Diffusion-Jump GNNs: Homophiliation via Learnable Metric Filters

Ahmed Begga Hachlafi
University of Alicante, Spain
ahmedbegga@gmail.com

Francisco Escolano
University of Alicante, Spain
escolano.ua@gmail.com

Miguel Angel Lozano
University of Alicante, Spain
malozano@ua.es

Edwin R. Hancock
University of York, U.K.
edwin.hancock@york.ac.uk

Abstract

High-order Graph Neural Networks (HO-GNNs) have been developed to infer consistent latent spaces in the heterophilic regime, where the label distribution is not correlated with the graph structure. However, most of the existing HO-GNNs are *hop-based*, i.e., they rely on the powers of the transition matrix. As a result, these architectures are not fully reactive to the classification loss and the achieved structural filters have static supports. In other words, neither the filters' supports nor their coefficients can be learned with these networks. They are confined, instead, to learn combinations of filters. To address the above concerns, we propose DIFFUSION-JUMP GNNS− a method relying on asymptotic diffusion distances that operates on *jumps*. A *diffusion-pump* generates pairwise distances whose projections determine both the support and coefficients of each structural filter. These filters are called jumps because they explore a wide range of scales in order to find bonds between scattered nodes with the same label. Actually, the full process is controlled by the classification loss. Both the jumps and the diffusion distances react to classification errors (i.e. they are learnable). *Homophiliation*, i.e., the process of learning piecewise smooth latent spaces in the heterophilic regime, is formulated as a Dirichlet problem: the known labels determine the border nodes and the diffusion-pump ensures a minimal deviation of the semi-supervised grouping from a canonical unsupervised grouping. This triggers the update of both the diffusion distances and, consequently, the jumps in order to minimize the classification error. The Dirichlet formulation has several advantages. It leads to the definition of *structural heterophily*, a novel measure beyond edge heterophily. It also allows us to investigate links with (learnable) diffusion distances, absorbing random walks and stochastic diffusion. Finally, our experimental results outperform significantly those of the state-of-the-art both in homophilic and heterophilic datasets. We are very competitive for large graphs.

1 Introduction

The success of Graph Neural Networks (GNNs) relies on their convolutional architecture [1][2][3]. Their *aggregate and combine* mechanism provides a significant degree of expressiveness. However, in the heterophilic regime, such a mechanism (initially designed for homophilic graphs) results in the over-smoothing issue (shadowing of the internal representations of the nodes, due to a non-selective aggregation). In [4], three solutions are explored: (i) ego and neighbor embedding separation, (ii) higher-order neighborhoods, and (iii) a combination of intermediate representations. The purpose of these mechanisms is to enforce the internal representations of the node features so that the resulting latent space becomes consistent (piecewise smooth), for instance, when the downstream task is node-classification [5][6][7].

In this paper, we explore High-Order GNNs (HO-GNNs). One type of HO-GNNs results from *rewiring the edges* in the graph. For instance, the method in [8] explores neighborhoods of several orders (hops) selecting those orders who provide a high correlation between the node features. GATs [3] are also a well-known rewiring method: the strength of each edge in the input graph is given by a trainable weight. Such a weight is corrected if the concatenation of the node features associated with the corresponding edge has a negative impact on node-classification. Diffwire [9] is another trainable rewiring method. The basic idea of Diffwire is to estimate the commute-times distance between each pair of nodes and use the distance matrix to mask the original adjacency matrix. Other non-differentiable rewiring methods are mainly addressed to alleviate the over-squashing issue (bottlenecks obstruct the message-passing process). A couple of recent examples are [10] and [11].

A second type of HO-GNNs are *Deep/Sequential hop-based* methods, i.e. those models that address over-smoothing with a deep architecture. GGCNs [12] attenuate over-smoothing by performing edge correction (corrected edge weights are learned from node degrees, and signed edges are learned from node features). However, Shortest-Paths-MPNNs [13] and Ordered-GNNs [14] are more focused on performing robust aggregations. Shortest-Paths-MPNNs compute the shortest paths between any pair of nodes. Then, for each node, several separate aggregations are performed (each one for increasing lengths of the shortest paths); then, the resulting embeddings are weighted. Ordered-GNNs rely on a similar principle: for each node, the hierarchy of a tree rooted in that node is aligned with the hops wrt this node in the graph. As neighboring nodes within k hops form a depth$-k$ subtree, aggregations for shallow sub-trees precede those for deeper ones. Interestingly, Ordered-GNNs introduce a differentiable way of deciding the split point between sub-trees.

Finally, *Shallow/Parallel hop-based* methods explore several hop orders in parallel and then integrate the resulting embedding (e.g. via concatenation). MixHop [15], FSGNNs [16] and DualNets [17] compute several powers $\mathbf{P}^k,\ k = 1, 2, \dots, K$ of the normalized adjacency matrix (transition matrix) $\mathbf{P} = \mathbf{D^{-1}A}$. Each power feeds a different GNN. The resulting embeddings are weighed and concatenated for later discrimination. SIGN [18] is similar to MixHop but it precomputes the aggregations $\mathbf{P}^k\mathbf{X}$ for the sake of scalability. More recently, the Simple Graph Convolution (SGG) method [19] improves MixHop by learning polynomials of the transition matrix.

Finally, Generalized PageRank GNNs (GPR-GNNs)[20]learns jointly the best embedding of each node feature and the best weight of each hop. This is very interesting, since the suitability of $\mathbf{P}^k$, which is encoded by a weight γ_k, influences the latent space of the features $\mathbf{H}^0 = f_\Theta(\mathbf{X})$ through a learnable function $f_\Theta(.)$. As a result the $k-$th embedding is $\mathbf{H}^k = \mathbf{P}^k\mathbf{H}^0$. This mechanism allows GPR-GNNs to avoid over-smoothing and trade node and topology feature informativeness. However, this strategy produces inconsistent results since GPR-GNNs are better suited for heterophilic graphs, instead of being also useful for homophilic graphs.

Main Limitation of HO-GNNs. Most of the existing HO-GNNs explore different powers of the normalized adjacency matrix (transition matrix) $\mathbf{P}$. In other words, they are completely *hop-based*. As a result, the HO-GNNs exploit the labels of the semi-supervised learning process either to alleviate the over-smoothing issue (in the sequential case [12][13] [14]) or to weigh the importance of each hop order (in the parallel case [15][16][17][18]). However, as the structure of the input graph is *static*, the hops are static as well. Consequently, *the labels cannot be backpropagated to change the structure of the hops, but only the relative importance of each hop or the extent of its aggregation support*.

Implications. As a result, dealing with heterophilic graphs goes beyond the potential achievements of hop-based approaches (see our Experiments in Section 5). Despite high-order hops being able of connecting distant nodes with the same label, such connections can be neither attenuated nor amplified for the sake of the classification loss. In other words, the probability that

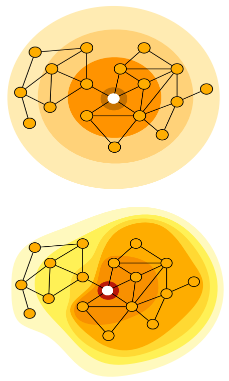


Figure 1: Hop-hierarchy (Top) vs Jump-hierarchy (Bottom). Diffusion distances contract the similarity space due to *structural forces*.

a random walk links two nodes is an *in-place coefficient*, not the realization of a probabilistic event. In this regard, parallel HO-GNNs claim that the powers $\mathbf{P}^k$ can be interpreted as a bank of *structural filters*, i.e. a bunch of aggregators inspired by convolutional filters such as Gabor receptive filds [15]. However, an expressive characterization of a structural filter requires that both its *support* (specification of what edges have a non-zero coefficient) and its *coefficients* are learnable.

Our contributions. In this paper, we address the problem of learning a bank of expressive structural filters as follows:

a) We re-formulate the problem of node-classification under heterophily in terms of a **Dirichlet problem**, i.e. we have *border nodes* (training set) where the classification is optimal and *interior nodes* (remaining nodes) where the resulting latent space (node embedding) must be as harmonic (piecewise smooth) as possible, even when the labeling is far from being harmonic over the graph.

b) We have a **diffusion pump** which generates *asymptotic diffusion distances* $d(i,j) = d^{t\rightarrow\infty}(i,j)$ between the nodes by learning the nontrivial top eigenvectors of $\mathbf{P}$ subject to the labeling of the training set.

c) Given the diffusion distances $d(i,j)$ we compute the *jump hierarchy* (see Figure 1). For a node i we have that $i \in J_i^0$ $(k=0)$ is the closest node wrt itself; $j_1 \in J_i^1$ $(k=1)$ are nodes so that only j_1 is closer to node i than any of them, $j_2 \in J_i^2$ $(k=2)$ are nodes so that only the nodes in J^1 are closer to i than any of them, and so on. Each of the sets $J^k = \bigcup_{i=1}^{|V|} J_i^k$, where V are the nodes of the graph $G=(V,E)$, is called a **jump**.

d) The edges $E_k = \{(i_k, j_k) \in V \times V : i_k, j_k \in J^k\}$ define the **support of the jump** and the **coefficients** $c(i_k, j_k) = g(d(i_k, i_k))$ are given by a function $g(.)$ of the diffusion distances (for example the neg-exponential). Then, the **structural filter** $\mathbf{J}^k$ is a matrix with non-zero coefficients only at E_k.

e) Each structural filter $\mathbf{J}^k$ with $k = 0, 1, \ldots, K$ feeds a GNN parameterized by $\mathbf{W}^k$ and the resulting embedding $\mathbf{H}^k = \sigma(\mathbf{J}^k\mathbf{X}\mathbf{W}^k)$ is weighted by a learnable parameter α_k subject to $\sum_{k=0}^{K} \alpha_k = 1$. All the weighted embeddings are concatenated and feed a forward network for classification.

In addition, we propose a novel metric called **structural heterophily** and we denote the process of generating homophilic embeddings from heterophilic graphs as **homophiliation**.

This paper is organized as follows. In Section 2, we formulate semi-supervised learning in terms of a Dirichlet problem. This allows us to measure heterophily in a structural way. Section 3 is devoted to formulating the loss functions and explaining the dynamics of the optimization process. This is done through the analysis of the main modules of our model. Section 4 provides more technical and formal details of our model and establishes links with related inspiring formulations. Our experiments are presented and discussed in Section 5. Finally, our conclusions and future work are summarized in Section 6.

2 Heterophily as the Loss of Harmonicity

Node-classification under heterophily can be posed as the following semi-supervised learning problem. Given an input graph, $G = (V, E)$ with adjacency matrix $\mathbf{A}$ and node features $\mathbf{X}$, there is a node subset $B \subset V$ whose labels $\ell(B)$ are known by the learner (border nodes). Similarly, the labels $\ell(U)$ of the remaining nodes, those in $U = V \sim B$, are hidden (unknown nodes).

Given the graph Laplacian $\triangle$, and a regularizer (minimizer of $\mathbf{x}^T\triangle\mathbf{x} := \sum_{i\sim j}(\mathbf{x}_i - \mathbf{x}_j)^2$), we have $\ell^* = \arg\min_\ell \ell^T\triangle\ell$, where ℓ^* is the smoothest labeling of V after propagating $\ell(B)$ to $\ell(U)$ through the edges of the graph. A Dirichlet solver ensures that the labeling ℓ^* is *Harmonic* (the label of a given unknown node is the average of those of its neighbors) subject to the labeling of the border nodes $\ell(B)$.

In the heterophilic regime, two neighboring nodes rarely share their labels. As a result, ${\ell^*}^T\triangle\ell^* \gg \mathbf{u}^T\triangle\mathbf{u}$, where $\mathbf{u}$ are the vectorized labels obtained by an alternative unsupervised learner. The unsupervised learner typically assumes that the labels $\mathbf{u}$ are correlated with the topology of the graph (homophily). In other words, *heterophily can be posed in terms of how much harmonicity is lost wrt the homophilic assumption*.

The objective of a GNN is to learn a parametric function $f_{\Theta}(\mathbf{A}, \mathbf{X}, \ell(B))$ returning $\mathbf{H}$, a matrix (embedding) of latent representations (one row per node) so that the embeddings of either border nodes or hidden nodes with the same label are grouped together. However, $f_{\Theta}(.)$ does not necessarily minimize $c(\mathbf{H})^T \triangle c(\mathbf{H})$, where $c(.)$ contains the vectorized classification labels. We need to infer a *hidden graph* $G' = (V, E')$ where $c(\mathbf{H})^T \triangle_{G'} c(\mathbf{H})$ is minimized. Actually, the edges E in the hidden graph should link nodes with the same label, even if they are not in the same community.

Structural Heterophily. Given the above formulation we may characterize heterophily in a structural way, namely *as the departure from a structural unsupervised grouping*. In particular, the ratio

$$\mathcal{R} = \frac{\ell^T \triangle \ell}{\mathbf{u}^T \triangle \mathbf{u}} \geq 1 \text{ , where } \ell \text{ is the ground-truth labeling,} \tag{1}$$

is close to the unit if the graph is homophilic (the structure is completely correlated with the labels). For $\mathcal{R} > 1$ the graph is heterophilic. The larger the ratio the larger the heterophily.

We use the example in Figure 1 to illustrate how $\mathcal{R}$ works. We have two communities, $V = A \bigcup \bar{A}$ (left and right respectively). The *white node* belongs naturally to the right one $\bar{A}$, and this is what an unsupervised structural clustering detects: the Fiedler vector $\mathbf{u} = \arg\min_{\mathbf{x} \neq \mathbf{0}, \mathbf{x} \perp \mathbf{1}} \mathbf{x}^T \triangle \mathbf{x}$ has positive components ($\approx +1$) in A and negative components (≈ -1) in $\bar{A}$.

The vector $\mathbf{u}$ is the smallest nontrivial eigenvector of $\triangle$ as well as the largest nontrivial eigenvector of $\mathbf{P}$. It has been argued that the top eigenvectors of $\mathbf{P}$ may be used to decompose the state space into metastable subspaces [21]. In other words, each of the two graph communities in Figure 1 defines a metastable state from which a random walker tries to escape (Section 4).

The average escape time is the inverse of the top nontrivial eigenvalue of $\mathbf{P}$, i.e. the inverse of the approximate spectral gap [22][23][24]. In our example, the spectral gap is very tiny so we can expect large escape times (see more details in Section 5). In particular, the two states defined by the Fiedler vector are very compact (they have low variability). As a result, all the pairs of nodes (i, j) inside each community have very similar asymptotic diffusion distances [25] $d^{t \to \infty}(i, j)$ according to the structural forces characterizing each metastable state.

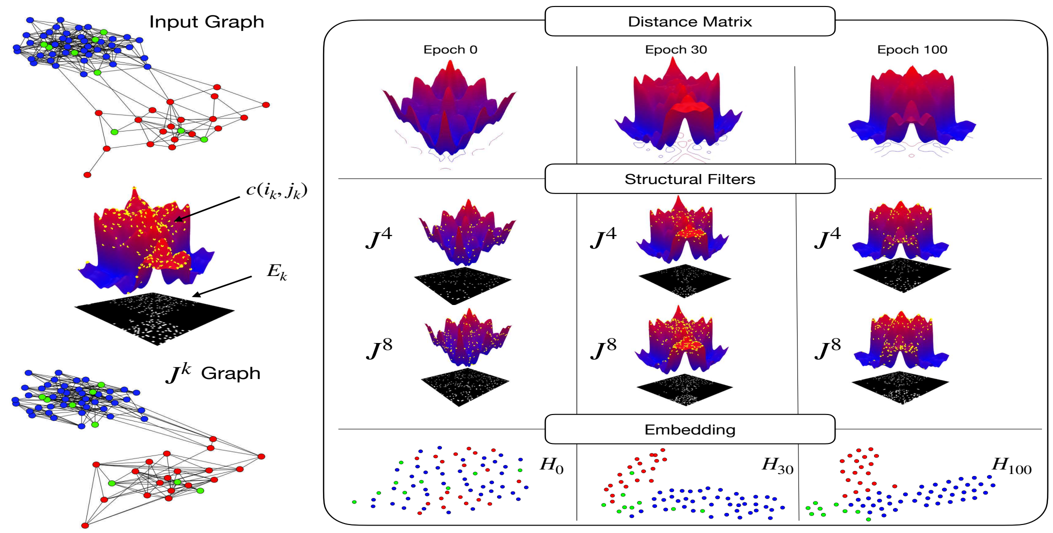
Consequently, the jump hierarchy defines a succession of unstable states $\mathbf{u}_1, \mathbf{u}_2, \ldots$ resulting from the expansion from $\bar{A}$: $\bar{A} \subseteq \bar{A}_1 \subseteq \bar{A}_2 \subseteq \ldots$. They are unstable because their Dirichlet energies $\mathbf{u}_k^T \triangle \mathbf{u}_k$ are greater than that of the unsupervised clustering (*ground energy*) $\mathbf{u}^T \triangle \mathbf{u}$.

Last, but by no means least, if we label the white node as belonging to A instead of belonging to $\bar{A}$ (i.e., *we introduce heterophily*), we also increase the Dirichlet energy wrt the ground energy, i.e. $\ell^T \triangle \ell > \mathbf{u}^T \triangle \mathbf{u}$. Why? This is because the new Fiedler vector $\mathbf{u}_{\ell}$ leading to the labeling ℓ does no longer induce a step function but a hyperbolic tangent with a positive slope. This is consistent with the increase of the spectral gap and the reduction of the escape time.

Therefore, one useful interpretation of heterophily in structural terms (departure from the ground energy) is the fact that heterophily relaxes Dirichlet energies in such a way that it is possible to escape from a community in a few jumps and then find nodes with the same label in other communities. Therefore, paying attention to several jumps simultaneously increases the chance of aggregating distant nodes with the same label, thus solving the heterophily issue.

3 Homophiliation: Losses and Modules

Homophiliation. Our computational model for node-classification under heterophily cannot only rely on finding Harmonic labelings but also on transforming the matrix of node features $\mathbf{X}$ into a piece-wise smooth embedding $\mathbf{H}$. The rows in $\mathbf{H}$ associated with nodes with the same label must be clustered together and these labels must be consistent with those of the border nodes $\ell(B)$. Such a process, i.e. the learning of $f_{\Theta}(\mathbf{A}, \mathbf{X}, \ell(B))$, results from solving the following optimization



.

Figure 2: Homophiliation while the jumps and pairwise distances are learned. Top-Left: a heterophilic graph. Center-Left: The current distance matrix $\mathcal{D}$ leads to a tridimensional weight distribution $c(i,j) = e^{-d(i,j)}$. Yellow points denote the support $E_k = \{(i_k, j_k)\}$ of the filter $\mathbf{J}^k$. The filter coefficients are given by the weights of the support $C_k = \{c(i_k, j_k)\}$. Bottom-Left: graph used for aggregation with this filter. In the right panel, we show the weight distribution (top), a couple of filters (middle), and the resulting homophiliation (bottom) for some epochs. In particular, we show $\mathbf{H}_0$, $\mathbf{H}_{30}$ and $\mathbf{H}_{100}$. In each epoch e, all the embeddings $\{\mathbf{H}_e^k\}$ contribute to identifying potential links between scattered nodes with similar labels. If any of these links is wrong, the matrix of pairwise distances $\mathcal{D}$ is updated.

problem:

$$\begin{aligned} \text{Min} \;\; & \mathcal{L} = \text{Tr}[f_\theta(\mathbf{A})^T \triangle f_\theta(\mathbf{A})] + \mathcal{L}_c(\{\mathbf{J}^k\}, \mathbf{X}, \ell(B)) \\ \text{s.t.} \;\; & \mathbf{U}^T\mathbf{U} = \mathbf{I} \\ & \mathbf{U} = f_\theta(\mathbf{A}),\ \mathcal{D}(i,j) = \|\nabla \mathbf{U}_{ij}\| \;\text{ and }\; \mathbf{J}^k = \Pi^k \cdot \exp\left(-\mathcal{D}\right)\ , \end{aligned} \tag{2}$$

where we have an interplay between the *Dirichlet loss* $\text{Tr}[f_\theta(\mathbf{A})^T \triangle f_\theta(\mathbf{A})]$ and the *classification loss* $\mathcal{L}_c$(Cross-Entrophy) as follows.

Diffusion Pump. Minimizing the structural heterophily so that $\mathcal{R} \approx 1$ (Eq. 1) in $G' = (V, E')$ implies learning Dirichlet energies close to the ground energy. However, in the heterophilic regime, we cannot minimize $c(\mathbf{H})^T \triangle_{G'} c(\mathbf{H})$ before discovering the optimal embedding $\mathbf{H}^*$. In the meanwhile, the Dirichlet formulation allows us to learn the smallest nontrivial eigenvectors of $\triangle$ as we do in the unsupervised setting (e.g. the Fiedler vector). These eigenvectors will be in the columns of $\mathbf{U}$, but they do not have a free form. The notation $\mathbf{U} = f_\theta(\mathbf{A})$ in the above optimization problem goes beyond emphasizing the learnability of $\mathbf{U}$. We also constrain the eigenvectors to be projections/transformations of the adjacency matrix $\mathbf{A}$.

We learn the eigenvectors $\mathbf{U}$ because it is key to computing diffusion distances between the nodes. In the following, we will replace $d(i,j)$ by $\mathcal{D}(i,j)$ when we need to emphasize the matrix nature of the pairwise distances. Each pairwise distance $\mathcal{D}(i,j)$ comes from the norm of $\nabla \mathbf{U}_{ij} = \mathbf{U}_{i:} - \mathbf{U}_{j:}$ (row-difference). As we will detail in Section 4 , $\|\nabla \mathbf{U}_{ij}\|$ approximates the asymptotic diffusion distance between two nodes i and j. Herein, we focus on the fact that nodes belonging to the same sub-structure (e.g. cluster or community) have similar distances. Back to Figure 1, if i and j belong to the same community, two random walks placed in i and j have similar escape probabilities. Therefore, we build a hierarchy of escape probabilities to characterize the respective reachability of any node wrt a given one. Interestingly, the hierarchy induced by hops is isotropic wrt each node whereas the hierarchy induced by escape probabilities is anisotropic. This latter hierarchy is built by specifying binary projection matrices Π^k which select the pairs of distances that support the creation/update each structural filter $\mathbf{J}^k$.

Exploration by Parallel Jumping. The diffusion pump triggers the creation of $K+1$ structural filters $\{\mathbf{J}^0, \mathbf{J}^1, \ldots, \mathbf{J}^K\}$ derived from their respective jumps J^k. Each filter $\mathbf{J}^k$ has its support $E_k = \{(i_k, j_k)\}$ and its coefficients $c(i_k, j_k) = e^{-d(i_k, i_k)}$. We illustrate this process in Figure 2 over a heterophilic graph. At any epoch, the optimizer creates a distance matrix $\mathcal{D}$ and weighs it: $C = \exp[-\mathcal{D}]$. The result is a weight distribution (middle-left). Each yellow point in the weight distribution belongs to the jump J^k. Consequently, the yellow points denote the edges of the filter support E_k and they are projected in the adjacency matrix below the distribution. The coefficients $c(i_k, j_k)$ of the filter are the heights of the yellow points. Finally, the edges in the graph depicted below the distribution are exactly those of the filter support. Neighbor aggregation wrt this filter $\mathbf{J}^k\mathbf{X}$ will be constrained to that graph and these weights.

The right panel in Figure 2 shows the evolution of the weight distribution (top), some filters (center), and the status of the homophiliation process (bottom). The optimization process is initially dominated by the diffusion pump since random weight distributions are explored first. As a result, scattered nodes with similar labels can be potentially aggregated: we start to implicitly build the hidden graph $G' = (V, E')$. The probability of aggregating distant nodes is leveraged by the fact that, during the first epochs, most of the $K+1$ filters $\mathbf{J}^k$ have a random nature independently of k, the filter order. Escape probabilities are relaxed during this *exploration stage*.

Classification Loss. Each filter, $\mathbf{J}^k$ becomes the aggregator of a naive GNN $\sigma(\mathbf{J}^k\mathbf{X}\mathbf{W}^k)$ which generates an embedding $\mathbf{H}^k$. This embedding is weighted by a learnable parameter α_k and concatenated with the remaining embeddings to feed a classification layer. Therefore, as soon as the structural filters discover interesting bonds for minimizing $\mathcal{L}_c$, the weights $\mathbf{W}^k$ of all the GNNs, the filters' coefficients, and the distance matrix will become more and more stable. At some point in the optimization process, the Dirichlet loss will be stabilized and the exploration stage ends. Later on, the classification loss will refine the almost-homophilic global embedding $\mathbf{H}$. As a result, the embeddings of either border nodes or hidden nodes with the same label are grouped together in the latent space (for instance, see the column of Epoch 100 in Figure 2).

4 Methodological Details

4.1 Network Architecture

DIFFUSION-JUMP GNNS are neural networks $f_\Theta(\mathbf{A}, \mathbf{X}, \ell(B))$ resulting from the optimization problem stated in Eq. 2. The interplay between the Dirichlet loss and the classification loss is described above. In this Section, we give some technical details about the architecture of the network.

Diffusion pump. The pump is responsible for generating and updating the matrix of pairwise diffusion distances $\mathcal{D}$. For the generation, we solve any of the following equivalent problems:

$$\text{Min } \frac{Tr[\mathbf{U}^T \triangle \mathbf{U}^T]}{Tr[\mathbf{U}^T \mathbf{D} \mathbf{U}^T]} \equiv \text{Max } \frac{Tr[\mathbf{U}^T \mathbf{A} \mathbf{U}^T]}{Tr[\mathbf{U}^T \mathbf{D} \mathbf{U}^T]}\,, \tag{3}$$

both s.t. $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, where $\mathbf{U}_{n\times p} = f_\theta(\mathbf{A})$, $n = |V|$. Since $\mathbf{D}$ is the diagonal degree matrix, we have $\triangle := \mathbf{D} - \mathbf{A}$. As a result, the Min problem *approximates* the p smallest nontrivial eigenvectors of the normalized Laplacian $\tilde{\triangle} := \mathbf{D}^{-1/2}\triangle\mathbf{D}^{-1/2} = \mathbf{I} - \tilde{\mathbf{A}}$, where $\tilde{\mathbf{A}} := \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is the normalized adjacency. Equivalently, the Max problem *approximates* the p largest nontrivial eigenvectors of the transition matrix $\mathbf{P} := \mathbf{D}^{-1}\mathbf{A}$. Note that $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{P}}$ have the same eigenvectors and also that if λ is an eigenvalue of $\mathbf{P}$ then $1 - \lambda$ an eigenvalue for $\tilde{\triangle}$. Note also, that we use "*approximates*" instead of "*finds*". This is due to the limitations of Stochastic Gradient Descent (SGD) when solving the Trace-Ratio problems [26][27][28] in Eq. 3. In this regard, we have the following results with practical implications:

Theorem 4.1 (Fiedler Environments)**.** *The SGD solution of the Trace-Ratio Min problem in Eq. 3 can be posed in terms of Min Tr*$[\mathbf{U}^T(\triangle - \rho\mathbf{D})\mathbf{U}]$ *under orthonormality constraints. This leads to* $\triangle\mathbf{U} = \rho^*\mathbf{D}\mathbf{U}$*, i.e. to the orthogonal eigenfunctions of the normalized Laplacian* $\tilde{\triangle}$ *associated with* ρ^**. However,* ρ^* *is not necessarily an eigenvalue of* $\tilde{\triangle}$*, but an approximation of the Fiedler value*

λ_2: $\exists\, \epsilon > 0\ : |\lambda_2 - \rho^*| < \epsilon$. *As a result, the* p *columns* $\mathbf{u}_i$ *of* $\mathbf{U}$ *satisfy:* $\exists\, \delta > 0\ : \|\phi_2 - \mathbf{u}_i\| < \delta$, *where* ϕ_2 *denotes the Fiedler vector. Then, we obtain what we call a* Fiedler environment.

Corollary 4.2 (Asymptotic Diffusion Distances). *The norm of* $\nabla \mathbf{U}_{ij} := \mathbf{U}_{i:} - \mathbf{U}_{j:}$ *(row-difference) is proportional to the approximate commute time between nodes i and j, which is* $d(i,j)^{t\rightarrow\infty}$, *where* $d(i,j) := \|\nabla \mathbf{U}_{ij}\|$. *Therefore, the matrix* $\mathcal{D}$ *relies on Euclidean distances.*

We proof both results in *Appendix A: Formal Results with Practical Implications*. We also give there practical evidence of the need of solving Trace-Ratio problems in an SGD context, instead of solving the original Trace problem in Eq. 2. We also justify the convenience of conditioning $\mathbf{U}$ to $\mathbf{A}$, $\mathbf{U} = f_\theta(\mathbf{A})$. Actually, this setting is inspired in how the LINKX method [29] exploits the graph topology.

Jumps and Filters. The bank of learnable structural filters $\{\mathbf{J}^0, \mathbf{J}^1, \ldots, \mathbf{J}^K\}$ is the core of the high-order exploration. Each filter $\mathbf{J}^k$ has a *support* $E_k = \{(i_k, j_k) \in V \times V : i_k, j_k \in J^k\}$ with the pairs of nodes (edges of the filter) belonging to the jump J^k. In Eq. 2, this is implicitly defined with the expression $\mathbf{J}^k = \Pi^k \cdot \exp(-\mathcal{D})$, where Π^k is a $\{0,1\}^{n\times n}$ projection matrix defined, for $k > 0$, as follows:

$$\Pi^k(i,j) = \left\{ \begin{array}{cc} 1 - \Pi^{k-1}(i,j) & \text{if } j \in \text{Idx}[\text{top}k^{-1}(i)] \\ 0 & \text{otherwise} \end{array} \right. \tag{4}$$

with $\Pi^0 = \mathbf{I}$ and top$k^{-1}(i) = \{d(i,j_1), \ldots, d(i,j_k)\}$, where $d(i,j_l) \leq d(i,j_{l+1})$ for $j_l, j_{l+1} \in V$ and $l = 1, 2, \ldots, k-1$. Then, Idx[top$k^{-1}(i)$] are the sorted positions of the distances wrt the node i, i.e. the *distance ranks*. In this way, the product $\mathbf{J}^k = \Pi^k \cdot \exp(-\mathcal{D})$ is derivable wrt $\mathcal{D}$ as in [30]. Alternatively, we could also rely on the topK network [31].

Individual GNNs. Each structural filter $\mathbf{J}^k$ feeds a vanilla GNN which obtains a partial embedding $\mathbf{H}^k = \sigma(\mathbf{J}^k\mathbf{X}\mathbf{W}^k)$. The GNN also receives the $n \times F$ matrix of node features $\mathbf{X}$, and $\mathbf{x}_i$ denotes the transpose of the $i-$th row of $\mathbf{X}$. Since $\mathbf{J}^k = \Pi^k \cdot \exp(-\mathcal{D})$, then, for a given node i, its aggregation is given by $\mathbf{x}_i = \sum_j e^{-d(i,j)}\mathbf{x}_j$ instead of being $\mathbf{x}_i = \mathbf{P}^k\mathbf{x}_i$ as in MixHop [15] or $\mathbf{x}_i = \left(\sum_k \beta_k \mathbf{P}^k\right)\mathbf{x}_j$ as in Simple Graph Convolution (SGG) [19]. As the asymptotic diffusion distances are approximations of commute times (Theorem 4.1), our aggregation works as a kernel depending on learnable Euclidean distances.

Combining GNNs. Each partial embedding $\mathbf{H}^k = \sigma(\mathbf{J}^k\mathbf{X}\mathbf{W}^k)$ is weighted by a learnable parameter α_k, where all the α_k form a convex combination. Then, we concatenate all the weighted embeddings to form the global embedding $\mathbf{H} := \|_{k=1}^K \alpha_k \mathbf{H}^k = \|_{k=1}^K \alpha_k \sigma\left(\mathbf{J}^k\mathbf{X}\mathbf{W}^k\right)$. Since $\mathbf{H}$ feeds an MLP in order to minimize the classification loss $\mathcal{L}_c$ as in MixHop, the global embedding tends to retain the best partial embeddings for each node.

Homophilic Branch. One limitation of our method is that setting a small value for the hyperparameter K is not enough to deal with homophilic graphs. For this reason, we have added an extra GNN (the homophilic branch) that works as follows: $\mathbf{H}^{HB} = \sigma(\mathbf{A}\mathbf{X}\mathbf{W}^{HB})$. Therefore we concatenate $\mathbf{H} := \mathbf{H}|\alpha_{HB}\mathbf{H}^{HB}$, where $\sum_{k=0}^K \alpha_k + \alpha_{HB} = 1$. See the optimal learned coefficients in Figure 3.

4.2 Inspiring Methods

We conclude this Section by reviewing some **links with very inspiring methods** in the literature. For instance, our Dirichlet formulation is inspired by classical graph-based semi-supervised methods. In particular, the work in [32] addresses the problem of propagating known labels $\ell(B)$ to unknown nodes $u \in U$. Let $\mathbf{Y}$ be a $n \times c$ matrix where $\mathbf{Y}(i,j) = 1$ means that node $i \in B$ has label j and $\mathbf{Y}(i,j) = 0$ otherwise. Then, we have the following result:

Theorem 4.3 (Dichilet label propagation [32]). *The optimal label of each node* i *is given by* $\mathbf{y}(i) = \arg\max_{j\leq c} \mathbf{F}(i,j)$, *where* $\mathbf{F} = \beta\left(\mathbf{I} - \alpha\mathbf{P}\right)^{-1}\mathbf{Y}$, *being* $\mathbf{P}$ *the transition matrix and* $\alpha + \beta = 1$. *In addition,* $\mathbf{F}$ *minimizes* $\frac{1}{2}\left(Tr[\mathbf{F}^T \triangle \mathbf{F}] + \mu\sum_i \|\mathbf{F}_{i:} - \mathbf{Y}_{i:}\|^2\right)$, *where* $\mu > 0$ *is a regularization parameter satisfying* $\alpha = 1/(1+\mu)$.

Consequently, the diffusion pump in our model is governed by a similar equation: Eq. 2. We sketch the proof of the above theorem and its relationship with absorbing random walks [33] and semi-supervised image segmentation [34] in *Appendix A*.

Finally, another important source of inspiration was the design of escape probabilities in terms of diffusion equations. Actually, there is a substantial body of theory linking spectral clustering, random walks, diffusion distances, and metastable states [35][25][24] to be analyzed also in the same appendix. Herein, we only highlight the following result:

Theorem 4.4 ([24])**.** *Given a probability function in Boltzmann form $p(\mathbf{x}) = e^{-U(\mathbf{x})}$ in a given latent space $\mathbf{X}$, the random walk with transition matrix $\mathbf{P}$ converges to the stochastic differential equation $\dot{\mathbf{x}}(t) = -\nabla U(\mathbf{x}) + \sqrt{2}\dot{\mathbf{w}}(t)$, where $\mathbf{w}$ denotes Brownian motion. Also, the potential time scales describing the expected time of passage between clusters rely on the potential function $U(\mathbf{x})$.*

5 Experiments and Discussion

Experimental settings.Table 1 presents the results obtained by each model on the standard small-medium datasets [36] [37] [38]. To ensure consistency, we used the same 10 random splits (48%/32%/20%) provided by [37], along with the best configuration for each model. We place the code at `https://anonymous.4open.science/r/Diffusion-Jump-GNNs-8EE2/`. The above configurations were extracted from Table 1 [39] and Tables 3, 10, and 11 in [29]. Overall, our model outperformed all others or achieved a close second place, demonstrating strong competitiveness. We assessed the degree of *structural heterophily* using our metric $\mathcal{R}$. Specifically, we examined two heterophilic regimes: $\mathcal{R} < 8$ indicating low structural heterophily, and $\mathcal{R} \geq 8$ indicating high structural heterophily.

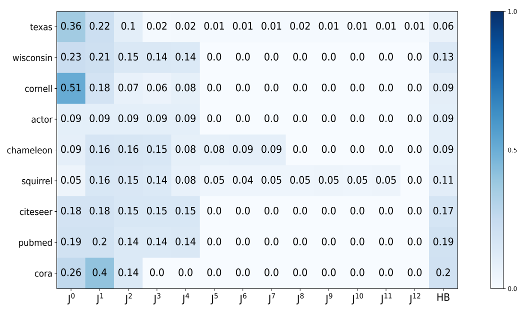


Figure 3: Optimal attention for each jump.

Table 1: Node-classification accuracies. Top three models are coloured by **First**, **Second**, **Third**.

	Texas	Wisconsin	Cornell	Actor	Squirrel	Chameleon	Citeseer	Pubmed	Cora
Hom level	0.11	0.21	0.30	0.22	0.22	0.23	0.74	0.80	0.81
$\mathcal{R}$	**18.37**	**6.90**	**6.03**	**209.58**	**20.62**	**8.30**	**5.78**	**7.64**	**7.36**
# Nodes	183	251	183	7,600	5,201	2,277	3,327	19,717	2,708
# Edges	295	466	280	26,752	198,493	31,421	4,676	44,324	5,278
# Classes	5	5	5	5	5	5	7	3	6
GGCN [40]	84.86 ± 4.55	86.86 ± 3.29	85.68 ± 6.63	37.54 ± 1.56	55.17 ± 1.58	77.14 ± 1.84	77.14 ± 1.45	89.15 ± 0.37	87.95 ± 1.05
GPRGNN [41]	78.38 ± 4.36	82.94 ± 4.21	80.27 ± 8.11	34.63 ± 1.22	31.61 ± 1.24	46.58 ± 1.71	77.13 ± 1.67	87.54 ± 0.38	87.95 ± 1.18
H2GCN [4]	84.86 ± 7.23	87.65 ± 4.89	82.70 ± 5.28	35.70 ± 1.00	36.48 ± 1.86	60.11 ± 2.15	77.11 ± 1.57	89.49 ± 0.38	87.87 ± 1.20
GCNII [42]	77.57 ± 3.83	80.39 ± 3.40	77.86 ± 3.79	37.44 ± 1.30	38.47 ± 1.58	63.86 ± 3.04	77.33 ± 1.48	90.15 ± 0.43	88.37 ± 1.25
Geom-GCN [37]	66.76 ± 2.72	64.51 ± 3.66	60.54 ± 3.67	31.59 ± 1.15	38.15 ± 0.92	60.00 ± 2.81	78.02 ± 1.15	89.95 ± 0.47	85.35 ± 1.57
PairNorm [43]	60.27 ± 4.34	48.43 ± 6.14	58.92 ± 3.15	27.40 ± 1.24	50.44 ± 2.04	62.74 ± 2.82	73.59 ± 1.47	87.53 ± 0.44	85.79 ± 1.01
GraphSAGE [2]	82.43 ± 6.14	81.18 ± 5.56	75.95 ± 5.01	34.23 ± 0.99	41.61 ± 0.74	58.73 ± 1.68	76.04 ± 1.30	88.45 ± 0.50	86.90 ± 1.04
GCN [1]	55.14 ± 5.16	51.76 ± 3.06	60.54 ± 5.30	27.32 ± 1.10	53.43 ± 2.01	64.82 ± 2.24	76.50 ± 1.36	88.42 ± 0.50	86.98 ± 1.27
GAT[3]	52.16 ± 6.63	49.41 ± 4.09	61.89 ± 5.05	27.44 ± 0.89	40.72 ± 1.55	60.26 ± 2.50	76.55 ± 1.23	87.30 ± 1.10	86.33 ± 0.48
MLP	80.81 ± 4.75	85.29 ± 6.40	81.89 ± 6.40	36.53 ± 0.70	28.77 ± 1.56	46.21 ± 2.99	74.02 ± 1.90	75.69 ± 2.00	87.16 ± 0.37
CGNN[44]	71.35 ± 4.05	74.31 ± 7.26	66.22 ± 7.69	35.95 ± 0.86	29.24 ± 1.09	46.89 ± 1.66	76.91 ± 1.81	87.70 ± 0.49	87.10 ± 1.35
MixHop [15]	77.84 ± 7.73	75.88 ± 4.90	73.51 ± 6.34	32.22 ± 2.34	43.80 ± 1.48	60.50 ± 2.53	76.26 ± 1.33	85.31 ± 0.61	87.61 ± 0.85
FSGNN(8-hop) [16]	87.30 ± 5.29	87.84 ± 3.37	87.84 ± 6.19	35.75 ± 0.96	74.10 ± 1.89	78.27 ± 1.28	77.40 ± 1.90	77.40 ± 1.93	87.93 ± 1.00
GRAFF [39]	88.38 ± 4.53	88.83 ± 3.29	84.05 ± 6.10	37.11 ± 1.08	58.72 ± 0.84	71.08 ± 1.75	77.30 ± 1.85	90.04 ± 0.41	88.01 ± 1.03
LINKX [29]	74.60 ± 8.37	75.49 ± 5.72	77.84 ± 5.81	36.10 ± 1.55	61.81 ± 1.80	68.42 ± 1.38	73.19 ± 0.99	87.86 ± 0.77	84.64 ± 1.13
ACMII-GCN++ [45]	88.38 ± 3.43	88.43 ± 3.66	86.49 ± 6.73	37.09 ± 1.32	67.40 ± 2.21	74.76 ± 2.20	77.12 ± 1.58	89.71 ± 0.48	88.25 ± 0.96
Ordered GNN [14]	86.22 ± 4.12	88.04 ± 3.63	87.03 ± 4.73	37.99 ± 1.00	62.44 ± 1.96	72.28 ± 2.29	77.31 ± 1.73	90.15 ± 0.38	88.37 ± 0.75
ASGC [19]	85.14 ± 3.06	86.06 ± 3.75	86.22 ± 3.58	36.33 ± 0.79	58.38 ± 1.08	73.16 ± 1.07	66.86 ± 0.86	78.72 ± 0.88	77.52 ± 1.61
DJ-GNN	92.43 ± 3.15	92.54 ± 3.70	87.03 ± 1.62	36.93 ± 0.84	73.48 ± 1.59	80.48 ± 1.46	77.50 ± 1.33	89.19 ± 0.32	88.43 ± 0.91

Low Structural Heterophily. For these datasets, we list their optimal number of jumps (hyperparameter K): Wisconsin ($K = 5$), Cornell ($K = 5$), Citeseer ($K = 5$), Pubmed ($K = 5$) and Cora ($K = 5$)) we find that a few jumps are enough for achieving or improving the SOTA. However, not all jumps are equally important. For instance, Wisconsin and Cornell rely mostly on the first two jumps (see Figure 3), while the remaining datasets rely on the *homophilic branch* (no jump). Actually, Citeseer, Pubmed, and Cora are the datasets with the smallest edge heterophily (Hom Level). In addition, we are the second best method only in Cornell (87.03 (ours) vs 87.84 (FSGNN(8-hop)) and we are very competitive in Citeseer (77.50 (ours) vs 78.02 (Geom-GCN)). In the first case, FSGNN(8-hop) uses a fully supervised split (60%/20%/20%) whereas we use the more severe semi-supervised split. For Citeseer, we note that the Geom-GCN method relies on the geometry of the latent space. In this regard, CITESEER is the dataset with the lowest structural heterophily ($\mathcal{R} = 5.78$), i.e. the geometry of the latent space is a fair representation of the topology of the graph. As a result, adding jumps may complicate that geometry: actually, the most important branches are $\mathbf{J}^0$ and the *homophilic branch* (no jump). Finally, in PUBMED, our method is slightly improved by ORDERED GNN (89.19 vs 90.15).

High Structural Heterophily. For these datasets [46] [47], we also list their optimal number of jumps (hyperparameter K): TEXAS ($K = 20$), SQUIRREL ($K = 8$), CHAMELEON ($K = 12$) and ACTOR ($K = 3$). Our best result is for TEXAS ($\mathcal{R} = 18.37$), where we significantly improve the SOTA (92.43 (ours) vs 88.38 (ACMII-GCN++, which is a multi-channel GCN with adaptive channel mixing)). In SQUIRREL, we are slightly outperformed by FSGNN(8-HOP) (73.48 vs 74.10) due, again to their use of a full supervised split (60%/20%/20%). We are very competitive in this dataset because the SQUIRREL graph is very dense and we only need $K = 8$ jumps to achieve good results. We are also the best model in CHAMELEON (whose structural heterophily is the smallest in this set): we obtain 80.48 vs the second-best model FSGNN(8-HOP) that achieves 78.27 (again with a split of 60%/20%/20%).

Parallel (Shallow) vs Sequential (Deep). Our method is Parallel (multi-branch shallow GNN) and its performance is the best or it is very competitive in small-medium datasets. There is one exception. In the PUBMED dataset, where we obtain 89.15, we are slightly outperformed by ORDERED GNN with only 5 layers (90.15) because this dataset is very homophilic. This also happens with GCNII which explores 2 to 2^6 layers. We can conclude that deep methods have a good performance in homophilic datasets but such a performance decays significantly in heterophilic ones.

Table 2: Node-classification accuracies in large graphs. Top three models are coloured by **First**, **Second**, **Third**.

	PENN94	ARXIV-YEAR	OGBN-ARXIV
HOM LEVEL	**0.47**	**0.21**	**0.66**
# NODES	41,554	169,343	169,343
# EDGES	1,362,229	1,166,243	1,166,243
# CLASSES	5	5	40
MLP	73.61 ± 0.40	36.70 ± 0.21	55.91 ± 0.15
GCN	82.47 ± 0.27	46.02 ± 0.26	59.61 ± 0.23
GAT	81.53 ± 0.55	46.05 ± 0.51	60.27 ± 0.21
MIXHOP	83.47 ± 0.71	51.81 ± 0.17	OOM
LINKX	84.71 ± 0.52	56.00 ± 1.34	55.31 ± 0.81
DJ-GNN	84.84 ± 0.34	49.21 ± 0.20	63.23 ± 0.12

We have also tested our model in **Very Large Graphs** (see Table 2). In this regard, we note that the memory requirements of our method $- O(n^2)$, where n is the number of nodes $-$, force us to decouple the diffusion pump from the jump exploration. We first learn the matrix of pairwise diffusion distances in an unsupervised way. Later, we use it in a static way to minimize the classification loss. Despite that limitation, we obtain a very competitive performance both for PENN94 (84.84 vs 84.71 with LINKX) and OGBN-ARXIV (63.23 vs 60.27 with GAT). However, our performance decreases in ARXIV-YEAR which is more heterophilic than the others: memory limitations force us to use only $K = 3$ hops for the three datasets.

Finally, we extend our experimental results in *Appendix B: SBM Analysis* and *Appendix C: Experimental and Computational Details*.

6 Conclusions and Future Work

In this paper, we propose DIFFUSION-JUMP GNNS, a multi-branch GNN architecture that addresses the heterophily issue from a structural perspective. Firstly, we define node-classification in terms of a Dirichlet problem. This allows us to define a new measure of heterophily: structural heterophily. Having this measure in mind, we formulate a loss function that governs the interplay between the two main components of our architecture: the **diffusion pump** (which generates diffusion distances) and the **parallel jumps** (which drive the exploration of links between nodes with similar labels). The most important contribution of our model is that the diffusion distances, and consequently the jumps and the structural filters derived from them, are fully learnable. Our experiments show that our model outperforms the SOTA or it is very competitive. Finally, our future work includes: a) scalability, in terms of memory and b) automatic jump selection.

Broader Impact. This research contributes to improving the reliability of graph learners in the heterophilic regime. Besides the well-known social benefits concerning the detection of malicious nodes in social networks, we consider the application of our learnable structural filters to fairness.

References

[1] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[2] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 2017.

[3] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

[4] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

[5] Meng Liu, Zhengyang Wang, and Shuiwang Ji. Non-local graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10270–10276, 2022.

[6] Jiong Zhu, Ryan A. Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K. Ahmed, and Danai Koutra. Graph neural networks with heterophily, 2021.

[7] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1725–1735. PMLR, 13–18 Jul 2020.

[8] Wendong Bi, Lun Du, Qiang Fu, Yanlin Wang, Shi Han, and Dongmei Zhang. Make heterophily graphs better fit gnn: A graph rewiring approach. *arXiv preprint arXiv:2209.08264*, 2022.

[9] Adrián Arnaiz-Rodríguez, Ahmed Begga, Francisco Escolano, and Nuria Oliver. DiffWire: Inductive Graph Rewiring via the Lovász Bound. In *The First Learning on Graphs Conference*. PMLR, 2022.

[10] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022.

[11] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio', and Michael Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology, 2023.

[12] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In Xingquan Zhu, Sanjay Ranka, My T. Thai, Takashi Washio, and Xindong Wu, editors, *IEEE International Conference on Data Mining, ICDM 2022, Orlando, FL, USA, November 28 - Dec. 1, 2022*, pages 1287–1292. IEEE, 2022.

[13] Ralph Abboud, Radoslav Dimitrov, and Ismail Ilkan Ceylan. Shortest path networks for graph property prediction. In *The First Learning on Graphs Conference*, 2022.

[14] Yunchong Song, Chenghu Zhou, Xinbing Wang, and Zhouhan Lin. Ordered GNN: Ordering message passing to deal with heterophily and over-smoothing. In *The Eleventh International Conference on Learning Representations*, 2023.

[15] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR, 2019.

[16] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Improving graph neural networks with simple architecture design, 2021.

[17] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Not all neighbors are friendly: Learning to choose hop features to improve node classification. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 4334–4338, New York, NY, USA, 2022. Association for Computing Machinery.

[18] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Benjamin Chamberlain, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020.

[19] Sudhanshu Chanpuriya and Cameron N Musco. Simplified graph convolution with heterophily. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[20] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

[21] Wilhelm Huisinga, Sean Meyn, and Christof Schütte. Phase transitions and metastability in markovian and molecular systems. *The Annals of Applied Probability*, 14(1):419–458, 2004.

[22] B. J. Matkowsky and Z. Schuss. Eigenvalues of the fokker–planck operator and the approach to equilibrium for diffusions in potential fields. *SIAM Journal on Applied Mathematics*, 40(2):242–254, 1981.

[23] Peter Hänggi, Peter Talkner, and Michal Borkovec. Reaction-rate theory: fifty years after kramers. *Rev. Mod. Phys.*, 62:251–341, Apr 1990.

[24] Boaz Nadler and Meirav Galun. Fundamental limitations of spectral clustering. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, page 1017–1024, Cambridge, MA, USA, 2006. MIT Press.

[25] Boaz Nadler, Stéphane Lafon, Ronald R. Coifman, and Ioannis G. Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, page 955–962, Cambridge, MA, USA, 2005. MIT Press.

[26] Huan Wang, Shuicheng Yan, Dong Xu, Xiaoou Tang, and Thomas Huang. Trace ratio vs. ratio trace for dimensionality reduction. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[27] Yangqing Jia, Feiping Nie, and Changshui Zhang. Trace ratio problem revisited. *IEEE Transactions on Neural Networks*, 20(4):729–735, 2009.

[28] T. T. Ngo, M. Bellalij, and Y. Saad. The trace ratio optimization problem. *SIAM Review*, 54(3):545–569, 2012.

[29] Derek Lim, Felix Matthew Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Prasad Bhalerao, and Ser-Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[30] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pages 2083–2092. PMLR, 2019.

[31] Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable top-k with optimal transport. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 20520–20531. Curran Associates, Inc., 2020.

[32] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003.

[33] Peter G. Doyle and J. Laurie Snell. Random walks and electric networks, 2000.

[34] L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.

[35] Marina Meilă and Jianbo Shi. A random walks view of spectral segmentation. In Thomas S. Richardson and Tommi S. Jaakkola, editors, *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, volume R3 of *Proceedings of Machine Learning Research*, pages 203–208. PMLR, 04–07 Jan 2001. Reissued by PMLR on 31 March 2021.

[36] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93, Sep. 2008.

[37] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *CoRR*, abs/2002.05287, 2020.

[38] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *CoRR*, abs/1909.13021, 2019.

[39] Francesco Di Giovanni, James Rowbottom, Benjamin P. Chamberlain, Thomas Markovich, and Michael M. Bronstein. Graph neural networks as gradient flows: understanding graph convolutions via energy, 2022.

[40] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. *CoRR*, abs/2102.06462, 2021.

[41] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

[42] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. *CoRR*, abs/2007.02133, 2020.

[43] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns, 2019.

[44] Takenori Yamamoto. Crystal graph neural networks for data mining in materials science. Technical report, Research Institute for Mathematical and Computational Sciences, LLC, Yokohama, Japan, 2019. https://github.com/Tony-Y/cgnn.

[45] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks, 2022.

[46] Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. Social structure of facebook networks. *CoRR*, abs/1102.2166, 2011.

[47] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *CoRR*, abs/2005.00687, 2020.

[48] Alan Edelman, Tomás A. Arias, and Steven T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[49] Zhiqiang Xu, Xin Cao, and Xin Gao. Convergence analysis of gradient descent for eigenvector computation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 2933–2939. AAAI Press, 2018.

[50] Jing An, Lexing Ying, and Yuhua Zhu. Why resampling outperforms reweighting for correcting sampling bias with stochastic gradients. In *International Conference on Learning Representations*, 2021.

[51] Huaijun Qiu and Edwin R. Hancock. Clustering and embedding using commute times. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1873–1890, 2007.

[52] A. K. Chandra, P. Raghavan, W. L. Ruzzo, and R. Smolensky. The electrical resistance of a graph captures its commute and cover times. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 574–586, New York, NY, USA, 1989. Association for Computing Machinery.

[53] Boaz Nadler, Stéphane Lafon, Ronald R. Coifman, and Ioannis G. Kevrekidis. Diffusion maps, spectral clustering and reaction coordinates of dynamical systems. *Applied and Computational Harmonic Analysis*, 21(1):113–127, 2006. Special Issue: Diffusion Maps and Wavelets.

[54] Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Lió. Directional graph networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 748–758. PMLR, 18–24 Jul 2021.

[55] Sudhanshu Chanpuriya and Cameron N Musco. Simplified graph convolution with heterophily. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[56] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019.

[57] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[58] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

7 Appendix

7.1 Appendix A: Formal Results with Practical Implications

For the sake of clarity, in this appendix we develop the key concepts of the theorems stated in the paper instead of providing detailed proofs. Our emphasis here is on the practical implications of each result. For more details, we refer the reader to the cited papers.

The Trace Ratio Problem. Min $\mathbf{U}^T\triangle\mathbf{U}$ is achieved by $\mathbf{V}_{n\times p}$ whose p columns are given by the eigenvectors of the p smallest eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_p$ of the Laplacian $\triangle$. Then, $\text{Tr}[\mathbf{V}^T\triangle\mathbf{V}] = \lambda_2 + \ldots + \lambda_p$, when the graph $G = (V, E)$ with adjacency matrix $\mathbf{A}$ is connected. However, $\mathbf{V}$ does not necessarily minimize $\text{Tr}[\mathbf{V}^T\mathbf{D}\mathbf{V}]$. As a result, we have that

$$\rho^* := \text{Min}_{\mathbf{U}^T\mathbf{U}=\mathbf{I}}\frac{\text{Tr}[\mathbf{U}^T\triangle\mathbf{U}]}{\text{Tr}[\mathbf{U}^T\mathbf{D}\mathbf{U}]} \leq \frac{\text{Tr}[\mathbf{V}^T\triangle\mathbf{V}]}{\text{Tr}[\mathbf{V}^T\mathbf{D}\mathbf{V}]} \leq \frac{\lambda_2 + \ldots + \lambda_p}{d_1 + \ldots + d_p}\ , \tag{5}$$

where $d_1 \leq d_2 \leq \ldots \leq d_n$ are the sorted degrees. As a result, we have the following bounds:

$$\frac{\lambda_2 + \ldots + \lambda_p}{d_{p+1} + \ldots + d_n} \leq \rho^* \leq \frac{\lambda_2 + \ldots + \lambda_p}{d_1 + \ldots + d_p}\ . \tag{6}$$

The definition of ρ^* plays a key role in the original trace-ratio optimization. Following [28], such a problem is formulated in scalar terms, i.e. in terms of finding

$$\rho^* = \arg\min_{\mathbf{U}^T\mathbf{U}=\mathbf{I}} f(\rho) := \text{Tr}[\mathbf{U}^T\triangle\mathbf{U}] - \rho\text{Tr}[\mathbf{U}^T\mathbf{D}\mathbf{U}]\ . \tag{7}$$

Actually, for ρ^* we have have that

$$\text{Min}_{\mathbf{U}^T\mathbf{U}=\mathbf{I}}\text{Tr}[\mathbf{U}^T(\triangle - \rho^*\mathbf{D})\mathbf{U}] = 0\ . \tag{8}$$

Therefore, the trace-ratio problem can be solved by alternating two updating steps:

- $\mathbf{U}$: Given ρ, apply the Lanczos method to obtain the p largest eigenvalues of the transition matrix $\mathbf{P} - \rho\mathbf{D}$ (the smallest of $\triangle - \rho\mathbf{D}$) and their associated eigenvectors $\mathbf{U}$.
- ρ : Given the current eigenvectors $\mathbf{U}$, update $\rho = \frac{\text{Tr}[\mathbf{U}^T\triangle\mathbf{U}]}{\text{Tr}[\mathbf{U}^T\mathbf{D}\mathbf{U}]}$

In the above process, the update of $\mathbf{U}$ ensures the orthogonality constraint.

The Trace Ratio and SGD. However, solving the trace-ratio problem through gradient descent drives us to a different solution from the eigenvectors of $\triangle - \rho^*\mathbf{D}$. For instance, consider the Dirichlet loss $\mathcal{L}_D = \frac{\text{Tr}[\mathbf{U}^T\triangle\mathbf{U}]}{\text{Tr}[\mathbf{U}^T\mathbf{D}\mathbf{U}]}$. Then, its gradient (supposing that the orthogonality is enforced by a complementary loss) is given by:

$$\nabla\mathcal{L}_D := \frac{2\triangle\mathbf{U} - 2\rho\mathbf{D}\mathbf{U}}{\text{Tr}[\mathbf{U}^T\mathbf{D}\mathbf{U}]}\ . \tag{9}$$

Therefore, $\nabla\mathcal{L}_D = 0$ implies $\triangle\mathbf{U} = \rho^*\mathbf{D}\mathbf{U}$, where $\rho^* \to 0$ is the asymptotic value of the trace ratio. As a result, we have that the optimal $\mathbf{U}$ satisfy $\tilde{\triangle}\mathbf{U} = \rho^*\mathbf{U}$, i.e. the gradient descent converges to the (orthonormal) functions of the normalized Laplacian $\tilde{\triangle}$ associated with the value ρ^*. However, as ρ^* is not necessarily an eigenvalue of $\tilde{\triangle}$, but it is close to the Fiedler value λ_2, we denote $\mathbf{U}$ as a **Fiedler environment**. It is an environment since the p columns $\mathbf{u}_{:j}$ are mutually orthonormal and close to the Fiedler vector ϕ_2 insofar their Dirichlet energies $\rho^*_j = {\mathbf{u}_{:j}}^T\tilde{\triangle}\mathbf{u}_{:j}$ satisfy $|\lambda_2 - \rho^*_j| < \epsilon$ with $\epsilon \to 0$ (**Theorem** 4.1).

In our experiments, we have chosen the trace-ratio formulation because:

a) It leads an **implicit normalization** of the gradient $\nabla\mathcal{L}_D$, namely $\text{Tr}[\mathbf{U}^T\mathbf{D}\mathbf{U}]$.

b) The **gradient is more structured** when we apply the constrain $\mathbf{U} = f_\theta(\mathbf{A})$, where $\mathbf{A}$ is the adjacency matrix.

Regarding **a)**, our implicit normalization alleviates the problem of landing in local minima due to the orthonormalization constraint (that we also enforce in the global loss). As noted [48], constraints of the form $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ define a Riemannian manifold and the trace problem s.t. them is not geodesically convex. In [49] this is addressed by introducing a Riemannian gradient and retraction normalization.

However, our main gain in performance is achieved when we address **b)** via the joint effect of normalization and $\mathbf{U} = f_\theta(\mathbf{A})$. In our preliminary experiments, we compared the gradient when applying the constraint $\mathbf{U} = f_\theta(\mathbf{A})$ vs the one when doing only $\mathbf{U} = f_\theta(\mathbf{I})$. Discarding the biases, and the non-linearities in both cases, we have $\mathbf{U} = \mathbf{AW}$ vs $\mathbf{U} = \mathbf{W}$. For simplicity, we consider the gradient wrt a single column, i.e. we analyze $\mathbf{u} = \mathbf{Aw}$ vs $\mathbf{u} = \mathbf{w}$

$$\nabla\mathcal{L}'_{D,\theta} := \frac{2(\triangle - \rho\mathbf{D})(\mathbf{Aw})}{\text{Tr}[(\mathbf{Aw})^T\mathbf{D}(\mathbf{Aw})]} \quad \text{vs} \quad \nabla\mathcal{L}_{D,\theta} := \frac{2(\triangle - \rho\mathbf{D})\mathbf{w}}{\text{Tr}[\mathbf{w}^T\mathbf{Dw}]} \ . \tag{10}$$

Given a random initialization of $\mathbf{w}$, this vector plays the role of a random projector of the rows in $\mathbf{A}$. Following, the Johnson-Lindenstrauss Lemma, $\hat{\mathbf{w}} = \mathbf{Aw}$ tends to replicate the structure of the adjacency. Actually, if the entries of $\mathbf{w}_i \sim \mathcal{N}(0, 1)$ then, those of the projection satisfy $\hat{\mathbf{w}}_i \sim \mathcal{N}(0, d_i^2)$, where d_i is the degree of node i. As a result, if we have c well-defined communities in the graph $G = (V, E)$ with adjacency matrix $\mathbf{Aw}$, then the projection $\hat{\mathbf{w}}$ is near piecewise constant (actually the norm of the $i-$th row is preserved: $\|\mathbf{A}_{i:}\| \approx \|\hat{\mathbf{w}}\|$). As a result, the projection $\hat{\mathbf{w}}$ is more structured than $\mathbf{w}$ and this is propagated and even amplified during gradient descent. In addition, the normalization of $\nabla\mathcal{L}'_{D,\theta}$ is stronger than that of $\nabla\mathcal{L}_{D,\theta}$.

Overall, when evaluating the performance in SQUIRREL and CHAMELEON using only $\mathbf{U} = f_\theta(\mathbf{I})$ (i.e. using $\nabla\mathcal{L}_{D,\theta}$) we only obtain 41.38 ± 2.98 and 58.48 ± 4.69. However, using $\mathbf{U} = f_\theta(\mathbf{A})$ (gradient $\nabla\mathcal{L}'_{D,\theta}$) leads to 73.48 ± 1.59 and 80.48 ± 1.46 respectively.

Finally, a detailed impact of the two above formulations in the variance of the SGD as in [50] is beyond the scope of this paper.

Assymptotic Diffusions. As explained above, optimizing the Dirichlet loss leads to Fiedler environments, i.e. the rows $\mathbf{U}_{i:}$ contain the p nearly orthogonal eigenvectors with eigenvalues $\gamma_1 = 1 > \gamma_2 \geq \ldots \geq \gamma_p$. Following [25], the *diffusion distance* $D_j(i,j)$ at time t between two nodes i and j is defined spectrally as:

$$D_t^2(i,j) := \sum_k \frac{1}{\pi(k)}\left(p(k,t|i) - p(k,t|j)\right)^2 = \Gamma^{2t}\|\mathbf{U}^*_{i:} - \mathbf{U}^*_{j:}\|^2 \ , \tag{11}$$

where $\pi(k) := d_k/vol(G)$ are the components from the stationary probability distribution $\lim_{t\to\infty} p(j,t|i) = \pi$, i.e. the eigenvector $\mathbf{U}^*_{:1}$ corresponding to $\gamma_1 = 1$. In the above equation, $\mathbf{U}^*$ denote the true eigenvectors of the transition matrix $\mathbf{P}$, and $\Gamma := \text{diag}(\gamma_1, \gamma_2, \ldots, \gamma_p)$ is the diagonal matrix with the corresponding eigenvalues. Thus, Eq. 11 can be explained in the following terms:

a) $D_t^2(i,j)$ compare the probabilities that two random walks (one starting in i and another one in j) reach any other node k in time t.

b) The spectral interpretation relies on the spectral theorem applied to the transition matrix $\mathbf{P} = \mathbf{U}^*\Lambda\mathbf{U}^{*T}$. As a result, $\mathbf{P}^t = \mathbf{U}^*\Gamma^t\mathbf{U}^{*T} = \sum_{r=1}^n \gamma_r^t \mathbf{U}^*_{:r}\mathbf{U}^{*\,T}_{:r}$, with $n = |V|$.

However, since determining what is the correct diffusion time is very hard (it is usually a hyperparameter in some GNNs), we are interested in the asymptotic diffusion distance $D^2_{t\to\infty}$. Qiu and Hancock [51] determined that

$$\sum_{t=0}^{\infty} D_t^2(i,j) = \sum_{r=2}^{n} \frac{1}{1-\gamma_r}\left(\mathbf{U}^*_{ri} - \mathbf{U}^*_{rj}\right)^2 \tag{12}$$

i.e. eigenvalues $\{1-\gamma_r\}$ of the normalized Laplacian $\tilde{\triangle}$ are used instead of those of $\mathbf{P}$. Actually, the right side of the above equation is the well-known *commute times* [52] distance CT(i,j). Note that such a distance is dominated by the Fiedler value and vector: $\lambda_2 = (1-\gamma_2)$ and $\mathbf{U}^*_{:2}$, respectively. This fact simplifies the interpretation of our approximate diffusion distance as follows:

a) Our approximated eigenvectors, contained in the p columns of $\mathbf{U}$ have eigenvalues (Dirichlet energies) ρ_r^* close to ρ^* (the optimal trace ratio achieved by the Dirichlet loss).

b) Theorethically, we have that the smallest $\rho^*_{r_{\min}}$ satisfies $\lambda_2 \leq \rho^*_{r_{\min}}$. Therefore, if we order ρ^*_r in ascending order, then we obtain

$$\sum_{t=0}^{\infty} D_t^2(i,j) \approx \sum_{r=1}^{p} \frac{1}{\rho^*_r} \left(\mathbf{U}_{ri} - \mathbf{U}_{rj}\right)^2 \approx \text{CT}(i,j) \ . \tag{13}$$

c) However, in the heterophilic regime (where the labels break the structure) we usually have $\lambda_2 \ll \rho^*_{r_{\min}}$. See for instance the Fiedler environments obtained for SBMs in Figure 4 and the discussion below (*Appendix B*). As a result, in practice we have

$$d^{t\rightarrow\infty}(i,j) := \sum_{r=1}^{p} \left(\mathbf{U}_{ri} - \mathbf{U}_{rj}\right)^2 = \|\mathbf{U}_{i:} - \mathbf{U}_{j:}\|^2 = \alpha\text{CT}(i,j) \ \text{ where } \ \alpha \ll 1. \tag{14}$$

This proofs **Corollary** 4.2.

Escape Probabilities. Approximating commute times distances is very convenient for our jump-based analysis, since it is well known that the *escape probability* is related to the commute times distance [33]: $p_{esc} = \frac{1}{\text{CT}(i,j)}$. Escape probabilities are actually dependent on the spectral gap (approximated by the Fiedler value λ_2). This is illustrated in the very first Figure of this paper (Figure 1), where a random walker tries to escape from the community $\bar{A}$. A classic result [35] shows that the probability that a random walk started in its asymptotic ($t \rightarrow \infty$) distribution π is transitioning from $i \in \bar{A}$ to $j \in A$ in one step is $p_{esc}(\bar{A}, A) = \frac{\text{cut}(\bar{A},A)}{\text{vol}(\bar{A})}$, where $\text{cut}(\bar{A}, A) = \sum_{i\in\bar{A}, j\in A} e_{ij}$ (in the Figure we have that $\text{cut}(\bar{A}, A) = 1$).

Therefore, as $d^{t\rightarrow\infty}(i,j) = \alpha\text{CT}(i,j)$, with $\alpha \ll 1$ our jump-hierarchy is closer to that of the escape probability than choosing $\text{CT}(i,j)$ as asymptotic diffusion distance. In addition, we are sensitive to the spectral gap since the Fiedler environment contains approximations of the Fiedler vector, and the spectral gap is approximated in turn by the Dirichlet energy of the Fiedler vector.

Clustering and Metastable States. Minimizing the Dirichlet loss in conjunction with the classification loss (see Eq 2) leads to a trade-off between two *clustering* problems. On the one hand, we infer a piecewise-smooth latent space. On the other hand, we simultaneously try to preserve the structure of the input graph as much as possible. In both cases, we try to find metastable states. A metastable state is a concept borrowed from dynamical systems but basically, it is an equilibrium state in a random process (for instance the one defined by a random walk that tries to escape from a community in Figure 1). Metastable states are also characterized by wells in potential functions $U(\mathbf{x})$, where $\mathbf{x}$ is a state and its probability is given by the Boltzmann distribution $p(\mathbf{x}) = e^{-U(\mathbf{x})}$. Then, the characteristic relaxation processes and time scales of a given space are usually described by a Stochastic Diffusion Equation (SDE):

$$\dot{\mathbf{x}}(t) = -\nabla U(\mathbf{x}) + \sqrt{2}\dot{\mathbf{w}}(t) \tag{15}$$

where $\mathbf{w}$ denotes Brownian motion. In the above equation, we have a *drag* term (the gradient) that drives the process to a deep well, and a *random* term (the Brownian motion) that allows us to escape from local minima. During this process, we find different time scales: fast scales while we are moving through a given well, and slow scales when we try to escape from it. For instance, escaping from the right community in Figure 1 takes a long time which depends on the difference between the potential at the well $U(\mathbf{x}_{min})$ and that of the saddle point $U(\mathbf{x}_{max})$ [24]. This time is in turn the inverse of the spectral gap, i.e. there is a spectral interpretation of the SDE. Such interpretation comes from the analysis of the Fokker-Planck equation:

$$\partial_t p(\mathbf{x},t) = \nabla \cdot \left[\nabla p(\mathbf{x},t) + p(\mathbf{x},t)\nabla U(\mathbf{x})\right] \ , \tag{16}$$

This equation leads to the pdf of the SDE and it has a spectral interpretation. More precisely, the eigenvectors of $\mathbf{P}$ converge to the eigenfunctions $\Psi(\mathbf{x})$ of the Fokker-Planck equation as follows [24][53]:

$$\tilde{\nabla}\Psi(\mathbf{x}) := \triangle\Psi - \nabla\Psi \cdot \nabla U = -\mu\Psi(\mathbf{x}) \ , \tag{17}$$

where $\triangle = \nabla \cdot \nabla$ is the Laplacian and μ the eigenstates (eigenvalues). As a result, we may use the Fiedler vector to characterize the separation between two clusters. The steepest the Fiedler vector, the better the separation (**Theorem** 4.4). Interestingly, the third eigenvector $\tilde{\nabla}$ may not work well as a state separator when we have different spatial scales, as we will see in *Appendix B*.

Dirichlet Label Propagation. As we mentioned in the main paper, our Dirichlet formulation is inspired by classical graph-based semi-supervised methods. The work in [32] poses the problem of propagating known labels $\ell(B)$ to unknown nodes $u \in U$. Let $\mathbf{Y}$ be a $n \times c$ matrix where $\mathbf{Y}(i,j) = 1$ means that node $i \in B$ has label j and $\mathbf{Y}(i,j) = 0$ otherwise. Then, the optimal label of each node i is given by $\mathbf{y}(i) = \arg\max_{j \leq c} \mathbf{F}(i,j)$, where

$$\mathbf{F} = \beta\left(\mathbf{I} - \alpha\mathbf{P}\right)^{-1} \ , \tag{18}$$

being $\mathbf{P}$ the transition matrix and $\alpha + \beta = 1$. The $n \times c$ matrix $\mathbf{F}$ works as a basic node representation (not exactly a latent space) since each of is c rows is stochastic. However, its construction exploits the powers of $\mathbf{P}$ as follows:

$$\mathbf{F}(t) = (\alpha\mathbf{P})^{t-1}\mathbf{Y} + (1-\alpha)\sum_{i=0}^{t-1}(\alpha\mathbf{P})^{i}\mathbf{Y} \ . \tag{19}$$

and

$$\mathbf{F} = \lim_{t\to\infty} \mathbf{F}(t) = (1-\alpha)(\mathbf{I} - \alpha\mathbf{P})^{-1} \ . \tag{20}$$

In addition, we also noted that $\mathbf{F}$ is also the solution of the Dirichlet problem:

$$\text{Min} \ \ \mathcal{L}_Q = \frac{1}{2}\left(\text{Tr}[\mathbf{F}^T \triangle \mathbf{F}] + \mu \sum_i \|\mathbf{F}_{i:} - \mathbf{Y}_{i:}\|^2\right) \ . \tag{21}$$

where $\mu > 0$ is a regularization parameter satisfying $\alpha = 1/(1+\mu)$. The proof is obtained by setting the gradient to zero:

$$\begin{aligned} \nabla \mathcal{L}_{Q_{\mathbf{F}}} &= \mathbf{F} - \mathbf{P}\mathbf{F} + \mu(\mathbf{F} - \mathbf{Y}) \\ &= \mathbf{F} - \frac{1}{1+\mu}\mathbf{P}\mathbf{F} - \frac{\mu}{1+\mu}\mathbf{Y} \\ &= \mathbf{F} - \alpha\mathbf{P}\mathbf{F} - \beta\mathbf{Y} = 0 \ , \end{aligned} \tag{22}$$

which leads to Eq. 19.

Concerning the relationship of this formulation with absorbing random walks [33], the main idea is to extend the $n \times n$ transition matrix $\mathbf{P}$ so that:

a) We include an upper block with the $p \times p$ identity matrix $\mathbf{I}$. This block represents the p absorbing states, where $p = |\ell(B)|$. Then the $n \times p$ block $\mathbf{R}$ encodes the prior probabilities of reaching an absorbing state from a non-absorbing one.

b) The absorbing probabilities are given by $\mathbf{B} := (\mathbf{I}_{n\times n} - \mathbf{P})^{-1}\mathbf{R}$.

Finally, the random-walker version [34] is quite similar to the above one, but reorganizes the Laplacian matrix (**Theorem** 4.3).

7.2 Appendix B: SBM Analysis

The following experiment aims to illustrate the interplay between our novel measure of structural heterophily $\mathcal{R}$ and the extent of the spectral gap. We also show the Fiedler Environments and how they are influenced by the classification loss (labels). For each heterophilic regime, we show both the corresponding pairwise distance matrix (diffusion map) and the resulting homophiliation. We have depicted in Figure 4 the main ingredients of our approach as a means of illustrating some technical details introduced in *Appendix A*. In particular, when analyzing SBM graphs under structural heterophily we observe several interesting phenomena.

Original vs Learned. Instead of precalculating the eigenvectors, as in Directional GNNs [54], we learn them. Our learned (approximate) eigenvectors are relatively close to the Fiedler vector (in terms of how they discriminate the two classes). This is what we call *Fiedler Environments* but, in a semi-supervised setting, i.e. the learned eigenvectors are reactive both to the Dirichlet loss and to the classification loss. Despite being noisy, the vectors in the Fiedler Environments are able of partitioning a class when needed (especially for high values of $\mathcal{R}$).

Diffusion Map. Our pairwise distances are also reactive to semi-supervised classification. However, the Dirichlet loss tends to flatten the intra-communities distances as much as possible. Flattening

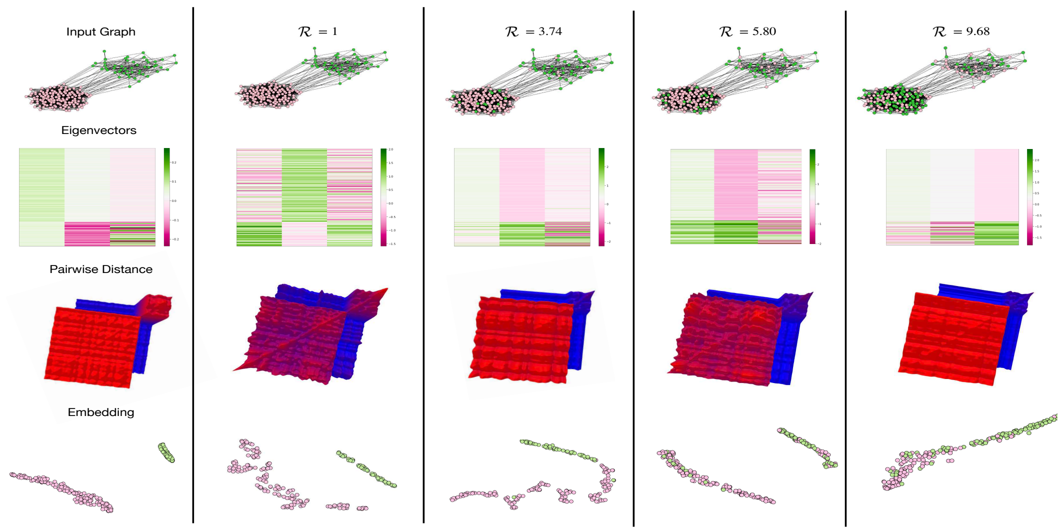


Figure 4: Structural Heterophily in SBMs. Left: The original homophilic graph (First row), its $p = 3$ eigenvectors (Second row), the pairwise distance matrix (Third row), and the resulting embedding. The remaining columns to the right have the same structure for increasing levels of structural heterophily. Note the evolution of the Fiedler Environments and the homophiliations. In all cases, we use $K = 10$ jumps.

is a mechanism to enforce intra-community diffusion in the homophilic regime. In the heterophilic regime, however, the diffusion map enforces exploration via high-order jumping (see lateral steps in the blue region and the loss of the red peak in the small community).

Embedding. We can also see how the embedding is affected by structural heterophily. When we have a structural cluster with nodes of two classes, the respective embeddings are correctly separated, but the margin of this separation decreases as $\mathcal{R}$ increases. This can be seen in graphs that have high $\mathcal{R} \gg 1$, where it is common to find subclusters of nodes that belong to the other classes.

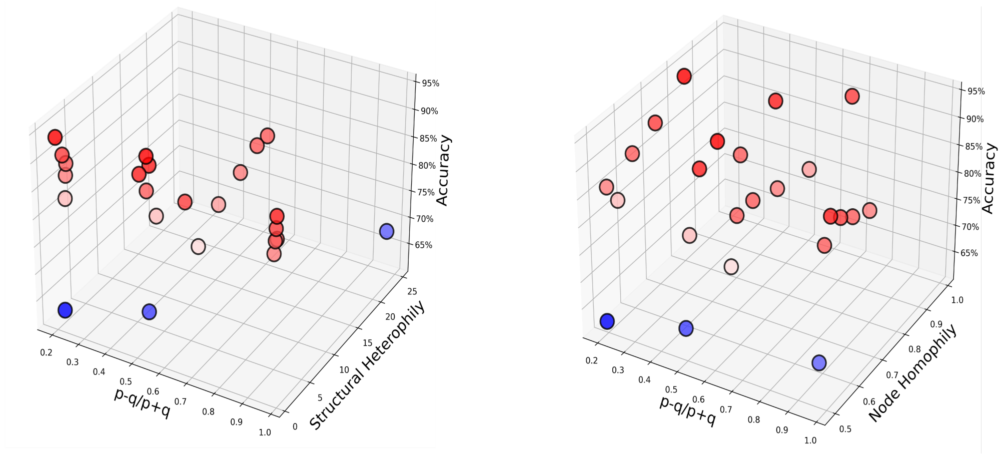


Figure 5: Interplay between heterophily and the spectral gap $\frac{p-q}{p+q}$. Left: Results wrt structural heterophily. Right: Results wrt node homophily.

Interplay between $\mathcal{R}$ and the gap. Finally, we extend the experiments of [55] by incorporating a third axis in addition to variate p and q. This new axis is the structural heterophily. We proceed as follows. We generate four *basic SBMs* attending to increasing spectral gaps: $\frac{p-q}{p+q} \in \{0.2, 0.5, 0.67, 0.98\}$. For each basic SBM we have generated six levels of increasing structural heterophily $\mathcal{R}$. In parallel, we also generate six levels of increasing node homophily as a means of complementing structural heterophily.

We show our results in Figure 5:

a) Small Gaps help. Our method is based on spectral clustering, which means that keeping the gap low factor is key. This helps our method to choose whether to jump outside the cluster looking for a node with the same label (Heterophilic regime) or to stay and only look around (homophilic regime). This common case is supported by our method without problems.

b) Low/Medium Structural Heterophily. If the structure is quite correlated with the label and the spectral gap is not too high, our method is able of achieving good results even when the structure is noisy.

c) Large Gaps lead to oversmoohing. Our worst performance is achieved when the inter-class message passing is massive. This leads to oversmothing due to the high connectivity of the graph. This high connectivity cannot be controlled by our pump (see the blue dots).

We have also performed the same experiment, but changing the structural heterophily measure to node homophily, in order to display the difference between both. Note that our measure fails when the spectral gap is large. This happens because the Dirichlet energy in a near-complete graph is minimal. This lack of structure leads $\mathcal{R}$ to consider that all the nodes are in the same cluster, i.e. is no heterophily).

7.3 Appendix C: Experimental and Computational Details

In this section, we provide details about the datasets (see Table 3) and all the parameters and configurations of our experiments (see Table 4 in order to better clarify the architecture and the results. DIFFUSION-JUMP GNNS is implemented in PyTorch [56], using PyTorch Geometric [57] and ogbn datasets [47]. For reproducibility, code, and instructions are available on our GitHub with all the selected configurations and logs. We have also included the computational (See Figure 6) in order to clarify the derivability of $topk$ in PyTorch.

Table 3: Statistics of the datasets used in our experiments.

DATASET	AVG D	DENSITY	NODE H	CLASS H
TEXAS	1.77	0.0090	0.07	0.001
WISCONSIN	2.05	0.0080	0.17	0.094
CORNELL	1.62	0.0080	0.11	0.047
ACTOR	3.94	0.0005	0.16	0.011
SQUIRREL	41.73	0.0080	0.09	0.025
CHAMELEON	15.85	0.0070	0.10	0.062
CITESEER	2.73	0.0008	0.71	0.627
PUBMED	4.49	0.0002	0.79	0.664
CORA	3.89	0.0014	0.83	0.776
PENN94	3.89	0.0014	0.83	0.776
OGBN-ARXIV	7	0.0004	0.66	0.444
ARXIV-YEAR	7	0.0004	0.22	0.272

In the following Table 4, we include the hyperparameters that have yielded the best results during the experimentation phase. It is worth noting that the experiments were conducted using the same 10 random splits as in [58], training during 700 epochs and utilizing early stopping.

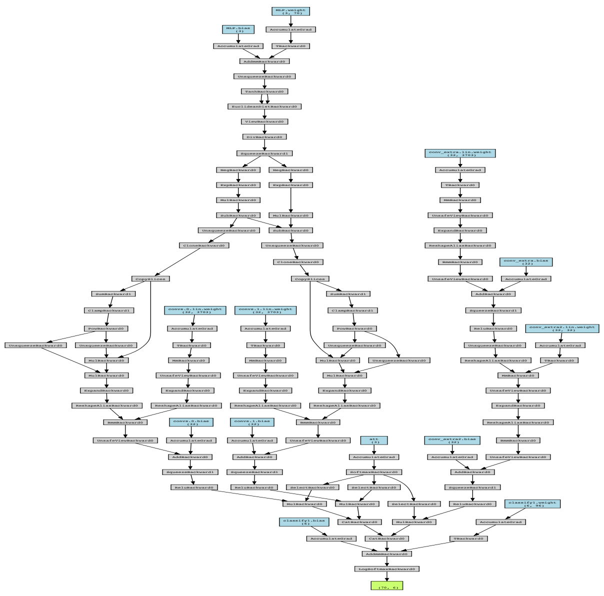


Figure 6: The Computational Graph for $K = 3$ jumps. All branches depend on the diffusion pump (top-left) except HB (the Homophily Branch, top-right).

Table 4: Best hyperparameters for our datasets.

DATASET	HIDDEN CHANNELS	DROPOUT	LR	WEIGHT DECAY	K/#JUMPS
TEXAS	64	0.2	0.03	0.0005	20
WISCONSIN	64	0.5	0.03	0.0005	5
CORNELL	128	0.5	0.03	0.001	5
ACTOR	16	0.2	0.03	0.0001	3
SQUIRREL	128	0.5	0.003	0.0005	8
CHAMELEON	128	0.35	0.003	0.0005	12
CITESEER	128	0.5	0.003	0.0005	5
PUBMED	128	0.3	0.01	0.0005	3
CORA	128	0.5	0.002	0.0005	5
PENN94	16	0.5	0.001	0.0001	3
OGBN-ARXIV	128	0.3	0.01	0.0005	3
ARXIV-YEAR	128	0.2	0.003	0.0005	3