked" and $R(s, t)$ is large. Thus, effective resistance measures how "well-connected" two nodes are inside the global geometry of the graph.

1.4.2 Connection to random walks

A random walk on G is the Markov chain that, from a node v, moves to a uniformly random neighbor of v. Its transition matrix is

$$\underline{\boldsymbol{P}} = \underline{\boldsymbol{D}}^{-1}\underline{\boldsymbol{A}} \tag{15}$$

so $\underline{\boldsymbol{P}}_{vu} = 1/d_v$ if $(v,u) \in E$.

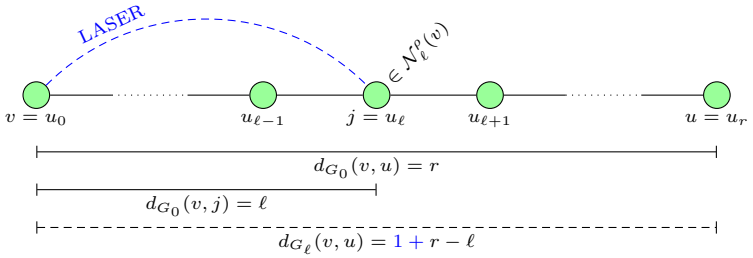
For two nodes u, v, the commute time $\texttt{CT}(u,v)$ is the expected number of steps for the random walk to start at u, reach v, and return to u again. It can be related to the effective resistance via

$$\texttt{CT}(u,v) = 2|E|R(u,v) \tag{16}$$

giving a geometric interpretation of how "far apart" two nodes are in terms of random-walk behavior, i.e. two nodes have small commute time exactly when they have small effective resistance.

2 Theoretical analysis

2.1 Jacobian sensitivity



We have a unique path

$$(v = u_0, u_1, \ldots, u_{r-1}, u_r = u),$$

and we assume $j = u_\ell$. From Corollary 2, for the full path from v to u:

$$(\hat{\underline{\boldsymbol{A}}}^{r})_{vu} = \frac{1}{\sqrt{d_v d_u}} \prod_{s=1}^{r-1} \frac{1}{d_{u_s}}$$

For the sub-path from $j = u_\ell$ to $u = u_r$ of length $r - \ell$, the same reasoning gives

$$(\hat{\underline{\boldsymbol{A}}}^{r-\ell})_{ju} = \frac{1}{\sqrt{d_j d_u}} \prod_{s=\ell+1}^{r-1} \frac{1}{d_{u_s}}.$$

Now we plug this into our expression:

$$\begin{aligned}
\frac{(\hat{\underline{\boldsymbol{A}}}_{\ell-1})_{vj}(\hat{\underline{\boldsymbol{A}}}^{r-\ell})_{ju}}{\frac{1}{\sqrt{d_v d_u}}\prod_{s=1}^{r-1}\frac{1}{d_{u_s}}} &= \frac{(\hat{\underline{\boldsymbol{A}}}_{\ell-1})_{vj}}{\frac{1}{\sqrt{d_v d_u}}\prod_{s=1}^{\ell-1}\frac{1}{d_{u_s}}} \cdot \frac{(\hat{\underline{\boldsymbol{A}}}^{r-\ell})_{ju}}{\prod_{s=\ell}^{r-1}\frac{1}{d_{u_s}}} \\
&\overset{(5)}{=} \frac{(\hat{\underline{\boldsymbol{A}}}_{\ell-1})_{vj}}{\frac{1}{\sqrt{d_v d_u}}\prod_{s=1}^{\ell-1}\frac{1}{d_{u_s}}} \cdot \frac{\frac{1}{\sqrt{d_j d_u}}\prod_{s=\ell+1}^{r-1}\frac{1}{d_{u_s}}}{\prod_{s=\ell}^{r-1}\frac{1}{d_{u_s}}} \\
&= \frac{(\hat{\underline{\boldsymbol{A}}}_{\ell-1})_{vj}}{\frac{1}{\sqrt{d_v d_u}}\prod_{s=1}^{\ell-1}\frac{1}{d_{u_s}}} \cdot \frac{1}{\sqrt{d_j d_u}} \cdot \frac{\prod_{s=\ell+1}^{r-1}\frac{1}{d_{u_s}}}{\frac{1}{d_{u_\ell}}\prod_{s=\ell+1}^{r-1}\frac{1}{d_{u_s}}} \\
&= \frac{(\hat{\underline{\boldsymbol{A}}}_{\ell-1})_{vj}}{\frac{1}{\sqrt{d_v d_u}}\prod_{s=1}^{\ell-1}\frac{1}{d_{u_s}}} \cdot \frac{1}{\sqrt{d_j d_u}} \cdot d_{u_\ell} \\
&= \frac{(\hat{\underline{\boldsymbol{A}}}_{\ell-1})_{vj}}{\frac{1}{\sqrt{d_v d_u}}\prod_{s=1}^{\ell-1}\frac{1}{d_{u_s}}} \cdot \sqrt{\frac{d_j}{d_u}}
\end{aligned}$$

So

$$\frac{(\hat{\underline{\boldsymbol{A}}}_{\ell-1})_{vj}(\hat{\underline{\boldsymbol{A}}}^{r-\ell})_{ju}}{\frac{1}{\sqrt{d_v d_u}}\prod_{s=1}^{r-1}\frac{1}{d_{u_s}}} = \frac{(\hat{\underline{\boldsymbol{A}}}_{\ell-1})_{vj}}{\frac{1}{\sqrt{d_v d_u}}\prod_{s=1}^{\ell-1}\frac{1}{d_{u_s}}} \cdot \sqrt{\frac{d_j}{d_u}} \tag{17}$$

Using

$$(\hat{\underline{\boldsymbol{A}}}_{\ell-1})_{vj} = \frac{1}{\sqrt{d_{v,\ell-1} d_{j,\ell-1}}}$$

and

$$\frac{1}{\sqrt{d_v d_u}}\prod_{s=1}^{\ell-1}\frac{1}{d_{u_s}} \leq \frac{1}{d_{\min}^{\ell}}$$

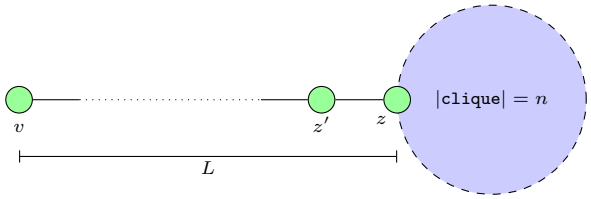
we obtain the bound

$$\frac{(\hat{\underline{\boldsymbol{A}}}_{\ell-1})_{vj}}{\frac{1}{\sqrt{d_v d_u}}\prod_{s=1}^{\ell-1}\frac{1}{d_{u_s}}} \geq \frac{(d_{\min})^{\ell}}{\sqrt{d_{v,\ell-1} d_{j,\ell-1}}} \tag{18}$$

Combining (17) and (18) yields

$$\boxed{\frac{(\hat{\underline{\boldsymbol{A}}}_{\ell-1})_{vj}(\hat{\underline{\boldsymbol{A}}}^{r-\ell})_{ju}}{\frac{1}{\sqrt{d_v d_u}}\prod_{s=1}^{r-1}\frac{1}{d_{u_s}}} \geq \frac{(d_{\min})^{\ell}}{\sqrt{d_{v,\ell-1} d_{j,\ell-1}}} \cdot \sqrt{\frac{d_j}{d_u}}} \tag{19}$$

2.2 Locality awareness



2.3 message passing paradigm

We consider the case where each node v has a feature $\vec{\boldsymbol{x}}_v^{(0)} \in \mathbb{R}^d$. It is common to stack the node features into a matrix $\underline{\boldsymbol{X}}^{(0)} \in \mathbb{R}^{n \times d}$ consistently with the ordering of $\underline{\boldsymbol{A}}$. Graph Neural Networks (GNNs) are functions defined on the featured graph that can output node, edge, or graph-level values. The most common family of GNN architectures are Message Passing Neural Networks (MPNNs), which compute latent node representations by stacking T layers of the form:

$$\boxed{\vec{\boldsymbol{x}}_v^{(t)} = \mathtt{up}^{(t)}(\vec{\boldsymbol{x}}_v^{(t-1)}, \mathtt{agg}^{(t)}(\{\vec{\boldsymbol{x}}_u^{(t-1)} : (v,u) \in E\}))} \tag{20}$$

for $t = 1, \ldots, T$, where $\mathtt{agg}^{(t)}$ is some permutation-invariant aggregation function, while $\mathtt{up}^{(t)}$ updates the node's current state with aggregated messages from its neighbors.

2.4 over-squashing and long-range interactions

While the message-passing paradigm usually constitutes a strong inductive bias, it is problematic for capturing long-range interactions due to a phenomenon known as over-squashing. Given two nodes u, v at distance $d_G(u,v) = r$, an MPNN will require $T \geq r$ layers to exchange messages between them. When the receptive fields of the nodes expand too quickly (due to volume growth properties characteristic of many real-world scale free graphs), the MPNN needs to aggregate a large number of messages into fixed-size vectors, leading to some corruption of the information. This effect on the propagation of information has been related to the Jacobian of node features decaying exponentially with r. More recently, it was shown that the Jacobian is affected by topological properties such as effective resistance.

2.5 graph rewiring

The main principle behind graph rewiring in GNNs is to decouple the input graph G from the computational one. Namely, rewiring consists of applying an operation $\mathcal{R}$ to $G = (V, E)$, thereby producing a new graph $\mathcal{R}(G) = (V, \mathcal{R}(E))$ on the same vertices but with altered connectivity. We begin by generalizing the MPNN formalism to account for the rewiring operation $\mathcal{R}$ as follows:

$$\vec{\boldsymbol{x}}_v^{(t)} = \mathtt{up}^{(t)}(\vec{\boldsymbol{x}}_v^{(t-1)}, \mathtt{agg}_G^{(t)}(\{\vec{\boldsymbol{x}}_u^{(t-1)} : (v,u) \in E\}), \mathtt{agg}_{\mathcal{R}(G)}^{(t)}(\{\vec{\boldsymbol{x}}_u^{(t-1)} : (v,u) \in \mathcal{R}(E)\})) \tag{21}$$

where a node feature is now updated based on information collected over the input graph G and the rewired one $\mathcal{R}(G)$, through (potentially) independent aggregation maps. Many rewiring-based GNN models simply exchange messages over $\mathcal{R}(G)$, i.e., they take $\mathtt{agg}_G = 0$. The idea of rewiring the graph is implicit to many GNNs, from using Cayley graphs, to virtual nodes and cellular complexes. Other works have studied the implications of directly changing the connectivity of the graph to de-noise it, or to explore multi-hop aggregations. Ever since over-squashing was identified as an issue in MPNNs, several novel rewiring approaches have been proposed to mitigate this phenomenon.

3 A general paradigm: dynamic rewiring with local constraints

Main idea. Our main contribution is a novel paradigm for graph rewiring that satisfies criteria (i)–(iii), leveraging a key principle: instead of considering a *single* rewired graph $\mathcal{R}(G)$, we use a *sequence* of rewired graphs $\{\mathcal{R}_\ell(G)\}_\ell$ such that for smaller ℓ, the new edges added in $\mathcal{R}_\ell(G)$ are more 'local' (with respect to the input graph G) and sampled based on optimizing a connectivity measure.

In this Section, we discuss a general graph-rewiring paradigm that can enhance any MPNN and satisfies the criteria (i)–(iii) described above. Given a graph G, consider a trajectory of rewiring operations $\mathcal{R}_\ell$, starting at $G_0 = G$, of the form:

$$G = G_0 \xhookrightarrow{\mathcal{R}_1} G_1 \xhookrightarrow{\mathcal{R}_2} \cdots \xhookrightarrow{\mathcal{R}_L} G_L \tag{22}$$

Since we think of G_ℓ as the input graph evolved along a dynamical process for ℓ iterations, we refer to G_ℓ as the *ℓ-snapshot*. For the sake of simplicity, we assume $\mathcal{R}_\ell = \mathcal{R}$, though it is straightforward to extend the discussion below to the more general case. In order to account for the multiple snapshots, we modify the layer form in (21) as

$$\vec{\boldsymbol{x}}_v^{(t)} = \mathtt{up}^{(t)}\left(\vec{\boldsymbol{x}}_v^{(t-1)}, \left(\mathtt{agg}_{G_\ell}^{(t)}(\{\vec{\boldsymbol{x}}_u^{(t-1)} : (v,u) \in E_\ell\})\right)_{0 \le \ell \le L}\right) \tag{23}$$

where, instead of aggregating messages over a single rewired graph, we now consider all the snapshots G_ℓ in the sequence.

Below we describe a rewiring paradigm based on an arbitrary ***connectivity measure*** $\mu : V \times V \to \mathbb{R}$ and ***locality measure*** $\nu : V \times V \to \mathbb{R}$. The measure μ can be any topological quantity that captures how easily different pairs of nodes can communicate in a graph, while the measure ν is any quantity that penalizes interactions among nodes that are 'distant' according to some metric on the input graph. In a nutshell, our choice of $\mathcal{R}$ *samples edges to add according to the constraint ν, prioritizing those that maximally benefit the measure μ.* By keeping this generality, we provide a universal approach to do graph-rewiring that can be of interest independently of the specific choices of μ and ν.

Improving connectivity while preserving locality. The first property we demand of the rewiring sequence is that for all nodes v, u, we have $\mu_{G_{\ell+1}}(v,u) \ge \mu_{G_\ell}(v,u)$ and that for *some* nodes, the inequality is *strict*. If we connect all pairs of nodes with low μ-value, however, we might end up adding non-local edges across distant nodes, hence quickly corrupting the locality of G. To avoid this, we *constrain* each rewiring by requiring the measure ν to take values in a certain range $\mathcal{I}_\ell \subset [0, \infty)$: an edge (v,u) appears in the ℓ-snapshot (for $1 \le \ell \le L$) according to the following rule:

$$(v,u) \in E_\ell \text{ if } \left(\mu_{G_0}(v,u) < \epsilon \text{ and } \nu_{G_0}(v,u) \in \mathcal{I}_\ell\right) \text{ or } (v,u) \in E_{\ell-1}. \tag{24}$$

To make the rewiring more efficient, the connectivity and locality measures are computed *once* over the input graph G_0. Since the edges to be added connect nodes with low μ-values, the rewiring makes the graphs G_ℓ friendlier to message-passing as ℓ grows. Moreover, by taking increasing ranges of values for the intervals $\mathcal{I}_\ell$, we make sure that new edges connect distant nodes, as specified by ν, only at later snapshots. Sequential rewiring allows us to interpolate between the given graph and one with better connectivity, creating intermediate snapshots that *progressively* add non-local edges. By accounting for all the snapshots G_ℓ in (21), the GNN can access both the input graph, and more connected ones, at a much *finer level* than 'instantaneous' rewirings, defined next.