Differentiable Cluster Graph Neural Network

Yanfei Dong [1 2] **Mohammed Haroon Dupty** [2] **Lambert Deng** [1] **Zhuanghua Liu** [2] **Yong Liang Goh** [2]
Wee Sun Lee [2]

Abstract

Graph Neural Networks often struggle with long-range information propagation and may underperform in the presence of heterophilous neighborhoods. We address both of these challenges with a unified framework that incorporates a clustering inductive bias into the message passing mechanism, using additional cluster-nodes. Central to our approach is the formulation of an optimal transport based clustering objective. However, optimizing this objective in a differentiable way is non-trivial. To navigate this, we adopt an iterative process, alternating between solving for the cluster assignments and updating the node/cluster-node embeddings. Notably, our derived optimization steps are themselves simple yet elegant message passing steps operating seamlessly on a bipartite graph of nodes and cluster-nodes. Our clustering-based approach can effectively capture both local and global information, demonstrated by extensive experiments on both heterophilous and homophilous datasets.

1. Introduction

Graph Neural Networks (GNNs) have emerged as prominent models for learning node representations on graph-structured data. Their architectures predominantly adhere to the message passing paradigm, where node embeddings are iteratively refined using features from its adjacent neighbors [1]–[3]. While this message passing paradigm has proven effective in numerous applications [4], [5], two prominent challenges have been observed. First, long-range information propagation over a sparse graph can be challenging [6]–[8]. Expanding the network's reach by increasing the number of layers is often suboptimal as it could encounter issues such as over-squashing [9], [10], where valuable long-range information gets diluted as it passes through

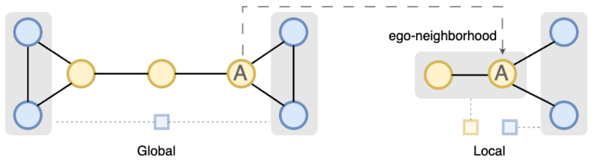


Figure 1: On the left is an instance where distant nodes are similar to each other. On the right is the heterophilous ego-neighborhood of node A where cluster patterns appear.

the graph's bottlenecks, diminishing its impact on the target nodes. Second, some graphs exhibit heterophily, where connected nodes are likely to be dissimilar. In such cases, aggregating information from the dissimilar neighbors could potentially be harmful and hinder the graph representation learning performance [11], [12].

In this paper, we focus on the task of supervised node classification and explore clustering as an inductive bias to address both challenges. We observe that cluster patterns appear globally when nodes that are far apart in the graph have similar features (see Fig. 1). By clustering nodes into multiple global clusters based on their features, we can establish connections between these distant nodes. Additionally, clusters may also be present locally among a node and its neighbours in the graph. Particularly for heterophilic graphs, there are likely more than one cluster in the local neighbourhood. Leveraging these local clusters for information aggregation, rather than across the entire neighborhood at once, can potentially help to avoid the problems caused by the heterophilic nature of the graph.

To embed the clustering inductive bias, we propose the use of a differentiable clustering-based objective function consisting of a weighted sum of global and local clustering terms. The global objective encourages clustering on all the nodes in the graph, while the local objective does so on subsets of nodes, where each subset consists of a node in the graph and its immediate neighbors. Optimizing the objective function would produce a set of node embeddings that can then be used for classifying the nodes in the graph.

To optimize the implicit clustering-based objective function, we employ a block coordinate descent approach by iteratively, (1) solving for a soft cluster assignment matrix that probabilistically assigns a node to a cluster, followed by, (2) updating the node and cluster embeddings given the

[1]PayPal [2]National University of Singapore. Correspondence to: Yanfei Dong <yanfei.dong43@gmail.com>.

Published at the 2^{nd} Differentiable Almost Everything Workshop at the 41^{st} International Conference on Machine Learning, Vienna, Austria. July 2024. Copyright 2024 by the author(s).

cluster assignment matrix. We view the first step of solving for the assignment matrix as an Optimal Transport (OT) problem [13], which aims to find the most cost-effective way to move from one probability distribution to another. In our context, the cost is defined as the distance between nodes and cluster-centroids, and the goal is to transport nodes to cluster centroids with minimum cost. We adopt the entropic regularized Sinkhorn distance [14] approximation for solving the OT problem. This approximation utilizes differentiable operations, ensuring end-to-end learning. For the second step of updating the node and cluster embeddings, we derive a closed-form solution that minimizes the objective function given the assignment matrix. The derived algorithm can be viewed as a message passing algorithm on a bipartite graph with an additional set of cluster-nodes. The cluster-nodes represent cluster centroids both globally of the entire graph and locally of each neighborhood. Importantly, our message-passing steps act as optimization steps that enforce the clustering inductive bias.

We name our method Differentiable Cluster Graph Neural Network (DC-GNN) and show its convergence. Additionally, both types of cluster-nodes can be viewed as providing new pathways among the original nodes. This helps to lower the graph's total effective resistance, which is useful in mitigating oversquashing [15].

DC-GNN is efficient and has a linear complexity with respect to the graph size (Appendix. F.1). We carry out comprehensive experiments on a variety of datasets, including both heterophilous and homophilous types. The results underscore the efficacy of our method, with our approach attaining state-of-the-art performance in multiple instances.

2. Methodology

In this section, we introduce our unified clustering-based GNN message passing method to address both long-range interactions and heterophilous neighborhood aggregation.

2.1. DC-GNN formulation

We begin by constructing a bipartite graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{C}, \mathcal{E})$. This bipartite graph is derived from the original graph $G = (V, E)$ and comprises two distinct sets of nodes. The first set, $\mathcal{V}$, is a direct copy of the nodes V from the original graph. The second set, $\mathcal{C}$, consists of cluster nodes divided into two categories: global clusters (Ω) and local clusters (Γ). In this bipartite graph, each node from the global clusters Ω connects to all nodes in $\mathcal{V}$, facilitating long-range interactions across distant nodes. Meanwhile, each node from the local clusters Γ is associated with a specific node i in $\mathcal{V}$, and connected to its ego-neighborhood, which includes node i and its one-hop neighbors. For a node i in $\mathcal{V}$, Γ_i represents the set of local clusters associated with it. Hence, $|\mathcal{C}| = |\Omega| + \sum_{i\in\mathcal{V}} |\Gamma_i|$. An illustration of bipartite graph construction is provided in Appendix. D.

There are three components in our Diffentiable Cluster Graph Neural Network (DC-GNN) architecture:

$$\begin{aligned} X &= \texttt{Transform}(\hat{X}) \\ Z &= \texttt{DC-MsgPassing}(\mathcal{G}, X) \\ Y &= \texttt{Readout}(Z) \end{aligned}$$

Given $\hat{X}$ as input node features, we first transform it with a learnable input transformation function to produce X. The transformed features are then put through the clustering-based message passing function to produce node embeddings Z. A learnable readout function is then used on Z to produce the final output Y. Then, DC-GNN is trained end-to-end with a loss function $\mathcal{L}_{\text{train}}$ via backpropagation. At the core of our design is the iterative `DC-MsgPassing` procedure, an implicit optimization algorithm that optimizes a clustering-based objective function $\mathcal{L}^{\lambda}_{\text{cluster}}$.

2.2. Clustering-based objective function $\mathcal{L}_{\text{cluster}}$

In this work, we aim to address over-squashing and heterophily by embedding a clustering inductive bias into the design of the GNN. One way to achieve this is by optimizing a clustering-based objective function. Motivated by the theory of optimal transport, we propose a novel soft clustering-based objective function.

We conceptualize the cluster assignment problem as an optimal transport problem, where the cost is defined as the distance between the embeddings of nodes and their cluster centroids. Therefore, we propose to minimize the overall cost, weighted by the soft cluster assignment matrix P which indicates the amount of assignment from a node to a cluster. Specifically, we have a single global soft cluster assignment matrix $P^{\Omega} \in \mathbb{R}^{|\mathcal{V}|\times|\Omega|}$ and local soft cluster assignment matrices $P^{\Gamma_i} \in \mathbb{R}^{|\mathcal{N}_i^+|\times|\Gamma_i|}$ for each node i. Let $d(u, v)$ represent the distance between two vectors u and v, z_i be the node embeddings to be learnt for each node $i \in \mathcal{V}$, x_i be the node features after an initial transformation by a multilayer perceptron, and c_j^{Ω}, $c_j^{\Gamma_i}$ be the embeddings of the j^{th} global and local cluster-node respectively. Then we define our objective function $\mathcal{L}_{\text{cluster}}$ as

$$\begin{aligned} \mathcal{L}_{\text{cluster}} = & \underbrace{\alpha \sum_{i\in\mathcal{V}} \sum_{j\in\Omega} P_{ij}^{\Omega} d(z_i, c_j^{\Omega})}_{\text{global clustering}} + \underbrace{\beta \sum_{i\in\mathcal{V}} d(z_i, x_i)}_{\text{node fidelity}} \\ & + (1-\alpha) \underbrace{\sum_{i\in\mathcal{V}} \sum_{u\in i\cup\mathcal{N}(i)} \sum_{j\in\Gamma_i} P_{uj}^{\Gamma_i} d(z_u, c_j^{\Gamma_i})}_{\text{local clustering}} \end{aligned} \quad (1)$$

The *global clustering* part optimizes the OT distance between node embeddings and global cluster-node embeddings. The *local clustering* term optimizes the OT distance between the embeddings of nodes and local cluster-nodes within each ego-neighborhood. The scalar parameter $\alpha \in [0,1]$ is a balancing factor between the two terms. Furthermore, the *node fidelity* term encourages the node embeddings to retain some information from the original node features [16], [17].

2.3. DC-MsgPassing: optimization via message passing

Since conventional OT solvers can be computationally prohibitive [18], we adopt the entropy regularized version of the OT distance that is designed for efficiency and offers a good approximation of OT distance [14].Specifically, let $h(P)$ be the entropy of the assignment matrix P, we propose the following refined objective function,

$$\mathcal{L}^{\lambda}_{\text{cluster}} \coloneqq \mathcal{L}_{\text{cluster}} - \frac{\alpha}{\lambda} h(P^{\Omega}) - \frac{(1-\alpha)}{\lambda} \sum_{i \in \mathcal{V}} h(P^{\Gamma_i}). \quad (2)$$

Direct optimization of $\mathcal{L}^{\lambda}_{\text{cluster}}$ as a training objective is intractable due to the presence of unobserved assignment matrices P^{Ω} and P^{Γ_i}, since we cannot compute Eq. (2) without estimating the cluster assignment values. To overcome this, we propose an iterative block coordinate descent algorithm called `DC-MsgPassing` that alternatively update the assignment matrices P^{Ω}, P^{Γ_i} and embeddings Z, C.

2.3.1. ASSIGNMENT UPDATE

Update for assignment matrices P^{Ω}, P^{Γ_i}: With node and cluster-node embeddings kept constant, we aim to update the P^{Ω}, P^{Γ_i} minimizing $\mathcal{L}_{\text{cluster}}$. We approach this cluster assignment problem from the perspective of optimal transport theory, and use P^{Ω} as an example without loss of generality. Our objective is to determine the optimal assignment matrix $P^{\Omega *} \in \mathbb{R}_{+}^{|\mathcal{V}| \times |\Omega|}$ for a given cost matrix $M \in \mathbb{R}_{+}^{|\mathcal{V}| \times |\Omega|}$, aligning with a target clustering distribution. Assuming a uniform distribution of data points across all clusters, we aim to find a mapping from $\mathbf{u}^{\top} = \left[|\mathcal{V}|^{-1} \ \cdots \ |\mathcal{V}|^{-1}\right]_{1 \times |\mathcal{V}|}$ to $\mathbf{v}^{\top} = \left[|\Omega|^{-1} \ \cdots \ |\Omega|^{-1}\right]_{1 \times |\Omega|}$, minimizing the overall cost. To formalize, optimizing $\mathcal{L}_{\text{cluster}}$ with respect to P^{Ω} is tantamount to solving: $\min_{P^{\Omega} \in U(\mathbf{u},\mathbf{v})} \langle P^{\Omega}, M \rangle$, where $U(\mathbf{u},\mathbf{v}) = \{P \in \mathbb{R}_{+}^{|\mathcal{V}| \times |\Omega|} : P\mathbf{1}_{|\Omega|} = \mathbf{u}, P^{\top}\mathbf{1}_{|\mathcal{V}|} = \mathbf{v}\}$. $\langle \cdot, \cdot \rangle$ is the Frobenius dot-product, and $\langle P^{\Omega}, M \rangle = \sum_{i \in \mathcal{V}} \sum_{j \in \Omega} P^{\Omega}_{ij} d(z_i, c^{\Omega}_j)$. Importantly, $M_{ij} = d(z_i, c^{\Omega}_j)$ can be seen as the cost of assigning node i to cluster j, and P^{Ω}_{ij} indicates the amount of assignment from node i to cluster j. This is a classical optimal transport problem.While such problems are typically solved via linear programming techniques, these approaches are computationally expensive [18]. To overcome it, we opt for the Sinkhorn distance [14] instead, which offers a good approximation to the optimal transport distance with additional entropic regularization, where $\lambda > 0$. Formally,

$$\langle P^{\Omega}_{\lambda}, M \rangle, \text{ where } P^{\Omega}_{\lambda} = \operatorname*{arg\,min}_{P^{\Omega} \in U(\mathbf{u},\mathbf{v})} \langle P^{\Omega}, M \rangle - \frac{1}{\lambda} h(P^{\Omega}). \quad (3)$$

The benefit of having this entropic regularization term $h(P^{\Omega})$ is that the solution $P^{\Omega *}_{\lambda}$ now has the form $P^{\Omega *}_{\lambda} = UBV$ [14], where $B = e^{-\lambda M}$, and U and V are diagonal matrices. Now the OT problem reduces to the classical matrix scaling problem, for which the objective is to determine if there exist diagonal matrices U and V such that the i^{th} row of the matrix UBV sums to $\mathbf{u}_i$ and the j^{th} column of UBV sums to $\mathbf{v}_j$. Since $e^{-\lambda M}$ is strictly positive, there exists a unique $P^{\Omega *}_{\lambda}$ that belongs to $U(\mathbf{u},\mathbf{v})$ [19], [20], which can be obtained by the well-known Sinkhorn–Knopp algorithm [20], [21].

To obtain $P^{\Omega *}_{\lambda}$, we run the Sinkhorn–Knopp algorithm which iteratively updates the matrix B by scaling each row of B by the respective row-sum, and each column of B by the respective column-sum. Formally, we have

$$B^{(t)}_{ij} = \frac{B^{(t-1)}_{ij}}{\sum_j B^{(t-1)}_{ij}} \mathbf{u}_i, \quad B^{(t+1)}_{ij} = \frac{B^{(t)}_{ij}}{\sum_i B^{(t)}_{ij}} \mathbf{v}_j. \quad (4)$$

After T steps, we update P^{Ω}_{ij} by the value of B_{ij}. Similarly, we update P^{Γ_i} in the *local clustering* term. The scaling operations in Sinkhorn–Knopp are *fully differentiable*, enabling end-to-end learning.

2.3.2. EMBEDDINGS UPDATE

With the updated assignment matrices P^{Ω}, P^{Γ_i}, we now derive the update functions for cluster-node and node embeddings. If we choose the distance function $d(\cdot)$ to be the squared Euclidean norm, $\mathcal{L}^{\lambda}_{\text{cluster}}$ is a quadratic model of node embeddings z_i, c_j. Then we can update the embeddings by minimizing the $\mathcal{L}^{\lambda}_{\text{cluster}}$with respect to each variable, which admits the following closed-form solution:

Update for cluster-node embeddings C:

$$C = \text{diag}(\mathbf{k}) P^{\top} Z, \quad (5)$$

Update for node embeddings Z:

$$Z = \gamma \Big[\beta X + \alpha P^{\Omega} C^{\Omega} + (1-\alpha) \sum_{u \in \mathcal{N}^{+}_i} \hat{P}^{\Gamma_u} C^{\Gamma_u} \Big], \quad (6)$$

where $\mathbf{k}^{\top} = \left[|\Omega|\mathbf{1}_{|\Omega|}; |\Gamma_i|\mathbf{1}_{|\Gamma_i|}; \cdots ; |\Gamma_{|\mathcal{V}|}|\mathbf{1}_{|\Gamma_i|}\right]_{1 \times |\mathcal{C}|}$, $P \in \mathbb{R}_{+}^{|\mathcal{V}| \times |\mathcal{C}|}$ is the overall assignment matrix, $\gamma = (\alpha|\mathcal{V}|^{-1} + \beta + 1 - \alpha)^{-1}$ is a constant, and $\hat{P}^{\Gamma_u} \in \mathbb{R}_{+}^{|\mathcal{V}| \times |\Gamma_u|}$

Table 1: Classification performance comparison across various heterophilous and homophilous datasets.

	Heterophilous						**Homophilous**		
	Penn94	**Genius**	**Cornell5**	**Amherst41**	**US-election**	**Wisconsin**	**Cora**	**Citeseer**	**Pubmed**
MLP	73.61 (0.40)	86.68 (0.09)	68.86 (1.83)	60.43 (1.26)	81.92 (1.01)	85.29 (3.31)	75.69 (2.00)	74.02 (1.90)	87.16 (0.37)
GCN	82.47 (0.27)	87.42 (0.37)	80.15 (0.37)	81.41 (1.70)	82.07 (1.65)	51.76 (3.06)	86.98 (1.27)	76.50 (1.36)	88.42 (0.50)
GAT	81.53 (0.55)	55.80 (0.87)	78.96 (1.57)	79.33 (2.09)	84.17 (0.98)	49.41 (4.09)	87.30 (1.10)	76.55 (1.23)	86.33 (0.48)
MixHop	83.47 (0.71)	90.58 (0.16)	78.52 (1.22)	76.26 (2.56)	85.90 (1.55)	75.88 (4.90)	87.61 (0.85)	76.26 (1.33)	85.31 (0.61)
GCNII	82.92 (0.59)	90.24 (0.09)	78.85 (0.78)	76.02 (1.38)	82.90 (0.29)	80.39 (3.40)	88.37 (1.25)	77.33 (1.48)	90.15 (0.43)
H_2GCN	81.31 (0.60)	OOM	78.46 (0.75)	79.64 (1.63)	85.53 (0.77)	87.65 (4.98)	87.87 (1.20)	77.11 (1.57)	89.49 (0.38)
WRGAT	74.32 (0.53)	OOM	71.11 (0.48)	62.59 (2.46)	84.45 (0.56)	86.98 (3.78)	88.20 (2.26)	76.81 (1.89)	88.52 (0.92)
GPR-GNN	81.38 (0.16)	90.05 (0.31)	73.30 (1.87)	67.00 (1.92)	84.49 (1.09)	82.94 (4.21)	87.95 (1.18)	77.13 (1.67)	87.54 (0.38)
GGCN	73.62 (0.61)	OOM	71.35 (0.81)	66.53 (1.61)	84.71 (2.60)	86.86 (3.29)	87.95 (1.05)	77.14 (1.45)	89.15 (0.37)
ACM-GCN	82.52 (0.96)	80.33 (3.91)	78.17 (1.42)	70.11 (2.10)	85.14 (1.33)	88.43 (3.22)	87.91 (0.95)	77.32 (1.70)	90.00 (0.52)
LINKX	84.71 (0.52)	90.77 (0.27)	83.46 (0.61)	81.73 (1.94)	84.08 (0.67)	75.49 (5.72)	84.64 (1.13)	73.19 (0.99)	87.86 (0.77)
GloGNN++	85.74 (0.42)	90.91 (0.13)	83.96 (0.46)	81.81 (1.50)	85.48 (1.19)	88.04 (3.22)	88.33 (1.09)	77.22 (1.78)	89.24 (0.39)
DC-GNN	**86.69** (0.22)	**91.70** (0.08)	**84.68** (0.24)	**82.94** (1.59)	**89.59** (1.60)	**91.67** (1.95)	**89.13** (1.18)	**77.93** (1.82)	**91.00** (1.28)

is the broadcasted local assignment matrix from $P^{\Gamma_u} \in \mathbb{R}_+^{|\mathcal{N}_u^+| \times |\Gamma_u|}$. See Appendix C.2, C.3 and C.4 for notation and full derivation. Intriguingly, this closed-form solution reveals that Z is updated by a linear combination of global and local cluster-node embeddings, weighted by cluster assignment probabilities. α balances the influence of local and long-range interactions while β scales the original node features that provide the initial residual [17].

Remark. Viewing cluster assignment probabilities as edge weights, Eq. (5) serves as message passing from nodes to cluster-nodes and Eq. (6) represents the message passing from cluster-nodes back to original nodes.

In summary, `DC-MsgPassing` is the key to our method. *Each iteration of* `DC-MsgPassing` *is one optimization step towards minimizing* $\mathcal{L}_{\text{cluster}}^{\lambda}$. Thus our message passing mechanism provides the needed inductive bias in learning the local and global clusters present in the data, while simultaneously learning node embeddings using the clustering information. `DC-MsgPassing` is generally well behaved, converging when enough iterations are performed.

Theorem 2.1 (Convergence of `DC-MsgPassing`). *Assuming the Sinkhorn–Knopp algorithm is run to convergence in each iteration, for any* $\lambda > 0$, *the value of* $\mathcal{L}_{\text{cluster}}^{\lambda}$ *produced by* `DC-MsgPassing` *is guaranteed to converge.*

Proof can be found in the Appendix C.1.

2.4. Training objective function $\mathcal{L}_{\text{train}}$

Our task is supervised node-classification. We use cross entropy loss $\mathcal{L}_{ce}$ to train DC-GNN along with two regularizing losses to facilitate clustering: $\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{ce}} + \omega_1 \mathcal{L}_{\text{ortho}} + \omega_2 \mathcal{L}_{\text{sim}}$. (See App. E for $\mathcal{L}_{\text{ortho}}$ and $\mathcal{L}_{\text{sim}}$).

3. Experiments

In this section, we empirically validate the capabilities of DC-GNN . Dataset, baselines, implementation and training details are in App. G.6 and G.7. We also conduct experiments on additional heterophilous datasets (App.G.1), in sparse label settings (App.G.2), oversquashing alleviation (App.G.3) and ablation studies (App.G.4).

3.1. Comparison with baselines on heterophilous graphs

We conduct experiments on heterophilous datasets where long-range information is beneficial and neighborhood aggregation needs special attention. We achieve state-of-the-art on all six datasets in Tab. 1. The performance uplift is especially pronounced on US-election and Wisconsin, where it outperforms by more than 3%. We also achieve state-of-the-art on four other heterophilous graphs(App. Tab. 3).

3.2. Comparison with baselines on homophilous graphs

We further run DC-GNN on three homophilous datasets, where it outperforming both general-purpose and heterophilous baselines (Tab. 1). Our strong performance on both homophilous and heterophilous graphs shows that DC-GNN is flexible and adaptive, capable of effective information aggregation on various homophily levels.

4. Conclusion

This paper tackles two challenges in GNNs: long-range information propagation and heterophilous neighborhood aggregation. We proposed a novel differentiable framework that seamlessly embeds a clustering inductive bias into the message passing mechanism, facilitated by the introduction of cluster-nodes. Central to our approach is an optimal transport based clustering objective, which we optimize through an iterative strategy, alternating between the computation of cluster assignments and the refinement of node/cluster-node embeddings. Importantly, the derived optimization steps effectively function as message passing operations on the bipartite graph. Our effectiveness in capturing both local nuances and global structures is validated by extensive experiments on various datasets.

References

[1] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.

[2] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” *NeurIPS*, vol. 29, pp. 3844–3852, 2016.

[3] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in *International conference on machine learning*, PMLR, 2017, pp. 1263–1272.

[4] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph convolutional neural networks for web-scale recommender systems,” in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 974–983.

[5] J. Zhou, G. Cui, S. Hu, *et al.*, “Graph neural networks: A review of methods and applications,” *AI Open*, vol. 1, pp. 57–81, 2020.

[6] Q. Li, Z. Han, and X.-M. Wu, “Deeper insights into graph convolutional networks for semi-supervised learning,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[7] K. Zhou, Y. Dong, K. Wang, *et al.*, “Understanding and resolving performance degradation in deep graph convolutional networks,” in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 2728–2737.

[8] T. K. Rusch, M. M. Bronstein, and S. Mishra, “A survey on oversmoothing in graph neural networks,” *arXiv preprint arXiv:2303.10993*, 2023.

[9] U. Alon and E. Yahav, “On the bottleneck of graph neural networks and its practical implications,” *arXiv preprint arXiv:2006.05205*, 2020.

[10] J. Topping, F. Di Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein, “Understanding over-squashing and bottlenecks on graphs via curvature,” *arXiv preprint arXiv:2111.14522*, 2021.

[11] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, “Beyond homophily in graph neural networks: Current limitations and effective designs,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 7793–7804, 2020.

[12] J. Zhu, R. A. Rossi, A. Rao, *et al.*, “Graph neural networks with heterophily,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 11 168–11 176.

[13] C. Villani *et al.*, *Optimal transport: old and new*. Springer, 2009, vol. 338.

[14] M. Cuturi, “Sinkhorn distances: Lightspeed computation of optimal transport,” *Advances in neural information processing systems*, vol. 26, 2013.

[15] M. Black, Z. Wan, A. Nayyeri, and Y. Wang, “Understanding oversquashing in gnns through the lens of effective resistance,” in *International Conference on Machine Learning*, PMLR, 2023, pp. 2528–2547.

[16] J. Klicpera, A. Bojchevski, and S. Günnemann, “Predict then propagate: Graph neural networks meet personalized pagerank,” *arXiv preprint arXiv:1810.05997*, 2018.

[17] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, “Simple and deep graph convolutional networks,” in *ICML*, PMLR, 2020, pp. 1725–1735.

[18] O. Pele and M. Werman, “Fast and robust earth mover’s distances,” in *2009 IEEE 12th international conference on computer vision*, IEEE, 2009, pp. 460–467.

[19] M. Menon, “Matrix links, an extremization problem, and the reduction of a non-negative matrix to one with prescribed row and column sums,” *Canadian Journal of Mathematics*, vol. 20, pp. 225–232, 1968.

[20] R. Sinkhorn, “Diagonal equivalence to matrices with prescribed row and column sums,” *The American Mathematical Monthly*, vol. 74, no. 4, pp. 402–405, 1967.

[21] R. Sinkhorn and P. Knopp, “Concerning nonnegative matrices and doubly stochastic matrices,” *Pacific Journal of Mathematics*, vol. 21, no. 2, pp. 343–348, 1967.

[22] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” *arXiv preprint arXiv:1312.6203*, 2013.

[23] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.

[24] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?” *arXiv preprint arXiv:1810.00826*, 2018.

[25] C. Cai and Y. Wang, “A note on over-smoothing for graph neural networks,” *arXiv preprint arXiv:2006.13318*, 2020.

[26] P. K. Banerjee, K. Karhadkar, Y. G. Wang, U. Alon, and G. Montúfar, “Oversquashing in gnns through the lens of information contraction and graph expansion,” in *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2022, pp. 1–8.

[27] K. Karhadkar, P. K. Banerjee, and G. Montúfar, “Fosr: First-order spectral rewiring for addressing oversquashing in gnns,” *arXiv preprint arXiv:2210.11790*, 2022.

[28] K. Nguyen, N. M. Hieu, V. D. Nguyen, N. Ho, S. Osher, and T. M. Nguyen, “Revisiting over-smoothing and over-squashing using ollivier-ricci curvature,” in *International Conference on Machine Learning*, PMLR, 2023, pp. 25 956–25 979.

[29] A. Arnaiz-Rodríguez, A. Begga, F. Escolano, and N. M. Oliver, “Diffwire: Inductive graph rewiring via the lovász bound,” in *The First Learning on Graphs Conference*, 2022.

[30] E. Chien, J. Peng, P. Li, and O. Milenkovic, “Adaptive universal generalized pagerank graph neural network,” *arXiv preprint arXiv:2006.07988*, 2020.

[31] G. Fu, P. Zhao, and Y. Bian, “P-laplacian based graph neural networks,” in *International Conference on Machine Learning*, vol. 162, PMLR, 2022, pp. 6878–6917.

[32] Y. Dong, K. Ding, B. Jalaian, S. Ji, and J. Li, “Adagnn: Graph neural networks with adaptive frequency response filter,” in *International Conference on Information and Knowledge Management*, ACM, 2021, pp. 392–401.

[33] J. Xu, E. Dai, X. Zhang, and S. Wang, “Hp-gmn: Graph memory networks for heterophilous graphs,” *arXiv preprint arXiv:2210.08195*, 2022.

[34] D. Jin, Z. Yu, C. Huo, *et al.*, “Universal graph convolutional networks,” in *Advances in Neural Information Processing Systems*, 2021, pp. 10 654–10 664.

[35] X. Li, R. Zhu, Y. Cheng, *et al.*, “Finding global homophily in graph neural networks when meeting heterophily,” *arXiv preprint arXiv:2205.07308*, 2022.

[36] S. Abu-El-Haija, B. Perozzi, A. Kapoor, *et al.*, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *ICML*, PMLR, 2019, pp. 21–29.

[37] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," *arXiv preprint arXiv:2002.05287*, 2020.

[38] J. Zhu, R. A. Rossi, A. Rao, *et al.*, "Graph neural networks with heterophily," *arXiv preprint arXiv:2009.13566*, 2020.

[39] D. Lim, F. Hohne, X. Li, *et al.*, "Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods," *Advances in Neural Information Processing Systems*, vol. 34, pp. 20 887–20 902, 2021.

[40] Y. Yan, M. Hashemi, K. Swersky, Y. Yang, and D. Koutra, "Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks," *arXiv preprint arXiv:2102.06462*, 2021.

[41] A. Tsitsulin, J. Palowitch, B. Perozzi, and E. Müller, "Graph clustering with graph neural networks," *Journal of Machine Learning Research*, vol. 24, no. 127, pp. 1–21, 2023.

[42] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 28, 2014.

[43] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," in *International Conference on Machine Learning*, PMLR, 2020, pp. 874–883.

[44] A. Duval and F. Malliaros, "Higher-order clustering and pooling for graph neural networks," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 426–435.

[45] J. Feng, Y. Chen, F. Li, A. Sarkar, and M. Zhang, "How powerful are k-hop message passing graph neural networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 4776–4790, 2022.

[46] B. Saha, D. Krotov, M. J. Zaki, and P. Ram, "End-to-end differentiable clustering with associative memories," in *International Conference on Machine Learning*, PMLR, 2023, pp. 29 649–29 670.

[47] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 132–149.

[48] L. Stewart, F. Bach, F. Llinares-López, and Q. Berthet, "Differentiable clustering with perturbed spanning forests," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[49] G. Bécigneul, O.-E. Ganea, B. Chen, R. Barzilay, and T. Jaakkola, "Optimal transport graph neural networks," *arXiv preprint arXiv:2006.04804*, 2020.

[50] C. Vincent-Cuaz, R. Flamary, M. Corneli, T. Vayer, and N. Courty, "Template based graph neural network with optimal transport distances," *Advances in Neural Information Processing Systems*, vol. 35, pp. 11 800–11 814, 2022.

[51] L. Yang, J. Gu, C. Wang, *et al.*, "Toward unsupervised graph neural network: Interactive clustering and embedding via optimal transport," in *2020 IEEE international conference on data mining (ICDM)*, IEEE, 2020, pp. 1358–1363.

[52] X. Li, Y. Grandvalet, R. Flamary, N. Courty, and D. Dou, "Representation transfer by optimal transport," *arXiv preprint arXiv:2007.06737*, 2020.

[53] Z. Zhang, F. Wu, and W. S. Lee, "Factor graph neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 8577–8587, 2020.

[54] Z. Zhang, M. H. Dupty, F. Wu, J. Q. Shi, and W. S. Lee, "Factor graph neural networks," *Journal of Machine Learning Research*, vol. 24, no. 181, pp. 1–54, 2023. [Online]. Available: http://jmlr.org/papers/v24/21-0434.html.

[55] M. H. Dupty and W. S. Lee, "Neuralizing efficient higher-order belief propagation," *arXiv preprint arXiv:2010.09283*, 2020.

[56] M. H. Dupty, Y. Dong, S. Leng, *et al.*, "Constrained layout generation with factor graphs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 12 851–12 860.

[57] V. G. Satorras and M. Welling, "Neural enhanced belief propagation on factor graphs," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2021, pp. 685–693.

[58] K. Yoon, R. Liao, Y. Xiong, *et al.*, "Inference in probabilistic graphical models by graph neural networks," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, IEEE, 2019, pp. 868–875.

[59] J. Franklin and J. Lorenz, "On the scaling of multidimensional matrices," *Linear Algebra and its applications*, vol. 114, pp. 717–735, 1989.

[60] G. W. Soules, "The rate of convergence of sinkhorn balancing," *Linear algebra and its applications*, vol. 150, pp. 3–40, 1991.

[61] D. Chakrabarty and S. Khanna, "Better and simpler error analysis of the sinkhorn–knopp algorithm for matrix scaling," *Mathematical Programming*, vol. 188, no. 1, pp. 395–407, 2021.

[62] O. Platonov, D. Kuznedelev, M. Diskin, A. Babenko, and L. Prokhorenkova, "A critical look at the evaluation of gnns under heterophily: Are we really making progress?" *arXiv preprint arXiv:2302.11640*, 2023.

[63] L. Müller, M. Galkin, C. Morris, and L. Rampášek, "Attending to graph transformers," *arXiv preprint arXiv:2302.04181*, 2023.

[64] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.

[65] J. Jia and A. R. Benson, "Residual correlation in graph neural network regression," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 588–598.

[66] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International conference on machine learning*, PMLR, 2016, pp. 40–48.

A. Notations

All notations are listed in Tab. 2.

B. Related Work

Prominent GNN models typically follow a message passing paradigm that iteratively aggregates information in a node's neighborhood [1]–[3], [22]–[24]. This local message passing, however, requires the stacking of multiple layers to pursue long-range information and can encounter issues such as over-smoothing [6], [8], [25] and over-squashing [9], [10], [26], [27]. To tackle oversquashing, existing works design graph rewiring techniques that change graph topology [10], [27]–[29]. Similar to some existing methods [15], our approach of adding cluster-nodes also changes graph topology and is shown to reduce effective resistance, thereby helpful in mitigating oversquashing [15].

Additionally, some graphs contain heterophilous neighborhoods, in which traditional aggregation promoting similarity among neighbors is suboptimal. [11], [12]. There are three types of approaches to address this, including design of high-pass filters in message passing [30]–[32], exploring global neighborhoods [33]–[36], and use of auxiliary graph structures [37]–[40]. Many of these approaches are often computationally expensive and may struggle on homophilous graphs. Our method seeks to explore global neighborhoods by introducing cluster-nodes as auxiliary structures and conduct clustered aggregation in local neighborhoods, with linear complexity.

Our work focuses on using a clustering inductive bias to enhance the supervised node classification task. This differs from tasks like graph clustering [41], [42] and graph pooling [43], [44], which are constrained by graph typology and aim to partition a graph into substructures. Instead, our approach primarily utilizes feature information for global clustering. Unlike previous methods, we also explore clustering within local neighborhoods, which has not been explored before. Additionally, we integrate the clustering bias into the message passing mechanism in an end-to-end differentiable framework.

Clustering with a differentiable pipeline has been explored in some works including in unsupervised and self-supervised settings [45]–[48]. However, these works do not explore application on graph structured data. Central to our approach is an Optimal Transport based clustering objective. OT has been recently applied to graph learning for tasks like graph classification [49], [50], regularizing node representations [51] and finetuning [52]. However, most of these methods leverage OT as a separate component and do not integrate it within message passing. Distinct from these approaches, we implicitly optimize an OT-based clustering objective function via message passing.

Our approach of message passing with additional cluster-nodes can also be viewed as the construction of factor graph neural network with the cluster-nodes representing factors, which has been effective in learning representation on structured data [53]–[58]. Our approach provides further evidence for the usefulness of factor graph based approaches.

Table 2: Table for notations.

Variable	Definition
$\mathcal{G}$	bipartite graph, denoted as $(\mathcal{V}, \mathcal{C}, \mathcal{E})$
G	original graph, denoted as (V, E)
V	set of nodes from the original graph G
$\mathcal{V}$	vertices of $\mathcal{G}$, direct copy of V
E	set of edges from the original graph G
$\mathcal{E}$	set of edges in the bipartite graph $\mathcal{G}$
$\mathcal{N}_i$	set of one-hop neighbors of node i
$\mathcal{N}_i^+$	node i and its one-hop neighbors (ego-neighborhood of i)
$\mathcal{C}$	set of cluster-nodes in the bipartite graph $\mathcal{G}$
Ω	set of global cluster-nodes
Γ	set of local cluster-nodes
Γ_i	set of local cluster-nodes associated with $\mathcal{N}_i^+$
C	set of cluster-node embeddings
Z	set of node embeddings
Y	predicted class probabilities
X_{input}	input features
X	transformed input features
P	overall cluster assignment matrix. $P \in \mathbb{R}_+^{\lvert\mathcal{V}\rvert\times\lvert\mathcal{C}\rvert}$
P^Ω	global cluster assignment matrix. $P^\Omega \in \mathbb{R}_+^{\lvert\mathcal{V}\rvert\times\lvert\Omega\rvert}$
P^Γ	local cluster assignment matrix. $P^{\Gamma_i} \in \mathbb{R}_+^{\lvert\mathcal{N}_i^+\rvert\times\lvert\Gamma_i\rvert}$ for each node i
$d(\cdot)$	distance function
z_i	node embeddings of node i
x_i	node features of node i after initial transformation
c_j^Ω	node embeddings of j^{th} global cluster-node
$c_j^{\Gamma_i}$	node embeddings of j^{th} local cluster-node for $\mathcal{N}_i^+$
α	scalar parameter that balances global and local clustering objectives. $\alpha \in [0, 1]$
β	scalar parameter for node fidelity term
$h(\cdot)$	entropy function
$P^{\Omega*}$	optimal global soft-assignment matrix. $P^{\Omega*} \in \mathbb{R}_+^{\lvert\mathcal{V}\rvert\times\lvert\Omega\rvert}$
M_{ij}	cost of assigning node i to cluster j
M	cost matrix
$\mathbf{u}^\top$	$\mathbf{u}^\top = \begin{bmatrix} \lvert\mathcal{V}\rvert^{-1} & \cdots & \lvert\mathcal{V}\rvert^{-1} \end{bmatrix}_{1\times\lvert\mathcal{V}\rvert}$
$\mathbf{v}^\top$	$\mathbf{v}^\top = \begin{bmatrix} \lvert\Omega\rvert^{-1} & \cdots & \lvert\Omega\rvert^{-1} \end{bmatrix}_{1\times\lvert\Omega\rvert}$
$U(\mathbf{u}, \mathbf{v})$	$U(\mathbf{u}, \mathbf{v}) = \{P \in \mathbb{R}_+^{\lvert\mathcal{V}\rvert\times\lvert\Omega\rvert} : P\mathbf{1}_{\lvert\Omega\rvert} = \mathbf{u}, P^\top\mathbf{1}_{\lvert\mathcal{V}\rvert} = \mathbf{v}\}$
$\langle\cdot,\cdot\rangle$	Frobenius dot product
P_{ij}^Ω	the amount of assignment from node $i \in \mathcal{V}$ to cluster $j \in \Omega$
λ	scalar for entropy regularization
B	initial value $e^{-\lambda M}$
U	diagonal matrix
V	diagonal matrix
T	number of iterations for Sinkhorn-Knopp algorithm
$\mathbf{k}$	$\mathbf{k}^\top = \begin{bmatrix} \lvert\Omega\rvert\mathbf{1}_{\lvert\Omega\rvert}; \lvert\Gamma_i\rvert\mathbf{1}_{\lvert\Gamma_i\rvert}; \cdots; \lvert\Gamma_{\lvert\mathcal{V}\rvert}\rvert\mathbf{1}_{\lvert\Gamma_i\rvert} \end{bmatrix}_{1\times\lvert\mathcal{C}\rvert}$
τ	class τ for our node classification task
τ'	class that is not class τ
Ω^τ	set of global cluster-nodes associated with class τ
L	number of `DC-MsgPassing` iterations
c_j^τ	the j^{th} cluster embedding associated with class τ
z_i^L	embeddings for node i after L iterations
γ	a constant. $\gamma = (\alpha N^{-1} + \beta + 1 - \alpha)^{-1}$
$\hat{P}^{\Gamma_u}$	$\hat{P}^{\Gamma_u} \in \mathbb{R}_+^{\lvert\mathcal{V}\rvert\times\lvert\Gamma_u\rvert}$ broadcasted local cluster assignment matrix from $P^{\Gamma_u} \in \mathbb{R}_+^{\lvert\mathcal{N}_i^+\rvert\times\lvert\Gamma_i\rvert}$, defined in Eq.(13)
$\lVert\cdot\rVert_F$	Frobenius norm
$\Lambda(\cdot)$	similarity function
$\mathcal{V}_+$	set of training nodes

C. Proof and derivation

C.1. Proof of Theorem 2.1

To prove Theorem 2.1, we first introduce the following Lemma.

Lemma C.1. *Let $\mathcal{L}^{\lambda}_{cluster}$ be the loss function optimized by the* `DC-MsgPassing` *algorithm when the optimal transport components are replaced by the entropic regularized versions. Then, $\mathcal{L}^{\lambda}_{cluster}$ is lower-bounded by:*

$$\mathcal{L}^{\lambda}_{\text{cluster}} \geq \frac{\alpha}{\lambda} \log \frac{1}{|\mathcal{V}||\Omega|} + \frac{1-\alpha}{\lambda} \sum_{i \in \mathcal{V}} \log \frac{1}{|\mathcal{N}_i^+||\Gamma_i|}.$$

Proof. Recall that Ω is the set of global cluster-nodes, and Γ_i refers to the set of local cluster-nodes associated with a node i. $\mathcal{V}$ is a direct copy of the nodes V from the original graph. $P^{\Omega} \in \mathbb{R}^{|\mathcal{V}| \times |\Omega|}$ and $P^{\Gamma_i} \in \mathbb{R}^{|\mathcal{N}_i^+| \times |\Gamma_i|}$ are the global and local soft cluster assignment matrices respectively. $\mathcal{N}_i^+$ refers to the ego neighborhood of node i.

Let $p(P^{\Omega})$ be the probability of assignment matrix P^{Ω} and $h(P^{\Omega})$ be its entropy, we can upper bound its entropy by:

$$\begin{aligned} h(P^{\Omega}) &= \mathbb{E}\left[\log \frac{1}{p(P^{\Omega})}\right] \\ &\leq \log \mathbb{E}\left[\frac{1}{p(P^{\Omega})}\right] && \text{(by Jensen's inequality)} \\ &= \log \sum_{i \in \mathcal{V}, j \in \Omega} p(P^{\Omega}_{ij}) \frac{1}{p(P^{\Omega}_{ij})} && (7) \\ &= \log \big(|\mathcal{V}| \times |\Omega|\big). && (8) \end{aligned}$$

The inequality is due to uniform distribution having the maximum entropy.

Similarly, for local assignment matrices P^{Γ_i}, we have:

$$h(P^{\Gamma_i}) \leq \log \big(|\mathcal{N}_i^+| \times |\Gamma_i|\big). \tag{9}$$

Note that distance $d(\cdot, \cdot) \geq 0$, with the above results we obtain:

$$\begin{aligned} \mathcal{L}^{\lambda}_{\text{cluster}} &= \alpha \left(\sum_{i \in \mathcal{V}} \sum_{j \in \Omega} P^{\Omega}_{ij} d(z_i, c^{\Omega}_j) - \frac{1}{\lambda} h(P^{\Omega}) \right) + \beta \sum_{i \in \mathcal{V}} d(z_i, x_i) && (10) \\ &\quad + (1-\alpha) \sum_{i \in \mathcal{V}} \left(\sum_{k \in \mathcal{N}_i^+} \sum_{j \in \Gamma_i} P^{\Gamma_i}_{kj} d(z_k, c^{\Gamma_i}_j) - \frac{1}{\lambda} h(P^{\Gamma_i}) \right) && (11) \\ &\geq -\frac{\alpha}{\lambda} h(P^{\Omega}) - \frac{(1-\alpha)}{\lambda} \sum_{i \in \mathcal{V}} h(P^{\Gamma_i}) && \text{(by } d(\cdot,\cdot) \geq 0) \\ &\geq -\frac{\alpha}{\lambda} \log \big(|\mathcal{V}| \times |\Omega|\big) - \frac{(1-\alpha)}{\lambda} \sum_{i \in \mathcal{V}} \log \big(|\mathcal{N}_i^+| \times |\Gamma_i|\big) && \text{(by (8) and (9))} \\ &= \frac{\alpha}{\lambda} \log \frac{1}{|\mathcal{V}| \times |\Omega|} + \frac{(1-\alpha)}{\lambda} \sum_{i \in \mathcal{V}} \log \frac{1}{|\mathcal{N}_i^+| \times |\Gamma_i|}, && (12) \end{aligned}$$

which concludes the proof. □

By Lemma C.1, there exists a lower bound of $\mathcal{L}^{\lambda}_{cluster}$. Therefore, to prove the convergence of our algorithm, we only need to show that the loss function is guaranteed to decrease monotonically in each iteration until convergence for the assignment update step and for the embeddings update step.

For the assignment update step, let P^{Ω} be the current assignment from the previous iteration and $P^{\Omega *}$ be the new assignment obtained as

$$P^{\Omega *} \in \underset{P^{\Omega} \in U(\mathbf{r}, \mathbf{c})}{\arg\min} \langle P^{\Omega *}, M \rangle - \frac{1}{\lambda} h(P^{\Omega *}).$$

The change in the loss function after this assignment step is then given by

$$\mathcal{L}^{\lambda}_{\text{cluster}}(P^{\Omega *}) - \mathcal{L}^{\lambda}_{\text{cluster}}(P^{\Omega}) \leq 0,$$

where the inequality holds by the convergence of Sinkhorn–Knopp algorithm [20], [21], [59]–[61]. Similarly, we have

$$\mathcal{L}^{\lambda}_{\text{cluster}}(P^{\Gamma_i *}) - \mathcal{L}^{\lambda}_{\text{cluster}}(P^{\Gamma_i}) \leq 0.$$

For the embeddings update step, let Z and C be the current embeddings from the previous iteration, and Z^* and C^* be the new embeddings obtained by Eq. (5) and Eq. (6). Let the $d(\cdot, \cdot)$ be the squared Euclidean distance, then since P is a positive constant in this step, the loss function is convex. Eq. (5) and Eq. (6) are closed form solutions to the loss function, then we have

$$\mathcal{L}^{\lambda}_{\text{cluster}}(Z^*, C^*) - \mathcal{L}^{\lambda}_{\text{cluster}}(Z, C) \leq 0.$$

Since $\mathcal{L}^{\lambda}_{cluster}$ has a lower bound and it decreases monotonically in each iteration, the value of $\mathcal{L}^{\lambda}_{cluster}$ produced by the message passing algorithm is guaranteed to converge.

C.2. Broadcasted assignment matrices

Let $f\colon \mathcal{Z} \times \mathcal{Z} \to \mathcal{Z}$ be a mapping from the index k of a node in the ego-neighborhood of the node u to the index i of the same node in the node set $\mathcal{V}$. Let $P^{\Gamma_u} \in \mathbb{R}_+^{|\mathcal{N}_u^+| \times |\Gamma_u|}$ be the local assignment matrix for the ego-neighborhood of node $u \in \mathcal{V}$. For any $u \in \mathcal{V}$, we define the broadcasted local assignment matrix $\hat{P}^{\Gamma_u} \in \mathbb{R}_+^{|\mathcal{V}| \times |\Gamma_u|}$ as

$$\hat{P}^{\Gamma_u}_{ij} = \begin{cases} P^{\Gamma_u}_{kj}, & \text{if } i = f(u, k) \\ 0, & \text{otherwise} \end{cases}. \tag{13}$$

Then, we can define the the overall assignment matrix $P \in \mathbb{R}_+^{|\mathcal{V}| \times |\mathcal{C}|}$, where $|\mathcal{C}| = |\Omega| + |\Gamma|$ and $|\Gamma| = \sum_{i \in \mathcal{V}} |\Gamma_i|$, as

$$P = \begin{bmatrix} P^{\Omega} & \hat{P}^{\Gamma_1} & \dots & \hat{P}^{\Gamma_{|\mathcal{V}|}} \end{bmatrix}. \tag{14}$$

Note that each element P_{ij} can be viewed as the edge weights of a node $i \in \mathcal{V}$ and a specific cluster-node $j \in \mathcal{C}$. In simple words, P includes all the global and local cluster assignment matrices, collated in a single matrix. This allows us to unify the message passing update in Eq.(6) for both local and global clustering terms.

C.3. Update for cluster-node embeddings C: derivation of Eq. (5)

Without loss of generality, we provide the full derivation for global cluster-node embeddings update function. With squared Euclidean norm as the distance function, *i.e.*, $d(u, v) = \|u - v\|^2$, we can derive that

$$\frac{\partial \mathcal{L}^{\lambda}_{\text{cluster}}}{\partial c^{\Omega}_j} = \frac{\partial}{\partial c^{\Omega}_j} \sum_{i \in \mathcal{V}} P^{\Omega}_{ij} \|z_i - c^{\Omega}_j\|^2 = 0$$

Then we have

$$2 \sum_{i \in \mathcal{V}} P^{\Omega}_{ij} (c^{\Omega}_j - z_i) = 0$$

By rearranging the terms

$$\sum_{i\in\mathcal{V}} P^{\Omega}_{ij} c^{\Omega}_j = \sum_{i\in\mathcal{V}} P^{\Omega}_{ij} z_i$$

Then one has

$$c^{\Omega}_j = \frac{\sum_{i\in\mathcal{V}} P^{\Omega}_{ij} z_i}{\sum_{i\in\mathcal{V}} P^{\Omega}_{ij}}$$

Since $P^{\Omega} \in U(\mathbf{u}, \mathbf{v})$, we have $\sum_i P^{\Omega}_{ij} = \frac{1}{|\Omega|}$ where $|\Omega|$ is the number of global clusters. Therefore,

$$c^{\Omega}_j = |\Omega| \sum_{i\in\mathcal{V}} P^{\Omega}_{ij} z_i \tag{15}$$

Similarly, with $|\Gamma_i|$ denoting the number of local clusters within the ego-neighborhood of node i, we have

$$c^{\Gamma_i}_j = |\Gamma_i| \sum_{u\in\mathcal{N}^+_i} P^{\Gamma_i}_{ij} z_u \tag{16}$$

Let $\mathbf{k}^\top = \left[|\Omega|\mathbf{1}_{|\Omega|}; |\Gamma_i|\mathbf{1}_{|\Gamma_i|}; \cdots; |\Gamma_{|\mathcal{V}|}|\mathbf{1}_{|\Gamma_i|}\right]_{1\times|\mathcal{C}|}$, where ; denotes concatenation. Then we can combine Eq. (15) and Eq. (16) together in one matrix equation,

$$C = \text{diag}(\mathbf{k}) P^\top Z, \tag{17}$$

where the overall assignment matrix P is defined in Eq. (14),

C.4. Update for node embeddings Z: derivation of Eq. (6)

We provide the full derivation of the node embeddings update function. This manifests as message passing from cluster-nodes to nodes. With squared Euclidean norm as the distance function, *i.e.*, $d(u, v) = \|u - v\|^2$, we can derive that

$$\frac{\partial \mathcal{L}^{\lambda}_{\text{cluster}}}{\partial z_i} = \alpha \sum_{j\in\Omega} 2P^{\Omega}_{ij}(z_i - c_j) + 2\beta(z_i - x_i) + (1-\alpha) \sum_{u\in\mathcal{N}^+_i} \sum_{j\in\Gamma_u} 2P^{\Gamma_u}_{ij}(z_i - c_j) = 0 \tag{18}$$

From Eq. (18), we obtain

$$\alpha \sum_{j\in\Omega} P^{\Omega}_{ij} z_i - \alpha \sum_{j\in\Omega} P^{\Omega}_{ij} c_j + \beta z_i - \beta x_i + (1-\alpha) \sum_{u\in\mathcal{N}^+_i} \sum_{j\in\Gamma_u} P^{\Gamma_u}_{ij} z_i - (1-\alpha) \sum_{u\in\mathcal{N}^+_i} \sum_{j\in\Gamma_u} P^{\Gamma_u}_{ij} c_j = 0$$

By rearranging the terms, we have

$$(\alpha \sum_{j\in\Omega} P^{\Omega}_{ij} + \beta + (1-\alpha) \sum_{u\in\mathcal{N}^+_i} \sum_{j\in\Gamma_u} P^{\Gamma_u}_{ij}) z_i = \alpha \sum_{j\in\Omega} P^{\Omega}_{ij} c_j + \beta x_i + (1-\alpha) \sum_{u\in\mathcal{N}^+_i} \sum_{j\in\Gamma_u} P^{\Gamma_u}_{ij} c_j$$

Since $P^{\Omega} \in U(\mathbf{u}, \mathbf{v})$, we have $\sum_j P^{\Omega}_{ij} = \frac{1}{|\mathcal{V}|}$. Similarly, $\sum_{j\in\Gamma_u} P^{\Gamma_u}_{ij} = \frac{1}{|\mathcal{N}^+_i|}$. Therefore, we can deduce that

$$(\frac{\alpha}{|\mathcal{V}|} + \beta + (1-\alpha) \sum_{u\in\mathcal{N}^+_i} \frac{1}{|\mathcal{N}^+_i|}) z_i = \alpha \sum_{j\in\Omega} P^{\Omega}_{ij} c_j + \beta x_i + (1-\alpha) \sum_{u\in\mathcal{N}^+_i} \sum_{j\in\Gamma_u} P^{\Gamma_u}_{ij} c_j$$

With $\sum_{u\in\mathcal{N}^+_i} \frac{1}{|\mathcal{N}^+_i|} = 1$, we have

$$(\frac{\alpha}{|\mathcal{V}|} + \beta + 1 - \alpha) z_i = \alpha \sum_{j\in\Omega} P^{\Omega}_{ij} c_j + \beta x_i + (1-\alpha) \sum_{u\in\mathcal{N}^+_i} \sum_{j\in\Gamma_u} P^{\Gamma_u}_{ij} c_j$$

This leads to

$$z_i = \frac{1}{\frac{\alpha}{|\mathcal{V}|} + \beta + 1 - \alpha}[\alpha \sum_{j\in\Omega} P_{ij}^{\Omega} c_j + \beta x_i + (1-\alpha) \sum_{u\in\mathcal{N}_i^+} \sum_{j\in\Gamma_u} P_{ij}^{\Gamma_u} c_j]$$

Finally, expressing in matrix form admits the following closed-form solution

$$Z = \frac{1}{\frac{\alpha}{|\mathcal{V}|} + \beta + 1 - \alpha}[\alpha P^{\Omega} C^{\Omega} + \beta X + (1-\alpha) \sum_{u\in\mathcal{N}_i^+} \hat{P}^{\Gamma_u} C^{\Gamma_u}]$$

where $\hat{P}^{\Gamma_u} \in \mathbb{R}_+^{|\mathcal{V}|\times|\Gamma_u|}$ is the broadcasted local cluster assignment matrix from $\hat{P}^{\Gamma_u} \in \mathbb{R}_+^{|\mathcal{N}_u^+|\times|\Gamma_u|}$ as defined in Eq.(13).

D. Illustration of bipartite graph formulation

We construct a bipartite graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{C}, \mathcal{E})$. The graph is derived from the original graph $G = (V, E)$ and comprises two distinct sets of nodes. The first set, $\mathcal{V}$, is a direct copy of the nodes V from the original graph. The second set, $\mathcal{C}$, consists of cluster nodes divided into two categories: global clusters (Ω) and local clusters (Γ).

In this bipartite graph, each node from the global clusters Ω connects to all nodes in $\mathcal{V}$. Meanwhile, each node from the local clusters Γ is associated with a specific node i in $\mathcal{V}$, and connected to its ego-neighborhood, which includes node i and its one-hop neighbors. For a node i in $\mathcal{V}$, Γ_i represents the set of local clusters associated with it. The total number of nodes in $\mathcal{C}$ is the sum of nodes in Ω and the nodes in all local clusters *i.e.*, $|\mathcal{C}| = |\Omega| + \sum_{i\in\mathcal{V}} |\Gamma_i|$. An illustration of the bipartite graph is provided in Fig. 2.

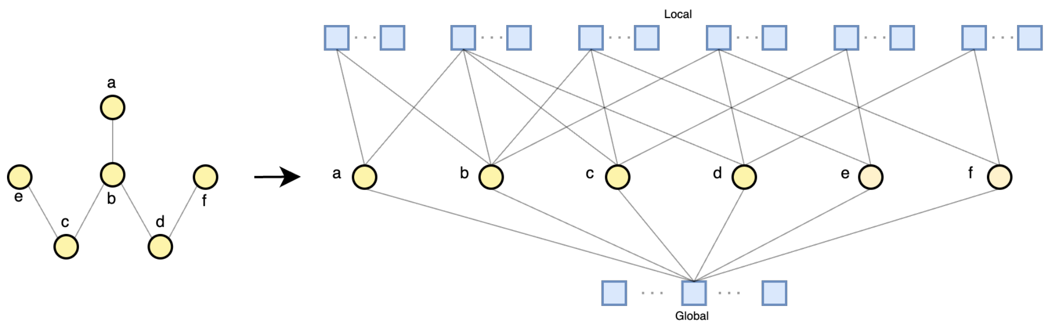


Figure 2: Based on the original graph on the left, we construct a bipartite graph on the right by adding local and global cluster-nodes. For each node in the original graph, a set of local cluster-nodes, represented by the blue boxes at the top, is connected to its ego-neighborhood. For example, the ego-neighborhood of node a includes itself and its one-hop neighbor node b. Therefore the local cluster-nodes for node a are connected to a and b. Meanwhile, a set of global cluster-nodes are added and connected to all nodes in the original graph, as represented by the blue boxes at the bottom.

E. More on training loss

E.1. Orthogonality loss ($\mathcal{L}_{\text{ortho}}$)

To encourage the clusters to be distinct, we adopt a regularizing orthogonality loss function [43] $\mathcal{L}_{\text{ortho}} = \left\| \frac{C^\top C}{\|C^\top C\|_F} - \frac{I_{|\Omega|}}{\sqrt{|\Omega|}} \right\|_F$, where $\|\cdot\|_F$ is the Frobenius norm. This pushes the cluster-nodes to be orthogonal to each other.

E.2. Similarity loss ($\mathcal{L}_{\text{sim}}$)

To further enhance the clustering process, we introduce $\mathcal{L}_{\text{sim}}$ that encourages node similarity to only a single cluster. To achieve this, we set $|\Omega|$ to be multiple of the number of classes and associate a set of cluster-nodes Ω^τ with each class τ. We then compute distances between the node embedding and the cluster-node embeddings associated with its labelled class, select the cluster-node embedding that is most similar to the node with a max operator, and push them closer. If a training node i belongs to class τ, c_j^τ is the j^{th} cluster embedding associated with class τ. Let Λ be a similarity function, and $\mathcal{V}_+$ be the set of training nodes, then $\mathcal{L}_{\text{sim}}$ is defined as

$$\mathcal{L}_{\text{sim}} = \frac{1}{|\Omega||\mathcal{V}_+|} \sum_{i \in \mathcal{V}_+} \Big[s_i^\tau + \log \sum_{\tau' \neq \tau} \exp\big(- s_i^{\tau'} \big) \Big], \tag{19}$$

$$\text{where } s_i^\tau = \max_{j \in |\Omega^\tau|} \Lambda\Big(z_i^L, c_j^\tau \Big). \tag{20}$$

F. More on DC-MsgPassing Algorithm

Algorithm of `DC-MsgPassing` can be found in Algo. 1. Each iteration of `DC-MsgPassing` consists of two alternative steps. First, with fixed embeddings Z and C, optimal clustering assignment matrices are calculated via Sinkhorn-Knopp algorithm (Eq. (4)). Then, the cluster-node and node embeddings are refined through message passing with the updated assignment matrices via Eq. (5) and Eq. (6). In practice, we could also add learnable components such as linear transformation matrices or MLPs for the messages in Eq. (5) and Eq. (6) to allow the network to fit the data distribution better.

Algorithm 1 `DC-MsgPassing`

```
Input: Bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{C}, \mathcal{E})$, Node features $X$, hyperparameters $\alpha$, $\beta$, $\lambda$
Output: $[z_i]_{i \in \mathcal{V}}$
 1: $Z = X$
 2: Initialize cluster embeddings $C$
 3: // Optimize $\mathcal{L}^\lambda_{\text{cluster}}$ via DC-MsgPassing
 4: for $l = 1, 2, \ldots, L$ do
 5:     Update Cluster Assignment Matrices:
 6:         $M_{ij} = d(z_i, c_j) \quad \forall i \in \mathcal{V}, j \in \mathcal{C}$
 7:         $B^{\{\Omega, \Gamma_i\}} = e^{-\lambda M}$
 8:         // Run Sinkhorn algorithm for $T$ steps (Eq. 4)
 9:         $P^{\{\Omega, \Gamma_i\}} = B^{\{\Omega, \Gamma_i\}}$
10:     Update Node and Cluster-node Embeddings:
11:         Calculate $C$ as per Eq. (5)
12:         Calculate $Z$ as per Eq. (6)
13: end for
14: Return: Z
```

F.1. Complexity analysis

The complexity of the global assignment update step is $O(T|\mathcal{V}||\Omega|) = O(|\mathcal{V}|)$ as $|\Omega| \ll |\mathcal{V}|$ and T are small constants. Both the complexity of the local assignment update step $O(T|\mathcal{E}|) = O(|E|)$ and the embeddings update step $O(|\mathcal{E}|) = O(|E|)$ are linear w.r.t. the number of edges. Thus the overall computational complexity is linear w.r.t. the size of the original graph $O(|E|)$, same order as standard GNNs. Runtime of `DC-MsgPassing` is measured in Appendix G.5.

G. Experiments

G.1. More results on heterophilous datasets

DC-GNN achieves state-of-the-art on four out of five heterophilous datasets proposed by [62], as shown in Tab. 3. Our performance is especially strong on Minesweeper, where we surpass existing baselines by 5%.

G.2. Benefit of capturing long range information in sparse label settings

Our method introduces shortcuts between distant nodes via the cluster-nodes. To empirically evaluate the effects of shortcut construction, we consider a generalized scenario on homophilous graphs where information from labeled nodes needs to propagate over a long distance to reach most unlabeled nodes. Specifically, we evaluate our method with only a small number of training labels per class.

Table 4: Classification accuracy with 5 labels per class.

	CORA	CITESEER	PUBMED
GCN	69.23 (3.39)	63.03 (4.48)	68.00 (3.75)
DC-GNN	**72.17** (1.76)	62.14 (2.65)	**75.07** (0.82)

Our hypothesis is that information propagation becomes more challenging when useful training information is more scarce, making the effects of shortcut construction more pronounced in sparsely-annotated graph datasets. The hypothesis is confirmed by experiment results in Tab. 4, where our method outperforms vanilla GCN on Cora and Pubmed with substantial improvements.

Table 3: Classification performance comparison on more recent heterophilous datasets [62]. Following [62], we report accuracy for Roman-empire and Amazon-ratings, and ROC AUC for the rest.

	Roman-empire	Amazon-ratings	Minesweeper	Tolokers	Questions
ResNet	65.88 (0.38)	45.90 (0.52)	50.89 (1.39)	72.95 (1.06)	70.34 (0.76)
ResNet+SGC	73.90 (0.51)	50.66 (0.48)	70.88 (0.90)	80.70 (0.97)	75.81 (0.96)
ResNet+adj	52.25 (0.40)	51.83 (0.57)	50.42 (0.83)	78.78 (1.11)	75.77 (1.24)
GCN	73.69 (0.74)	48.70 (0.63)	89.75 (0.52)	83.64 (0.67)	76.09 (1.27)
SAGE	85.74 (0.67)	53.63 (0.39)	93.51 (0.57)	82.43 (0.44)	76.44 (0.62)
GAT	80.87 (0.30)	49.09 (0.63)	92.01 (0.68)	83.70 (0.47)	77.43 (1.20)
GAT-sep	88.75 (0.41)	52.70 (0.62)	93.91 (0.35)	83.78 (0.43)	76.79 (0.71)
GT	86.51 (0.73)	51.17 (0.66)	91.85 (0.76)	83.23 (0.64)	77.95 (0.68)
GT-sep	87.32 (0.39)	52.18 (0.80)	92.29 (0.47)	82.52 (0.92)	78.05 (0.93)
GPS$^{\text{GAT+Performer}}$	87.04 (0.58)	49.92 (0.68)	91.08 (0.58)	84.38 (0.91)	77.14 (1.49)
H$_2$GCN	60.11 (0.52)	36.47 (0.23)	89.71 (0.31)	73.35 (1.01)	63.59 (1.46)
CPGNN	63.96 (0.62)	39.79 (0.77)	52.03 (5.46)	73.36 (1.01)	65.96 (1.95)
GPR-GNN	64.85 (0.27)	44.88 (0.34)	86.24 (0.61)	72.94 (0.97)	55.48 (0.91)
FSGNN	79.92 (0.56)	52.74 (0.83)	90.08 (0.70)	82.76 (0.61)	78.86 (0.92)
GloGNN	59.63 (0.69)	36.89 (0.14)	51.08 (1.23)	73.39 (1.17)	65.74 (1.19)
FAGCN	65.22 (0.56)	44.12 (0.30)	88.17 (0.73)	77.75 (1.05)	77.24 (1.26)
GBK-GNN	74.57 (0.47)	45.98 (0.71)	90.85 (0.58)	81.01 (0.67)	74.47 (0.86)
JacobiConv	71.14 (0.42)	43.55 (0.48)	89.66 (0.40)	68.66 (0.65)	73.88 (1.16)
DC-GNN	**89.96** (0.35)	51.11 (0.47)	**98.50** (0.21)	**85.88** (0.81)	**78.96** (0.60)

G.3. Alleviating oversquashing

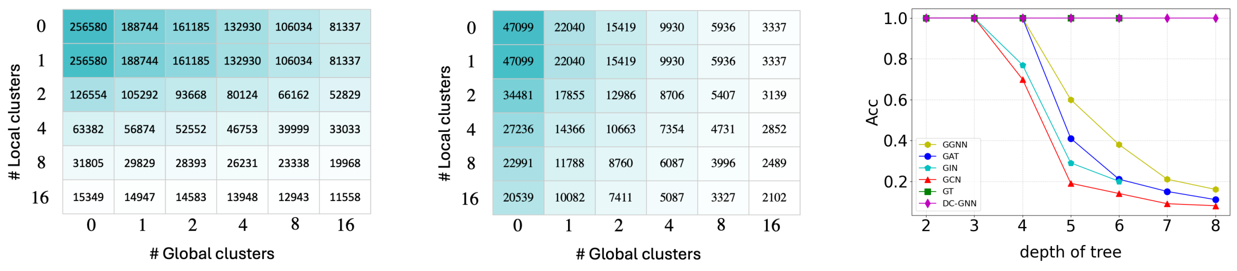
G.3.1. Definition of effective resistance

Let u and v be vertices of G. The effective resistance between u and v is defined as

$$R_{u,v} = (1_u - 1_v)^T L^+ (1_u - 1_v),$$

where 1_v is the indicator vector of the vertex v [15]. Let A be the adjacency matrix and D be the degree matrix. The Laplacian is $L = D - A$ and L^+ is the pseudoinverse of L. The total effective resistance (R_{tot}) of a graph is therefore the total sum of effective resistance between every pair of nodes.

G.3.2. Experiments to validate oversquashing mitigation

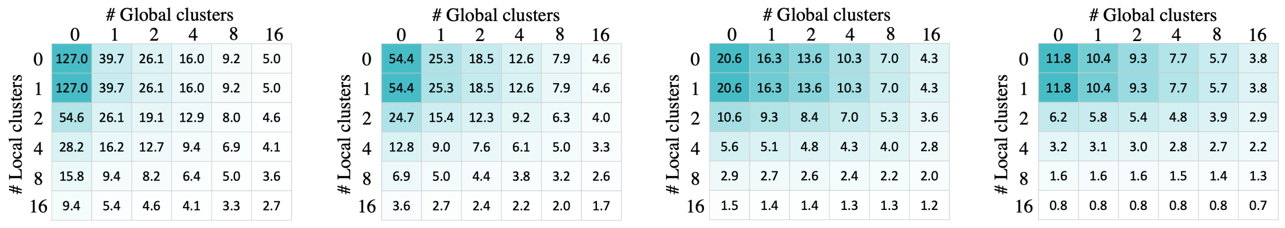


(a) Amherst41 (b) Wisconsin (c) Tree-NeighborsMatch

Figure 3: (a)-(b) Total effective resistance heatmap. (c) Accuracy for Tree-NeighborsMatch dataset.

Total effective resistance (R_{tot}) is established as an indicator of oversquashing [15], prompting the development of various graph rewiring strategies to diminish R_{tot} within the underlying graph and thus address oversquashing. Our approach contributes to this endeavor by introducing cluster-nodes. This effectively creates new pathways among the original nodes, thereby reducing the graph's R_{tot} and aiding in mitigating the oversquashing issue [15]. To validate this, we conduct an empirical analysis of the total pairwise effective resistance among the original nodes in our bipartite graph, with varying number of global and local clusters. Fig. 3(a) and Fig. 3(b) display a heatmap of R_{tot}, with darker shades representing higher R_{tot} values. The results indicate that R_{tot} decreases sharply as we increase the number of both global and local clusters.

To further validate the oversquashing mitigation capability, we conduct experiments on Tree-NeighborsMatch dataset [9], which requires long-range interaction between leaf nodes and the root node of tree graphs with varying depths. In Fig. 3(c), DC-GNN achieves perfect performance along with GT [63] on all depth settings, significantly outperforming other message passing GNNs.



(a) edge probability = 0.2 (b) edge probability = 0.3 (c) edge probability = 0.5 (d) edge probability = 0.8

Figure 4: Total effective resistance heatmap of Erdos-Renyi random graphs at different sparsity levels. Number of nodes is 10 for all settings.

We also measure effective resistance (R_{tot}) in synthetic random graphs with different degrees of sparsity. Results in Fig. 4 show that both global and local cluster-nodes contribute to reducing effective resistance, as demonstrated by decreasing R_{tot} values in both row and column directions. Additionally, the more drastic R_{tot} decrease from the first to last column in heatmap `(a)` compared to heatmap `(d)` show that global cluster-nodes play a more pronounced role in reducing effective resistance at a higher edge sparsity setting.

G.4. Ablation studies

G.4.1. Effects of each term in $\mathcal{L}^{\lambda}_{\text{cluster}}$

To validate the effectiveness of our `DC-MsgPassing` algorithm, we conduct an ablation study on the individual components of our objective function $\mathcal{L}^{\lambda}_{\text{cluster}}$. Specifically, we vary the parameters by setting (1) α to 0, (2) α to 1 and (3) β to 0, aiming to ablate the contributions of the global clustering term, local clustering term and the node fidelity term respectively.

Table 5: Effects of each term in $\mathcal{L}^{\lambda}_{\text{cluster}}$.

	Genius	US-election	Penn94	Amherst41
DC-GNN	91.70 (0.08)	89.59 (1.60)	86.69(0.22)	82.94 (1.59)
(-)Global	91.62 (0.07)	88.77 (2.21)	84.61 (0.42)	81.43 (1.53)
(-)Local	87.05 (0.09)	83.26 (1.77)	86.69 (0.22)	80.77 (2.04)
(-)Fidelity	91.08 (0.04)	87.84 (2.67)	86.69 (0.22)	82.28 (1.32)

Results from Tab. 5 indicate that the contributions of global and local clustering vary on different datasets. Specifically, the contribution of local clustering is dominant on Genius, US-election and Amherst41. This is expected as local clustering facilitates message passing via adjacent nodes and embeds graph structure information into the model. The contribution of global clustering is most pronounced on Penn94, indicating the usefulness of long-range information in Penn94 captured by global clustering term. Additionally, The results show that all three terms—local clustering, global clustering, and node fidelity—contribute to the overall efficacy of our model.

G.4.2. Effects of $\mathcal{L}_{\text{ortho}}$ and $\mathcal{L}_{\text{sim}}$

Table 6: Effects of $\mathcal{L}_{\text{ortho}}$ and $\mathcal{L}_{\text{sim}}$.

	Genius	US-election	Penn94	Amherst41
DC-GNN	91.70 (0.08)	89.59 (1.60)	86.69 (0.22)	82.94 (1.59)
(-) $\mathcal{L}_{\text{sim}}$	91.70 (0.08)	89.08 (1.46)	86.65 (0.15)	82.22 (1.03)
(-) $\mathcal{L}_{\text{ortho}}$	91.68 (0.08)	88.72 (1.20)	86.64 (0.27)	82.35 (1.25)
(-)$\mathcal{L}_{\text{sim}}$, $\mathcal{L}_{\text{ortho}}$	91.68 (0.08)	88.61 (1.57)	86.40 (0.25)	81.75 (1.07)

We introduced *orthogonality* ($\mathcal{L}_{\text{ortho}}$) and *similarity* ($\mathcal{L}_{\text{sim}}$) losses to regularize and assist the clustering process. In this ablation, we evaluate the effects of these losses on model performance. As shown in Tab. 6, $\mathcal{L}_{\text{ortho}}$ and $\mathcal{L}_{\text{sim}}$ generally help to improve the scores across datasets. The results provide some evidence that the two losses indeed facilitate the clustering process, plausibly by encouraging distinct cluster representations and node-cluster alignment as hypothesized.

G.4.3. Aggregation operation in similarity loss

After computing the similarity between a node and each of the multiple clusters from the same class, the choice of aggregation method is crucial. We evaluate the effectiveness of using the aggregation operator on Amherst41 and Penn94 datasets. Tab. 7 shows the effects of replacing the $\max$ aggregator with mean and sum in computing similarity loss. On both datasets, $\max$ outperforms both sum and mean, indicating the effectiveness of using $\max$ as the aggregation operation. Intuitively, taking the average of all similarity scores (mean) is sub-optimal. mean tends to make the node embeddings closer to the average of all clusters belonging to a same class, undermining the purpose of using multiple clusters . Similar to mean, summing up all similarity scores (sum) is more powerful yet requires more data to learn. $\max$ selects the maximum similarity score to compute similarity loss and guides the node embeddings closer to one of the clusters, thus preserving the power of diversity in representation.

Table 7: Effects of aggregator function in $\mathcal{L}_{sim}$.

AGG	**PENN94**	**AMHERST41**
MEAN	86.13 (0.12)	81.23 (1.34)
SUM	85.84 (0.26)	81.26 (1.58)
MAX	86.69(0.22)	82.94 (1.59)

G.4.4. EFFECTS OF NODE FIDELITY TERM

Table 8: Dirichlet Energy (DE) with different β values. Higher DE indicates increased node distinctiveness.

	$\beta = 0.0$	$\beta = 0.5$	$\beta = 1.0$
Wisconsin	0.741288	0.963261	0.978465
Citeseer	0.110083	0.151699	0.206022
Cora	0.218658	0.294024	0.330234

The node fidelity term encourages the node embeddings to retain some information from the original node features, which serve as initial residual. This technique can also potentially help to alleviate oversmoothing as shown in [16], [17]. To validate this, we conduct additional experiments to measure the normalized Dirichlet Energy (DE) [27] for DC-GNN on Wisconsin, Cora and Citeseer, using the implementation from [27].

We set β to 0, 0.5 and 1 for each dataset to measure how increased weightage of the node fidelity term influences DE, while keeping α constant at 0.5. As observed in Tab. 8, DE positively correlates with β on all datasets.

G.5. Runtime experiments

We measure the average runtime of a `DC-MsgPassing` layer on the three largest datasets used in our experiments against Pytorch Geometric [64] implementation of GATConv and GCNConv. `DC-MsgPassing` takes less than 4x times GCN and is faster than GAT on these datasets. The results show that `DC-MsgPassing` is competitive in terms of runtime, confirming our complexity analysis.

Table 9: Runtime and dataset statistics comparison on three large-scale datasets. Runtime results are in seconds.

	Penn94	**Cornell5**	**Genius**
# Nodes	41,554	18,660	421,961
# Edges	1,362,229	790,777	984,979
`DC-MsgPassing`	0.00852	0.00820	0.00708
GATConv	0.01242	0.01685	0.01636
GCNConv	0.00220	0.00242	0.00207
Multiples of GAT	0.69x	0.49x	0.43x
Multiples of GCN	3.88x	3.38x	3.42x

G.6. Dataset details

We conduct experiments on fourteen datasets, a mix of small-scale and large-scale datasets. Eleven of them are non-homophilous, including: (1) Roman-empire, Amazon-ratings, Minesweeper, Tolokers, Questions [62]; (2) Penn94, Genius, Cornell5, Amherst41 [39]; (3) Wisconsin [37]; (4) a US election dataset [65]. Three are homophilous citation networks: Cora, Citeseer and Pubmed [37]. We use the original train/validation/test splits when they exist. Otherwise we follow the splits specified in [17], [39], [62]. Descriptions and statistics of the datasets are below.

G.6.1. Dataset description

Roman-empire, Amazon-ratings, Minesweeper, Tolokers and Questions are five datasets proposed in [62] to better evaluate the performance of GNNs under heterophilous settings. The description of each dataset is as follows.

Roman-empire is based on the Roman Empire article from English Wikipedia. Each node in the graph represents one word in the text, and each edge between two words represents either one word following another word or if the two words are connected in the dependency tree. Node features is its FastText word embeddings. The task is to predict a node's syntactic role.

Amazon-ratings is based on the Amazon product co-purchasing network metadata. Nodes represent products and edges connect products frequently purchased together. Node features are the mean of FastText embeddings for words in product description. The task is to predict the class of products' ratings.

Minesweeper is a synthetic dataset inspired by the Minesweeper game. The graph is a regular 100x100 grid where each node is connected its eight neighboring nodes. 20% of the nodes are randomly assgined as mines. The node features are one-hot-encoded numbers of the neighboring mines. The task is to predict if the nodes are mines.

Tolokers is based on data from the Toloka crowdsourcing platform. Nodes represent workers who have participated in the selected projects, while edges connect two workers who work on the same task. Node features are based on worker's profile information and task performance. The task is to predict which workers have been banned.

Questions is based on question-answering data from website Yandex Q. Nodes represent users and edges connect an answer provider to a question provider. Node features are the mean of FastText word embeddings of user profile description, with an additional binary feature indicating users with no descriptions. The task is to predict if the users remain active on the website.

Penn94, Cornell5 and Amherst41 [39] are friendship network datasets extracted from Facebook of students from selected universities from 2005. Each node in the datasets represent a student, while node label represents the reported gender of the student. Node features include major, second major/minor, dorm/house, year, and high school.

Wisconsin [37] is a web page dataset collected from the computer science department of Wisconsin Madison. In this dataset, nodes represent web pages and edges are hyperlinks between them. Feature vectors of nodes are bag-of-words representations. The task is to classify the web pages into one of the five categories including student, project, course, staff and faculty.

Genius [39] is a sub-network from website genius.com, a crowd-sourced website of song lyrics annotations. Nodes represent users while edges connect users that follow each other. Node features include expertise scores expertise scores, counts of contributions and roles held by users. Around 20% of the users are marked with a "gone" label, indicating that they are more likely to be spam users. The task is to predict which users are marked.

US-election [65] is a geographical dataset extracted from statistics of Unite States election of year 2012. Nodes represent US counties, while edges connect bordering counties. Node features include income, education, population etc. The task is a binary classification to predict election outcome.

Cora, Citeseer and Pubmed [37] are citation graphs, where each node represents a scientific paper and two papers are connected when a paper cites the other. Each node is labeled with the research field and the task is to predict which field the paper belongs to. All three datasets are homophilous.

G.6.2. Dataset statistics

Tab. 10 covers statistics of datasets in Tab. 1. Tab. 11 covers statistics of datasets in Tab. 3.

Homophily matrix Homophily refers to the degree of similarity between connected neighboring nodes in terms of their features or labels. There are many types of homophily measures proposed, including edge homophily [11], node homophily [37], and improved edge homophily [39]. Homophily matrix proposed in [39] is an important metric, since it can better reflect class-wise homophily. The homophiliy matrix is defined as:

$$H_{c_1,c_2} = \frac{|(u,v) \in E : c_u = c_1,\ c_v = c_2|}{|(u,v) \in E : c_u = c_1|}, \tag{21}$$

Table 10: Dataset statistics for Tab. 1.

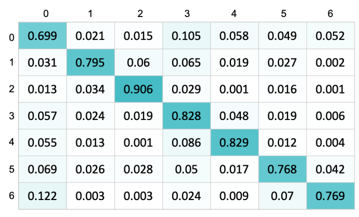
	Penn94	Cornell5	Amherst41	Genius	US-election	Wisconsin	Cora	Citeseer	Pubmed
Edge Hom.	0.47	0.47	0.46	0.61	0.83	0.21	0.81	0.74	0.80
Improved Edge Hom. [39]	0.046	0.09	0.05	0.08	0.54	0.094	0.766	0.627	0.664
# Nodes	41,554	18,660	2,235	421,961	3,234	251	2,708	3,327	19,717
# Edges	1,362,229	790,777	90,954	984,979	11,100	466	5,278	4,676	44,327
# Node Features	4814	4735	1193	12	6	1,703	1,433	3,703	500
# Classes	2	2	2	2	2	5	6	7	3

Table 11: Dataset statistics for Tab. 3.

	Roman-Empire	Amazon-Ratings	Minesweeper	Tolokers	Questions
Edge Hom.	0.05	0.38	0.68	0.59	0.84
Improved Edge Hom. [39]	0.01	0.12	0.009	0.17	0.08
# Nodes	22,662	24,492	10,000	11,758	48,921
# Edges	32,927	93,050	39,402	519,000	153,540
# Node Features	300	300	7	10	301
# Classes	18	5	2	2	2

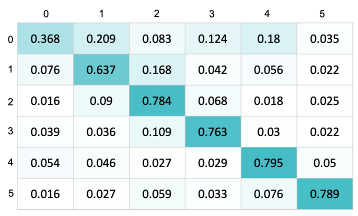
for classes c_1 and c_2, $H_{c_1 c_2}$ denotes the proportion of edges between from nodes of class c_1 to nodes of class c_2. A homophilous graph has high values on the diagonal entries of H.

Fig. 5 are the homophily matrices for three well-known homophilous datasets: Cora, Citeseer and Pubmed [66]. High homophily is signified by the high numbers in diagonal cells, whereas values of non-diagonal cells are mostly less than 0.1. This is different from the homophily matrices of heterophilous datasets, where values of non-diagonal cells are similar or even higher than diagonal cells.



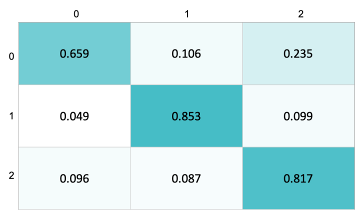
	0	1	2	3	4	5	6
0	0.699	0.021	0.015	0.105	0.058	0.049	0.052
1	0.031	0.795	0.06	0.065	0.019	0.027	0.002
2	0.013	0.034	0.906	0.029	0.001	0.016	0.001
3	0.057	0.024	0.019	0.828	0.048	0.019	0.006
4	0.055	0.013	0.001	0.086	0.829	0.012	0.004
5	0.069	0.026	0.028	0.05	0.017	0.768	0.042
6	0.122	0.003	0.003	0.024	0.009	0.07	0.769

(a) Cora



	0	1	2	3	4	5
0	0.368	0.209	0.083	0.124	0.18	0.035
1	0.076	0.637	0.168	0.042	0.056	0.022
2	0.016	0.09	0.784	0.068	0.018	0.025
3	0.039	0.036	0.109	0.763	0.03	0.022
4	0.054	0.046	0.027	0.029	0.795	0.05
5	0.016	0.027	0.059	0.033	0.076	0.789

(b) Citeseer



	0	1	2
0	0.659	0.106	0.235
1	0.049	0.853	0.099
2	0.096	0.087	0.817

(c) Pubmed

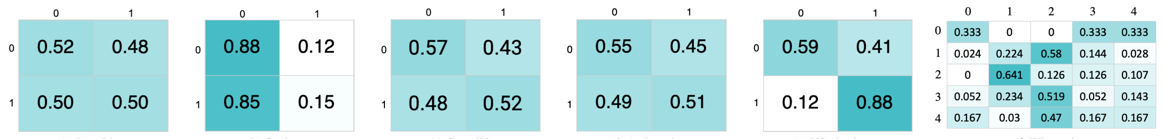
Figure 5: Homophily matrix for three homophilous datasets.

We show in Fig. 6 the homophily matrices for some heterophilous datasets for comparison.

G.7. Implementation details

Implementation details. For global cluster-nodes, we use trainable lookup embeddings to initialize the embeddings. For local cluster-nodes, we fix the number of local clusters to be 2 for every ego-neighborhoods, and initialize the two cluster-node embeddings by the central node features and the average of neighboring node features respectively. In practice, for local clustering cost matrices M, we rescale and normalize the distance before running the Sinkhorn-Knopp algorithm for numerical stability.

Training Settings We conduct each experiment of DC-GNN using three distinct data splits and present the corresponding mean and standard deviation of the performance metrics. The experiments are executed on a range of GPUs—specifically, the V100, A100, GeForce RTX 2080, or 3090—based on their availability at the time the experiments are conducted. For optimization, we employ the Adam algorithm and undertake a grid search of hyperparameters, the specifics of which are in Tab. 12 and Tab. 13. Should the baseline results be publicly accessible, we directly incorporate them into our report. For the datasets where baseline results are missing (Cornell5, Amherst41, US-election), we reproduced them following the code



(a) Penn94 (b) Genius (c) Cornell5 (d) Amherst41 (e) US-election (f) Wisconsin

Figure 6: Homophily matrix for heterophilous datasets.

in [35].

Hyperparameters for DC-GNN For experiments in Tab. 1 and Tab. 3, we fix some hyperparameters and perform grid search for other hyperparameters. To facilitate reproducibility, we document the details of the hyperparameters and search space in Tab. 12 and Tab. 13 respectively. T^{Ω} refer to the number of iterations of Sinkhorn-Knopp algorithm when solving P^{Ω}, and T^{Γ} is the number of Sinkhorn-Knopp iterations for solving P^{Γ}. We set $|\Omega|$ to be multiples of the number of classes in a dataset.

Table 12: Hyper-parameter search space of DC-GNN for datasets in Tab. 1

.

	Penn94	**Cornell5**	**Amherst41**	**Genius**	**US-election**	**Wisconsin**	**Cora**	**Citeseer**	**Pubmed**
lr	0.005	0.005	0.005	0.005	0.005	0.01	0.005	0.001, 0.002	0.005
λ	2	2, 5	2	2	2,5	2	2	2	2
T^{Ω}	10,5,3	10,5,3	10,5,3	10,5,3	10,5,3	10,5,3	10,5,3	10,5,3	10,5,3
T^{Γ}	5, 3, 1	5, 3, 1	5, 3, 1	5, 3, 1	5, 3, 1	5, 3, 1	5, 3, 1	5, 3, 1	5, 3, 1
$\|\Gamma_i\|$	2	2	2	2	2	2	2	2	2
$\|\Omega\|$	2,4, 8, 16, 30	2,4, 8, 16, 30	2,4, 8, 16, 30	2,4,8	2,4, 8, 16, 30	5, 10, 20	6,12,24,48	7, 14, 28	6, 12
α	0. 0.2, 0.5, 0.8, 1	0. 0.2, 0.5, 0.8, 1	0. 0.2, 0.5, 0.8, 1	0. 0.2, 0.5, 0.8, 1	0.2, 0.5, 0.8	0.2, 0.5, 0.8	0.2, 0.5, 0.8	0.2, 0.5, 0.8	0.2, 0.5, 0.8
β	0, 0.2, 0.5, 0.8	0, 0.2, 0.5, 0.8	0, 0.2, 0.5, 0.8	0, 0.2, 0.5, 0.8	0.2, 0.5, 0.8	0.2, 0.5, 0.8	0.2, 0.5, 0.8	0.2, 0.5, 0.8	0.2, 0.5, 0.8
# layers in MLP	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2
L : # layers	2, 5	2, 5	2, 5	2, 4, 8,16	2, 5	2, 5	2,4,8,10	2,4,8	2,4,8
ω_1	0.001, 0.01	0.001, 0.01	0.001, 0.01	0.001,0.01	0.001, 0.01	0.001, 0.01	0, 0.001, 0.01, 0.05	0.001, 0.01	0.001, 0.01
ω_2	0.005, 0.05	0.005, 0.05	0.005, 0.05	0,0.005,0.05	0.005, 0.05	0.005, 0.05	0.005, 0.05, 0.08, 0.1	0.005, 0.05	0.005, 0.05
epochs	30	30	30	3000	500	200	200	50	500
weight_decay	5e-4	5e-4	5e-4	5e-4	5e-4	1e-3	5e-4	5e-4	5e-4
aggregation	mean, sum	mean, sum	mean, sum	mean, sum	mean, sum	mean, sum	mean, sum	mean, sum	mean, sum
dropout	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
normalization	None	None	None	LN	None	None	None	None	None
hidden_channels	16, 32, 64, 128	16, 32, 64, 128	16, 32, 64, 128	16, 32	16, 32, 64, 128	16, 32, 64, 128	16, 32, 64, 128	16, 32, 64, 128	16, 32, 64, 128

Table 13: Hyper-parameter search space of DC-GNN for datasets in Tab. 3

	Roman-Empire	**Amazon-Ratings**	**Minesweeper**	**Tolokers**	**Questions**
lr	0.005	0.005	0.005	0.005	0.001
λ	2	2	2	2	2
T^{Ω}	5	5	5	5	5
T^{Γ}	3	3	3	3	3
K^{Γ}	2	2	2	2	2
K^{Ω}	18	5, 10	2,4,8	8, 16	2,4,8
α	0.2, 0.5, 0.8	0.2, 0.5, 0.8	0.2, 0.5, 0.8	0.2, 0.5, 0.8	0.2, 0.5, 0.8
β	0.2, 0.5, 0.8	0.2, 0.5, 0.8	0.2, 0.5, 0.8	0.2, 0.5, 0.8	0.2, 0.5, 0.8
# layers in MLP	1,2	1,2	1,2	1,2	1,2
L : # layers	2, 4, 8,16,20	2, 4, 8	2, 4, 8	2, 4, 8	2, 4, 8, 16
ω_1	0.001, 0.01	0.001, 0.01	0.001, 0.01	0.02	0.001, 0.01
ω_2	0.005, 0.05	0.005, 0.05	0.005, 0.05	0.001	0.001
epochs	3000	3000	3000	3000	3000
weight_decay	5e-4	5e-4	5e-4	5e-4	5e-4
aggregation	mean, sum	mean, sum	mean, sum	mean, sum	mean, sum
dropout	0.2	0.2	0.2	0.2	0.2
normalization	None, LN	None, LN	None, LN	None, LN	None, LN
hidden_channels	16, 32, 64	16, 32, 64	16, 32, 64	16, 32, 64	16, 32, 64

Hyperparameters for baselines We used the code provided by GloGNN [35] to reproduce the baseline results for dataset Cornell5, Amherst41, and US-election. The grid search space for hyper-parameters are listed below. Note that some hyper-parameters only apply to a subset of baselines. All other baselines results are obtained from [35] and [62].

- MLP: hidden dimension $\in \{16, 32, 64\}$, number of layers $\in \{2, 3\}$. Activation function is ReLU.
- GCN: lr $\in \{.01, .001\}$, hidden dimension $\in \{4, 8, 16, 32, 64\}$. Activation function is ReLU.
- GAT: lr $\in \{.01, .001\}$. hidden channels $\in \{4, 8, 12, 32\}$ and gat heads $\in \{2, 4, 8\}$. number of layers $\in \{2\}$. We use the ELU as activation.
- MixHop: hidden dimension $\in \{8, 16, 32\}$, number of layers $\in \{2\}$.
- GCNII: number of layers $\in \{2, 8, 16, 32, 64\}$, strength of initial residual connection $\alpha \in \{0.1, 0.2, 0.5\}$, hyperparameter for strength of the identity mapping $\theta \in \{0.5, 1.0, 1.5\}$.
- H_2GCN: hidden dimension $\in \{16, 32\}$, dropout $\in \{0, .5\}$, number of layers $\in \{1, 2\}$. Model architecture follows Section 3.2 of [11].
- WRGAT: lr $\in \{.01\}$, hidden dimension $\in \{32\}$.
- GPR-GNN: lr $\in \{.01, .05, .002\}$, hidden dimension $\in \{16, 32\}$.
- GGCN: lr $\in \{.01\}$, hidden channels $\in \{16, 32, 64\}$, number of layers $\in \{1, 2, 3\}$, weight decay $\in \{1e^{-7}, 1e^{-2}\}$, decay rate $\in \{0, 1.5\}$, dropout rate $\in \{0, .7\}$,
- ACM-GCN: lr $\in \{.01\}$, weight decay $\in \{5e^{-5}, 5e^{-4}, 5e^{-3}\}$, dropout $\in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, hidden channels $\in \{64\}$, number of layers $\in \{2\}$, display step $\in \{1\}$.
- LINKX: hidden dimension $\in \{16, 32, 64\}$, number of layers $\in \{1, 2\}$. Rest of the hyper-parameter settings follow [39].
- GloGNN++: lr $\in \{.001, .005, .01\}$, weight decay $\in \{0, .01, .1\}$, dropout $\in \{0, .5, .8\}$, hidden channels $\in \{128, 256\}$, number of layers $\in \{1, 2\}$, $\alpha \in \{0, 1\}$, $\beta \in \{0.1, 1\}$, $\gamma \in \{0.2, 0.5, 0.9\}$, $\delta \in \{0.2, 0.5\}$, number of normalization layers $\in \{1, 2\}$, orders $\in \{1, 2, 3\}$.