Position-aware Structure Learning for Graph Topology-imbalance by Relieving Under-reaching and Over-squashing

Qingyun Sun
Beihang University
Beijing, China
sunqy@buaa.edu.cn

Jianxin Li
Beihang University
Beijing, China
lijx@buaa.edu.cn

Haonan Yuan
Beihang University
Beijing, China
yuanhn@act.buaa.edu.cn

Xingcheng Fu
Beihang University
Beijing, China
fuxc@act.buaa.edu.cn

Hao Peng
Beihang University
Beijing, China
penghao@act.buaa.edu.cn

Cheng Ji
Beihang University
Beijing, China
jicheng@act.buaa.edu.cn

Qian Li
Beihang University
Beijing, China
liqian@act.buaa.edu.cn

Philip S. Yu
University of Illinois at Chicago
Chicago, USA
psyu@uic.edu

ABSTRACT

Topology-imbalance is a graph-specific imbalance problem caused by the uneven topology positions of labeled nodes, which significantly damages the performance of GNNs. What topology-imbalance means and how to measure its impact on graph learning remain under-explored. In this paper, we provide a new understanding of topology-imbalance from a global view of the supervision information distribution in terms of under-reaching and over-squashing, which motivates two quantitative metrics as measurements. In light of our analysis, we propose a novel position-aware graph structure learning framework named PASTEL, which directly optimizes the information propagation path and solves the topology-imbalance issue in essence. Our key insight is to enhance the connectivity of nodes within the same class for more supervision information, thereby relieving the under-reaching and over-squashing phenomena. Specifically, we design an anchor-based position encoding mechanism, which better incorporates relative topology position and enhances the intra-class inductive bias by maximizing the label influence. We further propose a class-wise conflict measure as the edge weights, which benefits the separation of different node classes. Extensive experiments demonstrate the superior potential and adaptability of PASTEL in enhancing GNNs' power in different data annotation scenarios.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Learning latent representations**.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557419

KEYWORDS

graph representation learning, graph neural networks, imbalance learning, graph structure learning, node classification

ACM Reference Format:
Qingyun Sun, Jianxin Li, Haonan Yuan, Xingcheng Fu, Hao Peng, Cheng Ji, Qian Li, and Philip S. Yu. 2022. Position-aware Structure Learning for Graph Topology-imbalance by Relieving Under-reaching and Over-squashing. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557419

1 INTRODUCTION

Graph learning [13, 24, 52] has gained popularity over the past years due to its versatility and success in representing graph data across a wide range of domains [9, 14, 26, 40, 50]. Graph Neural Networks (GNNs) [39, 47] have been the "battle horse" of graph learning, which propagate the features on the graph by exchanging information between neighbors in a message-passing paradigm [15]. Due to the asymmetric and uneven topology, learning on graphs by GNNs suffers a specific imbalance problem, i.e., topology-imbalance. Topology-imbalance [7] is caused by the uneven position distribution of labeled nodes in the topology space, which is inevitable in real-world applications due to data availability and the labeling costs. For example, we may only have information for a small group of users within a local community in social networks, resulting in a serious imbalance of labeled node positions. The uneven position distribution of labeled nodes leads to uneven information propagation, resulting in the poor quality of learned representations.

Although the imbalance learning on graphs has attracted many research interests in recent years, most of them focus on the class-imbalance issue [30, 46], i.e., the imbalanced number of labeled nodes of each class. The topology-imbalance issue is proposed recently and is still under-explored. The only existing work, ReNode [7], provides an understanding of the topology-imbalance issue from the perspective of label propagation and proposes a sample re-weighting method. However, ReNode takes the node topological

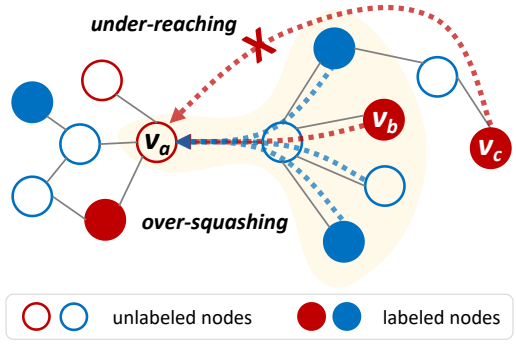


Figure 1: Schematic diagram of under-reaching and over-squashing in the topology-imbalance issue.

boundaries as decision boundaries based on a homophily assumption, which does not work with real-world graphs. The strong assumption leads to poor generalization and unsatisfied performance of ReNode (see Section 5.2.1). There are **two remaining questions**: *(1) Why does topology-imbalance affect the performance of graph representation learning?* and *(2) What kind of graphs are susceptible to topology-imbalance?* To answer the above two questions, how to measure the influence of labeled nodes is the key challenge in handling topology-imbalance due to the complex graph connections and the unknown class labels for most nodes in the graph.

New understanding for topology-imbalance. In this work, we provide a new understanding of the topology-imbalance issue from a global view of the supervision information distribution in terms of under-reaching and over-squashing: **(1) Under-reaching**: the influence of labeled nodes decays with the topology distance [3], resulting in the nodes far away from labeled nodes lack of supervision information. In Figure 1, the node v_a cannot reach the valuable labeled node v_c within the receptive field of the GNN model, resulting in the quantity of information it received is limited. **(2) Over-squashing**: the supervision information of valuable labeled nodes is squashed when passing across the narrow path together with other useless information. In Figure 1, the valuable supervision information of v_b to v_a is compressed into a vector together with the information of many nodes belonging to other classes, resulting in the quality of supervision information that v_a received being poor. Then we introduce two metrics (reaching coefficient and squashing coefficient) to give a quantitative analysis of the relation between the learning performance, label positions, and graph structure properties. We further draw a conclusion that *better reachability and lower squashing to labeled nodes lead to better classification performance for GNN models.*

Present work. In light of the above analysis, we propose a **P**osition-**A**ware **STructur**E **L**earning method named **PASTEL**, which directly optimizes the information propagation path and solves the problem of topology-imbalance issue in essence. The key insight of PASTEL is to enable nodes within the same class to connect more closely with each other for more supervision information. Specifically, we design a novel *anchor-based position encoding mechanism* to capture the relative position between nodes and incorporate the position information into structure learning. Then we design a *class-wise conflict measure* based on the Group PageRank, which measures the influence from labeled nodes of each class and acts as a guide to increase the intra-class connectivity via adjusting edge weight. The main contributions are as follows:

- We provide a new understanding of the topology-imbalance issue from the perspective of supervision information distribution in terms of under-reaching and over-squashing and provide two new quantitative metrics for them.
- Equipped with the proposed position encodings and class-wise conflict measure, PASTEL can better model the relationships of node pairs and enhance the intra-class inductive bias by maximizing the label influence.
- Experimental results demonstrate that the proposed PASTEL enjoys superior effectiveness and indeed enhances the GNN model's power for in-the-wild extrapolation.

2 RELATED WORK

2.1 Imbalance Learning

Imbalanced classification problems [18, 41] have attracted extensive research attention. Most existing works [16, 25] focus on the class-imbalance problem, where the model performance is dominated by the majority class. The class-imbalance learning methods can be roughly divided into two types: data-level re-sampling and algorithm-level re-weighting. **Re-sampling** methods re-sample [2, 5, 48] or augment data [30] to balance the number of data for each class during the data selection phase. **Re-weighting** methods [4, 11, 32] adjust different weights to different data samples according to the number of data during the training phase.

For the graph-specific topology-imbalance issue as mentioned in Section 1, directly applying these methods to the graph data fails to take the special topology properties into consideration. ReNode [7] is the first work for the graph topology-imbalance issue, which follows the paradigm of classical re-weighting methods. Specifically, ReNode defines an influence conflict detection based metric and re-weights the labeled nodes based on their relative positions to class boundaries. However, ReNode is limited by its homophily assumption and only has a slight performance improvement. *In this paper, PASTEL alleviates topology-imbalance by learning a new structure that maximizes the intra-class label influence, which can be seen as "label re-distribution" in the topology space.*

2.2 Graph Structure Learning

Graph structure learning [55] learns an optimized graph structure for representation learning and most of them aim to improve the robustness [20, 54] of GNN models. There are also some works [8, 10, 12, 38, 42] that utilize the structure learning to improve the graph representation quality. As for the over-squashing problem, [45] assigns different weights to edges connected to two nodes of the same class for better representations. However, [45] still fails with the issue of under-reaching. SDRF [42] rewires edges according to the Ricci curvatures to solve the over-squashing problem by only considering topology properties.

Multiple measurements in existing structure learning works are leveraged for modeling node relations, including node features [53], node degrees [20], node encodings [51] and edge attributes [54].

The node positions play an important role in generating discriminative representations [49] and are seldom considered in structure learning. *In this work, we advance the structure learning strategy for the graph topology-imbalance issue and introduce a position-aware framework to better capture the nodes' underlying relations.*

3 UNDERSTANDING TOPOLOGY-IMBALANCE

In this section, we provide a new understanding of the topology-imbalance issue in terms of under-reaching and over-squashing. Then we perform a quantitative analysis of the relations between them to answer two questions:

Q1: Why does topology-imbalance affect the performance of graph representation learning?

Q2: What kind of graphs are susceptible to topology-imbalance?

3.1 Notations and Preliminaries

Consider a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the set of N nodes and $\mathcal{E}$ is the edge set. Let $\mathbf{A} \in \mathbb{R}^{N\times N}$ be the adjacency matrix and $\mathbf{X} \in \mathbb{R}^{N\times d_0}$ be the node attribute matrix, where d_0 denotes the dimension of node attributes. The diagonal degree matrix is denoted as $\mathbf{D} \in \mathbb{R}^{N\times N}$ where $\mathbf{D}_{ii} = \sum_{j=1}^{N} \mathbf{A}_{ij}$. The graph diameter is denoted as $D_{\mathcal{G}}$. Given the labeled node set $\mathcal{V}_L$ and their labels $\mathcal{Y}_L$ where each node v_i is associated with a label y_i, *semi-supervised node classification* aims to train a node classifier $f_\theta : v \to \mathbb{R}^C$ to predict the labels $\mathcal{Y}_U$ of remaining nodes $\mathcal{V}_U = \mathcal{V} \setminus \mathcal{V}_L$, where C denotes the number of classes. we separate the labeled node set $\mathcal{V}_L$ into $\{\mathcal{V}_L^1, \mathcal{V}_L^2, \cdots, \mathcal{V}_L^C\}$, where $\mathcal{V}_L^i$ is the nodes of class i in $\mathcal{V}_L$.

3.2 Understanding Topology-Imbalance via Under-reaching and Over-squashing

In GNNs, node representations are learned by aggregating information from valuable neighbors. The quantity and quality of the information received by the nodes decide the expressiveness of their representations. We perceive the imbalance of the labeled node positions affects the performance of GNNs for two reasons:

(1) **Under-reaching**: The influence from labeled nodes decays with the topology distance [3], resulting in that the nodes far away from labeled nodes lack supervision information. When the node can't reach enough valuable labeled nodes within the receptive field of the model, the quantity of information it received is limited.

(2) **Over-squashing**: The receptive field of GNNs is exponentially-growing and all information is compressed into fixed-length vectors [1]. The supervision information of valuable labeled nodes is squashed when passing across the narrow path together with other useless information.

3.3 Quantitative Analysis

To provide quantitative analysis for topology-imbalance, we propose two metrics for reachability and squashing. First, we define a reaching coefficient based on the shortest path, which determines the minimum layers of GNNs to obtain supervision information:

Definition 1 (**Reaching coefficient**). *Given a graph $\mathcal{G}$ and labeled node set $\mathcal{V}_L$, the reaching coefficient RC of $\mathcal{G}$ is the mean length of the shortest path from unlabeled nodes to the labeled nodes of their corresponding classes:*

$$RC = \frac{1}{|\mathcal{V}_U|} \sum_{v_i \in \mathcal{V}_U} \frac{1}{|\mathcal{V}_L^{y_i}|} \sum_{v_j \in \mathcal{V}_L^{y_i}} \left(1 - \frac{\log |\mathcal{P}_{sp}(v_i, v_j)|}{\log D_{\mathcal{G}}}\right), \quad (1)$$

where $\mathcal{V}_L^{y_i}$ denotes the nodes in $\mathcal{V}_L$ whose label is y_i, $\mathcal{P}_{sp}(v_i, v_j)$ denotes the shortest path between v_i and v_j, and $|\mathcal{P}_{sp}(v_i, v_j)|$ denotes its length, and $D_{\mathcal{G}}$ is the diameter of graph $\mathcal{G}$. Specifically, for the unconnected v_i and v_j, we set the length of their shortest path as $D_{\mathcal{G}}$.

The reaching coefficient reflects how long the the distance when the GNNs passes the valuable information to the unlabeled nodes. Note that $RC \in [0, 1)$ and larger RC means better reachability.

For the quantitative metric of over-squashing, we define a squashing coefficient using the Ricci curvature to formulate it from a geometric perspective. The Ricci curvature [28] reflects the change of topology properties of the two endpoints of an edge, where the negative $Ric(v_i, v_j)$ means that the edge behaves locally as a shortcut or bridge and positive $Ric(v_i, v_j)$ indicates that locally there are more triangles in the neighborhood of v_i and v_j [27, 42].

Definition 2 (**Squashing coefficient**). *Given a graph $\mathcal{G}$, the squashing coefficient SC of $\mathcal{G}$ is the mean Ricci curvature of edges on the shortest path from unlabeled nodes to the labeled nodes of their corresponding classes:*

$$SC = \frac{1}{|\mathcal{V}_U|} \sum_{v_i \in \mathcal{V}_U} \frac{1}{|\mathcal{N}_{y_i}(v_i)|} \sum_{v_j \in \mathcal{N}_{y_i}(v_i)} \frac{\sum_{e_{kt} \in \mathcal{P}_{sp}(v_i, v_j)} Ric(v_k, v_t)}{|\mathcal{P}_{sp}(v_i, v_j)|}, \quad (2)$$

where $\mathcal{N}_{y_i}(v_i)$ denotes the labeled nodes of class y_i that can reach v_i, $Ric(\cdot, \cdot)$ denotes the Ricci curvature, and $|\mathcal{P}_{sp}(v_i, v_j)|$ denotes the length of shortest path between v_i and v_j.

We leverage the *Ollivier-Ricci curvature* [28] as $Ric(\cdot, \cdot)$ here:

$$Ric(v_k, v_t) = \frac{Wasserstein(mass_k, mass_t)}{d_{geo}(v_k, v_t)}, \quad (3)$$

where $Wasserstein(\cdot, \cdot)$ is the Wasserstein distance, $d_{geo}(\cdot, \cdot)$ is the geodesic distance function, and $mass_k$ is the mass distribution [28] of node v_k. Note that SC can be either positive or negative and larger SC means lower squashing because the ring structures are more friendly for information sharing.

In Figure 2 and Figure 3, we show the relation between the reaching coefficient RC, the squashing coefficient SC, and the classification accuracy. The higher the accuracy, the darker and larger the corresponding scatter. First, we analyze the performance of GCN when trained on the same graph structure but with different labeled nodes. In Figure 2, we generate a synthetic graph by the Stochastic Block Model (SBM) [19] with 4 classes and 3,000 nodes. We randomly sample some nodes as the labeled nodes 10 times and scatter the classification accuracy in Figure 2. We can observe that even for the same graph structure, the difference in positions of labeled nodes may bring up to 15.42% difference in accuracy. There is a significant positive correlation between the reaching coefficient, the squashing coefficient, and the model performance.

Then we analyze the performance of GCN when trained with the same labeled nodes but on different graph structures. In Figure 3, we set the labeled nodes to be the same and generate different structures between them by controlling the edge probability between

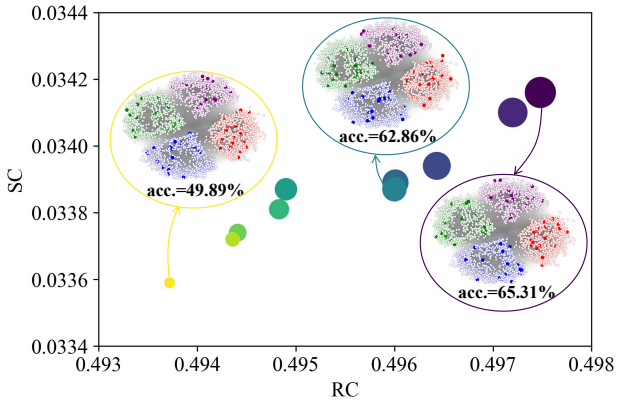


Figure 2: Predictions of GCN with the same graph structure and different labeled nodes.

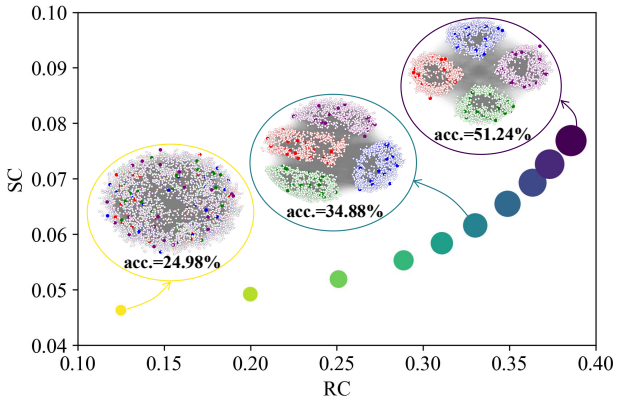


Figure 3: Predictions of GCN with and the same labeled nodes and different graph structures.

communities in the SBM model. We can observe that with the same supervision information, there is up to a 26.26% difference in accuracy because of the difference in graph structures. There is also a significant positive correlation between the reaching coefficient, the squashing coefficient, and the model performance. When the graph shows better community structure among nodes of the same class, the node representations can be learned better.

Therefore, we make the following conclusions: (1) Topology-imbalance hurts the performance of graph learning in the way of under-reaching and over-squashing. (for Q1) (2) The proposed two quantitative metrics can effectively reflect the degree of topology-imbalance. Graph with poor reachability (i.e., smaller *RC*) and stronger squashing (i.e., smaller *SC*) is more susceptible to topology-imbalance. (for Q2) (3) Optimizing the graph structure can effectively solve the topology-imbalance issue. The above conclusions provide the guideline for designing the framework of PASTEL, i.e., balance the supervision information distribution by learning a structure with better reachability and lower squashing.

4 ALLEVIATE TOPOLOGY-IMBALANCE BY STRUCTURE LEARNING

In this section, we introduce **PASTEL**, a **P**osition-**A**ware **ST**ructur**E** **L**earning framework, to optimize the information propagation path directly and address the topology-imbalance issue in essence. In light of the analysis in Section 3.2, PASTEL aims to learn a better structure that increases the intra-class label influence for each class and thus relieves the under-reaching and over-squashing phenomena. The overall architecture of PASTEL is shown in Figure 4.

4.1 Position-aware Structure Learning

To form structure with better intra-class connectivity, we use an anchor-based position encoding method to capture the topology distance between unlabeled nodes to labeled nodes. Then we incorporate both the merits of feature information as well as topology information to learn the refined structure.

Anchor-based Position Encoding. Inspired by the position in transformer [36, 43], we use an anchor-based position encoding method to capture the relative position of unlabeled nodes with respect to all the labeled nodes of the graph. Since we focus on maximizing the reachability between unlabeled nodes and labeled nodes within the same class, we directly separate the labeled node set $\mathcal{V}_L$ into C anchor sets $\{\mathcal{V}_L^1, \mathcal{V}_L^2, \cdots, \mathcal{V}_L^C\}$, where each subset $\mathcal{V}_L^c$ denotes the labeled nodes whose labels are c. The class-wise anchor sets help distinguish the information from different classes rather than treating all the anchor nodes the same and ignoring the class difference as in [49]. Concretely, for any node v_i, we consider a function $\phi(\cdot,\cdot)$ which measures the position relations between v_i and the anchor sets in graph $\mathcal{G}$. The function can be defined by the connectivity between the nodes in the graph.

$$\mathbf{p}_i = \left(\phi\left(v_i, \mathcal{V}_L^1\right), \phi\left(v_i, \mathcal{V}_L^2\right), \cdots, \phi\left(v_i, \mathcal{V}_L^C\right)\right), \tag{4}$$

where $\phi(v_i, \mathcal{V}_L^c)$ is the position encoding function defined by the connectivity between the node v_i and the anchor set $\mathcal{V}_L^c$ in graph. Here we choose $\phi(v_i, \mathcal{V}_L^c)$ to be the mean length of shortest path between v_i and nodes in $\mathcal{V}_L^c$ if two nodes are connected:

$$\phi(v_i, \mathcal{V}_L^c) = \frac{\sum_{v_j \in \mathcal{N}_c(v_i)} |\mathcal{P}_{sp}(v_i, v_j)|}{|\mathcal{N}_c(v_i)|}, \tag{5}$$

where $\mathcal{N}_c(v_i)$ is the nodes connected with v_i in $\mathcal{V}_L^c$ and $|\mathcal{P}_{sp}(v_i, v_j)|$ is the length of shortest path between v_i and v_j. Then we transform the position encoding into the d_0 dimensional space:

$$\mathbf{h}_i^p = \mathbf{W}_\phi \cdot \mathbf{p}_i, \tag{6}$$

where $\mathbf{W}_\phi$ is a trainable vector. If two nodes have similar shortest paths to the anchor sets, their position encodings are similar.

Position-aware Metric Learning. After obtaining the position encoding, we use a metric function that accounts for both node feature information and the position-based similarities to measure the possibility of edge existence. PASTEL is agnostic to various similarity metric functions and we choose the widely used multi-head cosine similarity function here:

$$a_{ij}^p = \frac{1}{m}\sum_{h=1}^{m} \cos\left(\mathbf{W}_h \cdot \left(\mathbf{z}_i || \mathbf{h}_i^p\right), \mathbf{W}_h \cdot \left(\mathbf{z}_j || \mathbf{h}_j^p\right)\right), \tag{7}$$

where m is the number of heads, $\mathbf{W}_h$ is the weight matrix of the h-th head, $\mathbf{z}_i$ denotes the representation vector of node v_i and $||$ denotes

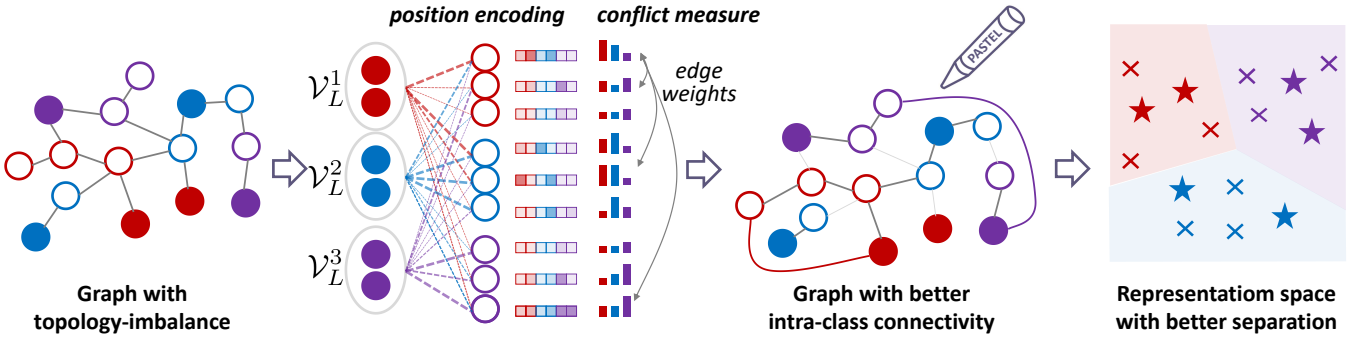


Figure 4: Overall architecture of PASTEL. PASTEL encodes the relative position between nodes with the labeled nodes as anchor sets $\{\mathcal{S}\}$ and incorporates the position information with node features for structure learning. For each pair of nodes, PASTEL uses the class-wise conflict measure as the edge weights to learn a graph with better intra-class connectivity.

concatenation. The effectiveness of the position-aware structure learning is evaluated in Section 5.3.1.

4.2 Class-wise Conflict Measure

We aim to increase the intra-class connectivity among nodes, thereby increasing the supervision information they received and their influence on each other. Here we propose a class-wise conflict measure to guide what nodes should be more closely connected. According to the inherent relation of GNNs with Label Propagation [7, 45], we use the *Group PageRank* [6] as a conflict measure between nodes. Group PageRank (GPR) extends the traditional PageRank[29] into a label-aware version to measure the supervision information from labeled nodes of each class. Specifically, for class $c \in \{1, 2, \cdots, C\}$, the corresponding GPR matrix is

$$\mathbf{P}^{gpr}(c) = (1-\alpha)\mathbf{A}'\mathbf{P}^{gpr}(c) + \alpha\mathbf{I}_c, \tag{8}$$

where $\mathbf{A}' = \mathbf{A}\mathbf{D}^{-1}$, α is the random walk restart probability at a random node in the group and $\mathbf{I}_c \in \mathbb{R}^n$ is the teleport vector:

$$\mathbf{I}_c^i = \begin{cases} \frac{1}{|\mathcal{V}_L^c|}, & if\ y_i = c \\ 0, & otherwise \end{cases} \tag{9}$$

where $|\mathcal{V}_L^c|$ is the number of labeled nodes with class c. We calculate the GPR for each group individually and then concatenate all the GPR vectors to form a final GPR matrix $\mathbf{P}^{gpr} \in \mathbb{R}^{N\times C}$ as in [6]:

$$\mathbf{P}^{gpr} = \alpha\left(\mathbf{E} - (1-\alpha)\,\mathbf{A}'\right)^{-1}\mathbf{I}^*, \tag{10}$$

where $\mathbf{E}$ is the unit matrix of nodes and $\mathbf{I}^*$ is the concatenation of $\{\mathbf{I}_c, c = 1, 2, \cdots, C\}$. Under $\mathbf{P}^{gpr}$, node v_i corresponds to a GPR vector $\mathbf{P}_i^{gpr}$ (the i-th row of $\mathbf{P}^{gpr}$), where the c-th dimension represents the the supervision influence of labeled nodes of class c on node v_i. The GPR value contains not only the global topology information but also the annotation information.

For each node pair nodes v_i and v_j, we use the Kullback Leiber (KL) divergence of their GPR vectors to measure their conflict when forming an edge:

$$\kappa_{ij} = \mathrm{KL}\left(\mathbf{P}_i^{gpr}, \mathbf{P}_j^{gpr}\right). \tag{11}$$

The distance of GPR vectors reflects the influence conflict of different classes when exchanging information. We use a cosine annealing mechanism to calculate the edge weights by the relative ranking of the conflict measure:

$$w_{ij} = \frac{1}{2}\left[-\cos\frac{\mathrm{Rank}(\kappa_{ij})}{|\mathcal{V}|\times|\mathcal{V}|} * \pi + 1\right], \tag{12}$$

where $Rank(\cdot)$ is the ranking function according to the magnitude. The more conflicting the edge is, the less weight is assigned to it. With the class-wise conflict measure, we aim to learn a graph structure that makes the GPR vectors of nodes have "sharp" distributions focusing on their ground-truth classes. Then w_{ij} is used as the connection strength of edge e_{ij}, with the corresponding element $\tilde{a}_{ij}^P$ in the adjacency matrix being:

$$\tilde{a}_{ij}^P = w_{ij} \cdot a_{ij}^P. \tag{13}$$

The effectiveness of the class-wise conflict measure is evaluated in Section 5.3.2 and the change of GPR vectors is shown in Section 5.4.3.

4.3 Learning with the Optimized Structure

With the above structure learning strategy, we can obtain a position-aware adjacency $\mathbf{A}_P$ with maximum intra-class connectivities:

$$\mathbf{A}_P = \{\tilde{a}_{ij}^P, i, j \in \{1, 2, \cdots, N\}\}. \tag{14}$$

The input graph structure determines the learning performance to a certain extent. Since the structure learned at the beginning is of poor quality, directly using it may lead to non-convergence or unstable training of the whole framework. We hence incorporate the original graph structure $\mathbf{A}$ and a structure in a node feature view $\mathbf{A}_N$ as supplementary to formulate an optimized graph structure $\mathbf{A}^*$. Specifically, we also learn a graph structure $\mathbf{A}_N = \{a_{ij}^N, i, j \in \{1, 2, \cdots, N\}\}$ in a node feature view with each element being:

$$a_{ij}^N = \frac{1}{m}\sum_{h=1}^{m}\cos\left(\mathbf{W}_h \cdot \left(\mathbf{x}_i || \mathbf{h}_i^{p_0}\right), \mathbf{W}_h \cdot \left(\mathbf{x}_j || \mathbf{h}_j^{p_0}\right)\right), \tag{15}$$

where $\mathbf{x}_i$ is the feature vector of node v_i and $\mathbf{h}_i^{p_0}$ is the position encoding with the original structure. Then we can formulate an optimized graph structure $\mathbf{A}^*$ with respect to the downstream task:

$$\mathbf{A}^* = \lambda_1\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} + (1-\lambda_1)\left(\lambda_2 f(\mathbf{A}_N) + (1-\lambda_2)\, f(\mathbf{A}_P)\right), \tag{16}$$

where $f(\cdot)$ denotes the row-wise normalization function, λ_1 and λ_2 are two constants that control the contributions of original structure

Algorithm 1: The overall process of PASTEL

Input: Graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with node labels $\mathcal{Y}$; Number of heads m; Number of training epochs E; Structure fusing coefficients λ_1, λ_2; Loss coefficients $\beta_1, \beta_2, \beta_3$

Output: Optimized graph $\mathcal{G}^* = (\mathbf{A}^*, \mathbf{X})$, predicted label $\hat{\mathcal{Y}}$

Parameter initialization;
for $e = 1, 2, \cdots, E$ **do**
  // Learn position-aware graph structure
  Learn position encodings $\mathbf{h}_i^p \leftarrow$ Eq. (6);
  Learn edge possibility $a_{ij}^P \leftarrow$ Eq. (7);
  Calculate the Group PageRank matrix $\mathbf{P}^{gpr} \leftarrow$ Eq. (10);
  Calculate the class-wise conflict measure $w_{ij} \leftarrow$ Eq. (12);
  Obtain position-aware structure $\mathbf{A}_P \leftarrow$ Eq. (14);
  // Learn node representations
  Obtain the optimized structure $\mathbf{A}^* \leftarrow$ Eq. (16);
  Calculate representations and labels $\mathbf{Z}$, $\hat{\mathcal{Y}} \leftarrow$ Eq. (20);
  // Optimize
  Calculate the losses $\mathcal{L}_{cls} \leftarrow$ Eq. (21), $\mathcal{L}_{smooth} \leftarrow$Eq. (17), $\mathcal{L}_{con} \leftarrow$Eq. (18), and $\mathcal{L}_{spar} \leftarrow$Eq. (19);
  Update model parameters to minimize $\mathcal{L} \leftarrow$ Eq. (22).
end

and feature view structure, respectively. Here we use a dynamic decay mechanism for λ_1 and λ_2 to enable the position-aware structure $\mathbf{A}_P$ to play a more and more important role during training.

To control the quality of learned graph structure, we impose additional constraints on it following [10, 21] in terms of smoothness, connectivity, and sparsity:

$$\mathcal{L}_{smooth} = \frac{1}{N^2}\text{tr}\left(\mathbf{X}^T\mathbf{L}^*\mathbf{X}\right), \tag{17}$$

$$\mathcal{L}_{con} = \frac{1}{N}\mathbf{1}^T\log(\mathbf{A}^*\mathbf{1}), \tag{18}$$

$$\mathcal{L}_{spar} = \frac{1}{N^2}||\mathbf{A}^*||_F^2, \tag{19}$$

where $\mathbf{L}^* = \mathbf{D}^* - \mathbf{A}^*$ is the Laplacian of $\mathbf{A}^*$ and $\mathbf{D}^*$ is the degree matrix of $\mathbf{A}^*$. To speed up the computation, we extract a symmetric sparse non-negative adjacency matrix by masking off (i.e., set to zero) those elements in $\mathbf{A}^*$ which are smaller than a predefined non-negative threshold a_0. Then $\mathcal{G}^* = (\mathbf{A}^*, \mathbf{X})$ is input into the GNN-Encoder for the node representations $\mathbf{Z} \in \mathbb{R}^{N\times d}$, predicted labels $\hat{y}$ and classification loss $\mathcal{L}_{cls}$:

$$\mathbf{Z} = \text{GNN-Encoder}(\mathbf{A}^*, \mathbf{X}), \hat{\mathcal{Y}} = \text{Classifier}(\mathbf{Z}), \tag{20}$$

$$\mathcal{L}_{cls} = \text{Cross-Entropy}(\mathcal{Y}, \hat{\mathcal{Y}}). \tag{21}$$

The overall loss is defined as the combination of the node classification loss and graph regularization loss:

$$\mathcal{L} = \mathcal{L}_{cls} + \beta_1\mathcal{L}_{smooth} + \beta_2\mathcal{L}_{con} + \beta_3\mathcal{L}_{spar}. \tag{22}$$

The overall process of PASTEL is shown in Algorithm 1.

5 EXPERIMENT

In this section, we first evaluate PASTEL[1] on both real-world graphs and synthetic graphs. Then we analyze the main mechanisms of PASTEL and the learned structure. We mainly focus on the following research questions:

- **RQ1.** How does PASTEL perform in the node classification task? (Section 5.2)
- **RQ2.** How does the position encoding and the class-wise conflict measure influence the performance of PASTEL? (Section 5.3)
- **RQ3.** What graph structure PASTEL tend to learn? (Section 5.4)

5.1 Experimental Setups

5.1.1 Datasets. We conduct experiments on synthetic and real-world datasets to analyze the model's capabilities in terms of both graph theory and real-world scenarios. The real-word datasets include various networks with different heterophily degrees to demonstrate the generalization of PASTEL. Cora and Citeseer [35] are citation networks. Photo [37] and and Actor [31] are co-occurrence network. Chameleon and Squirrel [34] are page-page networks in Wikipedia. Since we focus on the topology-imbalance issue in this work, we set the number of labeled nodes in each class to be 20.

5.1.2 Baselines. We choose representative GNNs as backbones including GCN [22], GAT [44], APPNP [23], and GraphSAGE [17]. The most important baseline is ReNode [7], which is the only existing work for the topology-imbalance issue. We also include some graph structure learning baselines to illustrate the specific effectiveness of PASTEL for the topology-imbalance issue. DropEdge [33] randomly removes edges at each epoch as structure augmentation. To evaluate the effect of increasing the reachability randomly, we use a adding edges method named AddEdge, whose adding strategy is similar to DropEdge. SDRF [42] rewires edges according to their curvatures for the over-squashing issue. NeuralSparse [54] removes potentially task-irrelevant edges for clearer class boundaries. IDGL [10] updates the node representations and structure based on these representations iteratively.

5.1.3 Parameter Settings. For the GNN backbones, we set their depth to be 2 layers and adopt the implementations from the PyTorch Geometric Library in all experiments. We set the representation dimension of all baselines and PASTEL to be 256. We re-implement the NeuralSparse [54] and SDRF [42] and the parameters of baseline methods are set as the suggested value in their papers or carefully tuned for fairness. For DropEdge and AddEdge, we set the edge dropping/adding probability to 10%. For PASTEL, we set the number of heads $m = 4$ and the random walk restart probability $\alpha = 0.15$. The structure fusing coefficients (λ_1 and λ_2) and the loss coefficients (β_1, β_2 and β_3) are tuned for each dataset.

5.2 Evaluation (RQ1)

5.2.1 PASTEL for Real-world Graphs. We compare PASTEL with the baselines on several datasets on node classification. The overall Weighted-F1 (W-F1) scores and the class-balance Macro-F1 (M-F1) scores on different backbones are shown in Table 1. The best

[1]The code of PASTEL is available at https://github.com/RingBDStack/PASTEL.

Table 1: Weighted-F1 score and Macro-F1 score (% ± standard deviation) of node classification on real-world graph datasets.

		Cora		Citeseer		Photo		Actor		Chameleon		Squirrel	
Backbone	Model	W-F1	M-F1	W-F1	M-F1	W-F1	M-F1	W-F1	M-F1	W-F1	M-F1	W-F1	M-F1
GCN	original	79.4±0.9	77.5±1.5	66.3±1.3	62.2±1.2	85.4±2.8	84.6±1.3	21.8±1.3	20.9±1.4	30.5±3.4	30.5±3.3	21.9±1.2	21.9±1.2
	ReNode	80.0±0.7	78.4±1.3	66.4±1.0	62.4±1.1	86.2±2.4	85.3±1.6	21.2±1.2	20.2±1.6	30.3±3.2	30.4±2.8	22.4±1.1	22.4±1.1
	AddEdge	79.0±0.9	77.0±1.4	66.2±1.3	62.2±1.3	85.5±1.5	86.1±1.8	21.2±1.3	20.3±1.5	30.6±1.6	30.4±1.7	21.7±1.5	21.7±1.5
	DropEdge	79.8±0.8	77.8±1.0	66.6±1.4	63.4±1.6	86.8±1.7	85.4±1.3	22.4±1.0	21.4±1.3	30.6±3.5	30.6±3.3	22.8±1.2	22.8±1.2
	SDRF	82.1±0.8	80.6±0.8	69.6±0.4	66.6±0.3	> 5 days	> 5 days	> 5 days	> 5 days	39.1±1.2	39.0±1.2	> 5 days	> 5 days
	NeuralSparse	81.7±1.4	80.9±1.4	71.8±1.2	69.0±1.0	89.7±1.9	88.7±1.8	24.4±1.5	23.6±1.6	44.9±3.0	44.9±2.8	28.1±1.8	28.1±1.8
	IDGL	82.3±0.6	81.0±0.9	71.7±1.0	68.0±1.3	88.6±2.3	88.8±1.4	24.9±0.8	22.0±0.7	55.4±1.8	55.0±1.7	28.8±2.3	28.9±2.2
	PASTEL	**82.5±0.3**	**81.2±0.3**	**72.9±0.8**	**69.3±0.9**	**91.4±2.7**	**91.3±2.2**	**26.4±1.0**	**24.4±1.2**	**57.8±2.4**	**57.3±2.4**	**37.5±0.6**	**37.5±0.7**
GAT	original	78.3±1.5	76.4±1.7	64.4±1.7	60.6±1.7	88.2±2.9	86.2±2.6	21.8±1.2	20.9±1.1	29.9±3.5	29.9±3.1	20.5±1.4	20.5±1.4
	ReNode	78.9±1.2	77.2±1.5	64.9±1.6	61.0±1.5	89.1±2.4	87.1±2.6	21.5±1.2	20.5±1.1	29.2±2.3	29.1±2.0	20.4±1.8	20.4±1.8
	AddEdge	78.0±1.6	76.2±1.6	64.0±1.3	60.2±1.3	88.2±2.4	86.2±2.5	21.3±1.2	20.3±1.1	29.8±1.7	29.6±1.5	20.7±1.6	20.7±1.6
	DropEdge	78.7±1.3	76.9±1.5	64.5±1.4	60.5±1.3	88.9±1.9	87.1±2.1	22.9±1.2	21.8±1.1	30.3±1.6	30.2±1.2	21.2±1.5	21.2±1.5
	SDRF	77.9±0.7	75.9±0.9	64.9±0.6	61.9±0.9	> 5 days	> 5 days	> 5 days	> 5 days	43.0±1.9	42.5±1.9	> 5 days	> 5 days
	NerualSparse	81.4±4.8	79.4±4.8	64.8±1.5	61.9±1.3	90.2±2.5	88.0±2.3	23.4±1.7	**22.4±1.5**	45.6±2.1	45.5±1.8	28.8±1.3	28.8±1.3
	IDGL	80.6±1.0	79.7±0.9	66.5±1.5	61.9±1.9	89.9±3.1	87.7±2.6	22.4±1.5	21.8±1.2	48.4±4.0	47.8±3.1	27.0±2.6	27.0±2.6
	PASTEL	**81.9±1.4**	**80.7±1.2**	**66.6±1.9**	**62.0±1.7**	**91.8±3.2**	**89.4±2.9**	**24.4±2.6**	22.1±2.6	**52.1±2.7**	**52.5±2.8**	**35.3±0.9**	**35.3±0.8**
APPNP	original	80.6±1.6	79.3±1.2	66.5±1.5	62.3±1.5	89.3±1.6	86.3±1.7	21.1±1.5	20.7±1.1	35.3±4.0	35.0±3.8	23.1±1.6	23.1±1.6
	ReNode	81.1±0.9	79.9±0.9	66.6±1.7	62.4±1.6	89.6±1.4	87.2±1.3	20.2±2.0	20.0±1.7	33.5±2.5	33.3±2.3	23.9±2.0	23.9±2.0
	AddEdge	80.3±1.3	78.8±1.1	66.6±2.1	62.5±2.1	89.3±1.2	86.4±1.2	21.5±1.3	20.7±1.4	35.7±1.7	35.4±1.2	23.1±1.6	23.2±1.7
	DropEdge	80.9±1.4	79.4±1.2	66.7±2.0	63.0±1.9	90.0±1.2	87.0±1.2	21.8±1.8	20.8±1.4	36.0±1.7	35.7±1.6	23.3±1.7	23.3±1.7
	SDRF	80.7±0.9	79.1±0.8	67.1±0.6	63.1±0.8	> 5 days	> 5 days	> 5 days	> 5 days	36.5±2.1	35.8±2.1	> 5 days	> 5 days
	NerualSparse	81.1±1.4	79.9±1.2	66.8±1.9	62.7±1.9	91.3±1.8	89.4±1.6	21.8±1.9	**21.4±1.5**	39.1±2.9	38.7±2.8	28.3±1.5	28.3±1.5
	IDGL	81.3±0.9	**80.2±0.9**	67.0±1.3	62.9±1.3	91.6±1.3	88.6±2.2	21.4±2.4	20.1±2.4	41.2±2.2	40.6±2.6	29.6±2.3	29.7±2.2
	PASTEL	**82.0±1.0**	80.0±0.9	**67.3±1.3**	**63.2±1.5**	**92.3±3.1**	**89.9±2.5**	**22.5±2.0**	20.9±2.1	**44.2±3.2**	**43.8±3.4**	**34.6±1.6**	**34.6±1.6**
GraphSAGE	original	75.4±1.6	74.1±1.6	64.8±1.6	60.7±1.6	86.1±2.5	83.3±2.4	24.0±1.2	23.2±1.0	36.5±1.6	36.2±1.6	27.2±1.7	27.2±1.7
	ReNode	76.4±0.9	75.0±1.1	65.4±1.7	61.2±1.7	86.5±1.7	84.1±1.7	23.7±1.2	22.8±1.0	36.4±1.9	36.1±1.9	27.7±1.8	27.7±1.8
	AddEdge	75.2±1.2	73.7±1.2	65.0±1.4	60.9±1.3	86.1±2.8	83.4±2.6	23.8±1.7	23.2±1.6	36.5±1.5	36.2±1.3	26.9±2.1	26.9±2.1
	DropEdge	76.0±1.6	74.5±1.6	65.1±1.4	60.9±1.4	86.2±1.6	83.5±1.4	24.1±1.0	23.3±0.9	37.5±1.4	37.2±1.4	27.5±1.8	27.5±1.8
	SDRF	75.7±0.8	74.6±0.8	65.3±0.6	**61.4±0.6**	> 5 days	> 5 days	> 5 days	> 5 days	41.5±2.6	41.6±2.7	> 5 days	> 5 days
	NerualSparse	79.7±1.8	77.8±1.6	64.7±1.4	61.1±1.3	89.1±5.4	86.7±5.5	25.1±1.2	**24.4±1.1**	39.1±1.9	39.0±1.9	32.2±2.4	32.2±2.4
	IDGL	79.2±0.9	78.4±0.8	65.6±0.9	61.3±1.2	90.0±1.0	86.3±1.3	24.0±2.6	22.4±2.7	43.8±3.4	43.0±3.2	33.9±0.9	33.9±0.8
	PASTEL	**81.1±0.8**	**79.8±0.7**	**65.7±1.1**	**61.4±1.4**	**92.0±0.6**	**89.0±1.0**	**26.0±2.4**	23.6±2.7	**47.7±0.9**	**46.9±0.9**	**35.5±1.4**	**35.5±1.4**

Table 2: Weighted-F1 scores and improvements on graphs with different levels of topology-imbalance.

	Cora-L		Cora-M		Cora-H	
	RC	*SC*	*RC*	*SC*	*RC*	*SC*
	0.4130	-0.6183	0.4100	-0.6204	0.4060	-0.6302
	W-F1 (%)	Δ (%)	W-F1 (%)	Δ (%)	W-F1 (%)	Δ (%)
GCN	80.9±0.9	—	78.8±0.8	—	77.5±1.0	—
ReNode	81.3±0.7	↑0.4	79.3±0.8	↑0.5	78.3±1.1	↑0.8
SDRF	81.0±0.7	↑0.1	78.9±0.8	↑0.1	77.9±0.7	↑0.4
IDGL	82.5±1.0	↑1.6	80.4±1.0	↑1.6	81.6±1.1	↑4.1
PASTEL	**82.7±0.9**	**↑1.8**	**81.0±0.9**	**↑2.2**	**81.9±1.1**	**↑4.4**

results are shown in bold and the runner-ups are underlined. PASTEL shows overwhelming superiority in improving the performance of backbones on all datasets. It demonstrates that PASTEL is capable of learning better structures with a more balanced label distribution that reinforces the GNN models. ReNode [7] achieves fewer improvements on datasets of poor connectivity (e.g., CiteSeer) and even damages the performance of backbones on heterophilic datasets (e.g., Chameleon and Actor). We think it's because ReNode [7] detects conflicts by Personalized PageRank and fails to reflect the node topological position well when the graph connectivity is poor. Besides, ReNode takes the topology boundary as the decision boundary, which is not applicable for heterophilic graphs. AddEdge doesn't work in most cases, demonstrating that randomly adding edge is not effective in boosting the reachability. The structure augmentation strategy should be carefully designed considering the node relations. SDRF [42] can improve the performance, supporting our intuition that relieving over-squashing helps graph learning. But SDRF is still less effective than PASTEL because it only considers the topological properties rather than the supervision information. Both NeuralSparse [54] and IDGL [10] show good performance among the baselines, showing the effectiveness of learning better structures for downstream tasks. However, they are still less effective than PASTEL which takes the supervision information distribution into consideration.

5.2.2 PASTEL under Different Levels of Topology-imbalance. To further analyze PASTEL's ability in alleviating the topology-imbalance issue, we verify the PASTEL under different levels of topology-imbalance. We randomly sampled 1,000 training sets and calculate the reaching coefficient *RC* and squashing coefficient *SC* as introduced in Section 3.2. Then we choose 3 training sets with different levels of topology-imbalance according to the conclusion in Section 3.3 and we denote them as Cora-L, Cora-M, and Cora-H, according to the degree of topology imbalance. Note that larger *RC* means better reachability and larger *SC* means lower squashing. We evaluate PASTEL and several baselines with the GCN as the backbone and show the dataset information, the Weighted-F1 scores, and their improvements (Δ) over the backbones in Table 2. The performance of node representation learning generally gets

Table 3: Weighted-F1 scores (%) and improvements (Δ) on synthetic SBM graphs with different community structures.

	SBM-1		SBM-2		SBM-3		SBM-4		SBM-5		SBM-6		SBM-7	
p	0.5000		0.5000		0.5000		0.5000		0.5000		0.5000		0.5000	
q	0.0300		0.0100		0.0083		0.0071		0.0063		0.0056		0.0050	
RC	0.4979		0.4984		0.4990		0.4994		0.5002		0.5004		0.5009	
SC	0.0998		0.0999		0.1000		0.1001		0.1007		0.1017		0.1144	
	W-F1	Δ	W-F1	Δ	W-F1	Δ	W-F1	Δ	W-F1	Δ	W-F1	Δ	W-F1	Δ
GCN	40.29	—	42.37	—	42.99	—	44.13	—	45.19	—	45.21	—	45.22	—
ReNode	41.33	↑1.04	42.40	↑0.03	43.21	↑0.22	44.56	↑0.43	45.20	↑0.01	45.08	↓0.13	44.89	↓0.33
PASTEL	**45.67**	**↑5.38**	**57.61**	**↑15.24**	**58.33**	**↑15.34**	**60.29**	**↑16.16**	**66.41**	**↑21.22**	**66.45**	**↑21.24**	**66.57**	**↑21.35**

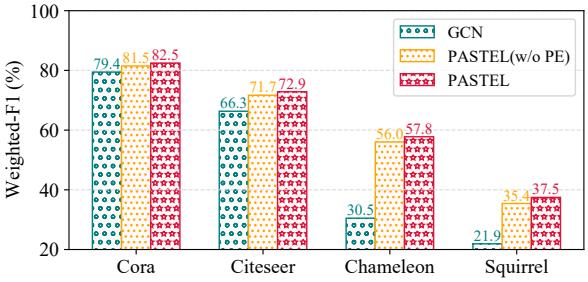


Figure 5: The impact of position encoding.

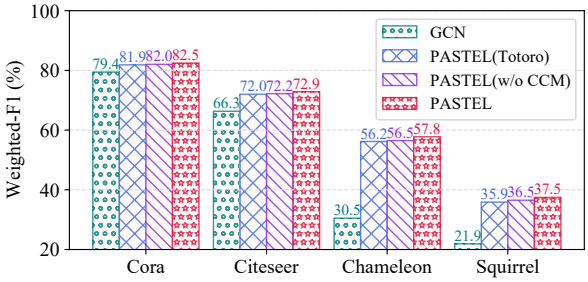


Figure 6: The impact of class-wise conflict measure.

worse with the increase of the topology-imbalance degree of the dataset. Both the node re-weighting method (i.e., ReNode [7]) and the structure learning methods (i.e., IDGL [10], SDRF [42] and PASTEL) can achieve more improvement with the increase of dataset topology-imbalance. PASTEL performs best on all datasets with different degrees of topology-imbalance and it can achieve up to 4.4% improvement on the highly topology-imbalance dataset.

5.2.3 PASTEL for Synthetic Graphs. We generate 7 synthetic graph datasets with different community structures using the Stochastic Block Model (SBM) $\mathcal{G}(N, C, p, q)$ [19], where the number of nodes $N = 3000$, the number of community $C = 6$, p denotes the edge probability within a community and q denotes the edge probability between communities. We show the classification Weighted-F1 scores and improvements are shown in Table 3. With a more clear community structure, the reaching coefficient *RC* increases and the squashing coefficient *SC* also increases, leading to the increase of GCN's performance, which agrees with the conclusion obtained in Section 3.3. ReNode shows unsatisfied performance in boosting the node classification. PASTEL can increase the classification weighted-F1 score by 5.38%-21.35% on SBM graphs with different community structures, showing superior effectiveness.

5.3 Analysis of PASTEL (RQ2)

We conduct ablation studies for the two main mechanisms of PASTEL, position encoding and class-wise conflict measure.

5.3.1 Impact of the Position Encoding. We design an anchor-based position encoding mechanism in Section 4.1, which reflects the relative topological position to labeled nodes and further maximizes the label influence within a class. To evaluate the effectiveness of position encoding, we compare PASTEL with a variant **PASTEL (w/o PE)**, which removes the position encoding and directly take the node features for metric learning in Eq. (7). Here we use the GCN as the backbone. As shown in Figure 5, the structure learning strategy of PASTEL contributes the most, which can achieve at most 25.5% improvement in terms of Weighted-F1 score with only node features. Although PASTEL (w/o PE) effectively improves the performance of backbones to some extent, the position encoding still benefits learning better structure to relieve the topology-imbalance with 1.0%-1.8% improvements than PASTEL (w/o PE).

5.3.2 Impact of the Class-wise Conflict Measure. We designed a class-wise conflict measure in Section 4.2 as edge weights to guide learning structures with better intra-class connectivity. Here, we compare PASTEL with its two variants to analyze the impact of the class-wise conflict measure: (1) **PASTEL (w/o CCM)**, which removes the class-wise conflict measure and directly takes the learned edge possibilities in Eq. (7) as the edge weights. (2) **PASTEL (Totoro)**, which takes the Totoro metric introduced in ReNode [7] as the conflict measure of nodes in Eq. (13). Here we use the GCN as the backbone. The comparison results are shown in Figure 6. On four datasets, PASTEL consistently outperforms the other two variants. Even without the conflict measure, PASTEL (w/o CCM) still shows better performance than PASTEL (Totoro), indicating the limitation of ReNode when capturing the relative topology positions without clear homophily structures.

5.4 Analysis of Learned Structure (RQ3)

We analyze the learned graph by PASTEL in terms of visualization and structural properties.

(a) Original Graph. (b) ReNode. (c) SDRF. (d) IDGL. (e) PASTEL.

Figure 7: Structure visualization. (a) Original graph of Cora and learned graphs by (b) ReNode, (c) SDRF, (d) IDGL and (e) PASTEL.

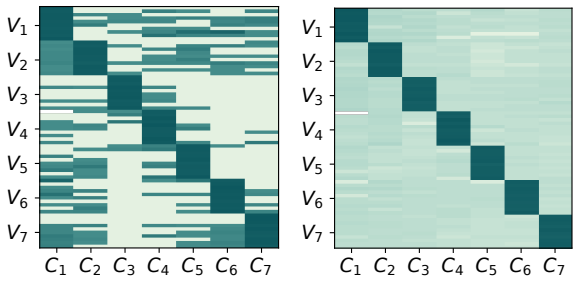
Table 4: Properties and performance of the original graph and learned graphs of Cora.

	Original Graph	ReNode	SDRF	IDGL	**PASTEL**
RC	0.4022	0.4022	0.4686	0.5028	0.5475
SC	-0.6299	-0.6299	-0.4942	-0.4069	-0.3389
W-F1 (%)	79.44	80.34	82.01	82.38	**82.86**

5.4.1 Structure Visualization. In Figure 7, we visualize the original graph of Cora and the graphs learned by ReNode [7], SDRF [42], IDGL [10] and PASTEL using *networkx*. For clarity, the edges are not shown. The solid points denote the labeled nodes, the hollow points denote the unlabeled nodes, and the layout of nodes denotes their connectivities. The node size in Figure 7(b) denotes the learned node weight in ReNode, and the solid lines and dashed lines in Figure 7(c) denote the added and deleted edges by SDRF, respectively. As we can observe, ReNode gives more weights to nodes in the topology center of each class and SDRF tends to build connections between distant or isolated nodes. Even though the structure learned by IDGL can make the nodes of a class close, there are still some overlapping and entangled areas between classes. Benefiting from the position encoding and class-wise conflict measure, PASTEL can obtain graph structure with clearer class boundaries.

5.4.2 Change of RC and SC. We also show the reaching coefficient *RC* and the squashing coefficient *SC* of the above graphs in Figure 7 and the Weighted-F1 score learned on them in Table 4. Here we choose the GCN as the model backbone. All of the structure learning methods (SDRF [42], IDGL [10] and PASTEL) learn structures with larger reaching coefficient and larger squashing coefficient, leading the performance improvement of node classification. This phenomenon supports our propositions in Section 3.3 again.

5.4.3 Change of GPR Vector. The class-wise conflict measure is calculated by the Group PageRank (GPR), which reflects the label influence of each class. We randomly choose 10 nodes for each class in Cora and show their GPR vectors $\mathbf{P}_i^{gpr}$ in the original graph in Figure 8(a) and the learned graph in Figure 8(b), respectively, where the color shade denotes the magnitude, V_i denotes 10 nodes of class i and C_i denotes the i-th class. In Figure 8(a), the off-diagonal color blocks are also dark, indicating that the label influence of each class that nodes obtained from the original graph is still entangled to some extent, which could bring difficulties to the GNN optimization. After the structure learning guided by the proposed class-wise conflict measure, Figure 8(b) exhibits 7 clear diagonal blocks and



(a) Original Graph. (b) Learned Graph.

Figure 8: Heat maps for the Group PageRank value of (a) the original graph and (b) the learned graph by PASTEL.

the gaps between the diagonal and off-diagonal block are widened, indicating that nodes can receive more supervision information of its ground-truth class. We can further make a conclusion that the class-wise conflict measure plays an important role on giving guidance for more class connectivity orthogonality.

6 CONCLUSION

We proposed a novel framework named PASTEL for the graph topology-imbalance issue. We provide a new understanding and two quantitative analysis metrics of topology-imbalance in the perspective of under-reaching and over-squashing, answering the questions that how topology-imbalance affects GNN's performance as well as what graphs are susceptible to it. PASTEL designs an anchor-based position encoding mechanism and a class-wise conflict measure to obtain structures with better in-class connectivity. Comprehensive experiments demonstrate the potential and adaptability of PASTEL. An interesting future direction is to incorporate the proposed two quantitative metrics into the learning process to address topology-imbalance more directly.

ACKNOWLEDGMENTS

The corresponding author is Jianxin Li. The authors of this paper were supported by the NSFC through grant 6187202 and the Academic Excellence Foundation of BUAA for PhD Students. Philip S. Yu was supported by NSF under grants III-1763325, III-1909323, III-2106758, and SaTC-1930941.

REFERENCES

[1] Uri Alon and Eran Yahav. 2021. On the bottleneck of graph neural networks and its practical implications. In *ICLR*.

[2] Shin Ando and Chun Yuan Huang. 2017. Deep over-sampling framework for classifying imbalanced data. In *ECML-PKDD*. Springer, 770–785.

[3] Eliav Buchnik and Edith Cohen. 2018. Bootstrapped graph diffusions: Exposing the power of nonlinearity. In *SIGMETRICS*. 8–10.

[4] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. 2019. Learning imbalanced datasets with label-distribution-aware margin loss. In *NeurIPS*, Vol. 32.

[5] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[6] Deli Chen, Yanyai Lin, Lei Li, Xuancheng Ren Li, Jie Zhou, Xu Sun, et al. 2020. Distance-wise graph contrastive learning. *arXiv preprint arXiv:2012.07437* (2020).

[7] Deli Chen, Yankai Lin, Guangxiang Zhao, Xuancheng Ren, Peng Li, Jie Zhou, and Xu Sun. 2021. Topology-Imbalance Learning for Semi-Supervised Node Classification. *NeurIPS* 34 (2021).

[8] Hao Chen, Yue Xu, Feiran Huang, Zengde Deng, Wenbing Huang, Senzhang Wang, Peng He, and Zhoujun Li. 2020. Label-aware graph convolutional networks. In *CIKM*. 1977–1980.

[9] Jinyu Chen, Chen Gao, Erli Meng, Qiong Zhang, and Si Liu. 2022. Reinforced Structured State-Evolution for Vision-Language Navigation. In *CVPR*. 15450–15459.

[10] Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in Neural Information Processing Systems* 33 (2020), 19314–19326.

[11] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. Class-balanced loss based on effective number of samples. In *CVPR*. 9268–9277.

[12] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In *ICML*. PMLR, 1972–1982.

[13] Xingcheng Fu, Jianxin Li, Jia Wu, Qingyun Sun, Cheng Ji, Senzhang Wang, Jiajun Tan, Hao Peng, and S Yu Philip. 2021. ACE-HGNN: Adaptive curvature exploration hyperbolic graph neural network. In *ICDM*. IEEE, 111–120.

[14] Chen Gao, Jinyu Chen, Si Liu, Luting Wang, Qiong Zhang, and Qi Wu. 2021. Room-and-object aware knowledge reasoning for remote embodied referring expression. In *CVPR*. 3064–3073.

[15] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*. PMLR, 1263–1272.

[16] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. 2017. Learning from class-imbalanced data: Review of methods and applications. *Expert systems with applications* 73 (2017), 220–239.

[17] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.

[18] Haibo He and Edwardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering* 21, 9 (2009), 1263–1284.

[19] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. 1983. Stochastic blockmodels: First steps. *Social networks* 5, 2 (1983), 109–137.

[20] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *SIGKDD*. 66–74.

[21] Vassilis Kalofolias. 2016. How to learn a graph from smooth signals. In *(AISTATS*. PMLR, 920–929.

[22] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[23] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*.

[24] Jianxin Li, Xingcheng Fu, Qingyun Sun, Cheng Ji, Jiajun Tan, Jia Wu, and Hao Peng. 2022. Curvature Graph Generative Adversarial Networks. In *WWW*. 1528–1537.

[25] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *ECCV*. 2980–2988.

[26] Vivi Nastase, Rada Mihalcea, and Dragomir R Radev. 2015. A survey of graphs in natural language processing. *Natural Language Engineering* 21, 5 (2015), 665–698.

[27] Chien-Chun Ni, Yu-Yao Lin, Jie Gao, Xianfeng David Gu, and Emil Saucan. 2015. Ricci curvature of the internet topology. In *INFOCOM*. IEEE, 2758–2766.

[28] Yann Ollivier. 2009. Ricci curvature of Markov chains on metric spaces. *Journal of Functional Analysis* 256, 3 (2009), 810–864.

[29] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[30] Joonhyung Park, Jaeyun Song, and Eunho Yang. 2021. GraphENS: Neighbor-Aware Ego Network Synthesis for Class-Imbalanced Node Classification. In *ICLR*.

[31] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. In *ICLR*.

[32] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *ICML*. PMLR, 4334–4343.

[33] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. Dropedge: Towards deep graph convolutional networks on node classification. In *ICLR*.

[34] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* 9, 2 (2021), cnab014.

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

[36] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *NAACL*.

[37] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[38] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Xingcheng Fu, Cheng Ji, and S Yu Philip. 2022. Graph structure learning with variational information bottleneck. In *AAAI*, Vol. 36. 4165–4174.

[39] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Yuanxing Ning, Philip S Yu, and Lifang He. 2021. Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism. In *WWW*. 2081–2091.

[40] Qingyun Sun, Hao Peng, Jianxin Li, Senzhang Wang, Xiangyun Dong, Liangxuan Zhao, S Yu Philip, and Lifang He. 2020. Pairwise learning for name disambiguation in large-scale heterogeneous academic networks. In *ICDM*. IEEE, 511–520.

[41] Yanmin Sun, Andrew KC Wong, and Mohamed S Kamel. 2009. Classification of imbalanced data: A review. *International journal of pattern recognition and artificial intelligence* 23, 04 (2009), 687–719.

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

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[44] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph Attention Networks. In *ICLR*.

[45] Hongwei Wang and Jure Leskovec. 2021. Combining Graph Convolutional Neural Networks and Label Propagation. *ACM Transactions on Information Systems (TOIS)* 40, 4 (2021), 1–27.

[46] Yu Wang, Charu Aggarwal, and Tyler Derr. 2021. Distance-wise Prototypical Graph Neural Network in Node Imbalance Classification. *arXiv preprint arXiv:2110.12035* (2021).

[47] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[48] Xu Xiaolong, Chen Wen, and Sun Yanfei. 2019. Over-sampling algorithm for imbalanced data classification. *Journal of Systems Engineering and Electronics* 30, 6 (2019), 1182–1191.

[49] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware graph neural networks. In *ICML*. PMLR, 7134–7143.

[50] Jinze Yu, Jiaming Liu, Xiaobao Wei, Haoyi Zhou, Yohei Nakata, Denis Gudovskiy, Tomoyuki Okuno, Jianxin Li, Kurt Keutzer, and Shanghang Zhang. 2022. Cross-Domain Object Detection with Mean-Teacher Transformer. In *ECCV*.

[51] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. 2019. Hierarchical graph pooling with structure learning. *arXiv preprint arXiv:1911.05954* (2019).

[52] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2020. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[53] Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, and Yanfang Ye. 2021. Heterogeneous graph structure learning for graph neural networks. In *AAAI*.

[54] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. 2020. Robust graph representation learning via neural sparsification. In *ICML*. PMLR, 11458–11468.

[55] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. 2022. Deep Graph Structure Learning for Robust Representations: A Survey. In *IJCAI*.