Generalized Equivariance and Preferential Labeling for GNN Node Classification

Zeyu Sun[1], Wenjie Zhang[1], Lili Mou[2], Qihao Zhu[1], Yingfei Xiong[1], Lu Zhang[1]

[1] Key Laboratory of High Confidence Software Technologies, MoE;
School of Computer Science, Peking University, 100871, P. R. China
[2] Department of Computing Science, Alberta Machine Intelligent Institute (Amii)
University of Alberta, Edmonton T6G 2E8, Canada
{szy_, zhang_wen_jie, zhuqh, xiongyf, zhanglucs}@pku.edu.cn, doublepower.mou@gmail.com

Abstract

Existing graph neural networks (GNNs) largely rely on node embeddings, which represent a node as a vector by its identity, type, or content. However, graphs with unattributed nodes widely exist in real-world applications (e.g., anonymized social networks). Previous GNNs either assign random labels to nodes (which introduces artefacts to the GNN) or assign one embedding to all nodes (which fails to explicitly distinguish one node from another). Further, when these GNNs are applied to unattributed node classification problems, they have an undesired equivariance property, which are fundamentally unable to address the data with multiple possible outputs. In this paper, we analyze the limitation of existing approaches to node classification problems. Inspired by our analysis, we propose a generalized equivariance property and a Preferential Labeling technique that satisfies the desired property asymptotically. Experimental results show that we achieve high performance in several unattributed node classification tasks.

1 Introduction

Graphs are a widely used type of data structure in computer science. A graph can be represented as $G = \langle V, E\rangle$, where V is a set of *nodes*, and E is a set of node pairs known as *edges* (directed or undirected). With the prosperity of deep learning techniques, graph neural networks (GNNs) are shown to be effective to various graph-related applications, such as program analysis (Mou et al. 2016), social networks (Hamilton, Ying, and Leskovec 2017), knowledge graphs (Hamaguchi et al. 2017), molecule analysis (Scarselli et al. 2009), and the satisfiability (SAT) problem (Zhang et al. 2020).

Existing GNNs highly rely on node embeddings, which are a vector representation of a node, typically based on its identity, type, or content. For example, a GNN for a knowledge graph typically embed an entity/concept (e.g., a “cat” and a “mammal”) as a vector (Wang, Ye, and Gupta 2018), whereas a GNN for molecules embeds the atom (e.g., hydrogen and oxygen atoms) as a vector (Scarselli et al. 2009).

In many applications, however, the nodes in a graph may not be attributed, and we call such a graph an *unattributed graph*. A common scenario is that there is no attribute related to the nodes. For example, community detection for large-scale social networks may lack the identity information of nodes, i.e., a person, possibly due to privacy concerns (Backstrom, Dwork, and Kleinberg 2007). Another scenario is that the attribute of a node is an artefact and captures no semantic meanings. Figure 1 shows a graph that represents a propositional satisfiability (SAT) problem (Selsam et al. 2019), where x_1 and c_1 are arbitrary namings of the *literals* (variable or its negations) and *clauses* (disjunction of literals), and could be renamed without changing the nature of the formula. If such an identifier is represented by table look-up embeddings, it would become an *artefact* in the GNN, because these embeddings do not represent common knowledge among different training samples. Nor do they generalize to new samples.

To encode unattributed graphs, previous methods typically adopt an arbitrary labeling for nodes and represent them by embeddings (Allamanis, Brockschmidt, and Khademi 2018; Wei et al. 2020). As mentioned, this introduces artefacts to GNNs. Recently, Selsam et al. (2019) have realized that such artefacts are undesired, and assign all nodes with the same embedding. However, this approach may suffer from the problem that the graph neural network becomes insensitive to the nodes.

In this work, we analyze unattributed node classification tasks, which require *equivariance*, i.e., the change of node labels should be reflected correspondingly in the output. To address the mentioned problems, a naïve idea is to still assign different node embeddings, but to eliminate such artefacts by an ensemble of multiple labelings. For training, the labeling is randomly sampled every time we process a data sample; during inference, an ensemble of multiple random labelings is adopted for a sample. In this way, the nodes are distinguishable given any labeling, but such artefacts are smoothed out by the ensemble average.

Our theoretical analysis, however, shows that such a naïve treatment does not work well for node classification. An equivariant GNN is unable to solve equivariant node classification problems where multiple outputs are appropriate for an input graph.

To this end, we propose a generalized equivariance property that is more suited to unattributed node classification. We further propose a Preferential Labeling approach, which

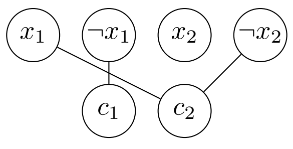

Figure 1: A SAT formula can be represented by a graph, where a node x_i is a *literal* (a variable or its negation) and a node c_i is a *clause* (disjunction of literal nodes). The corresponding SAT formula is the conjunction of clauses, and in this example, it is $\neg x_1 \wedge (x_1 \vee \neg x_2)$.

assigns multiple labelings during training but only updates the parameters with the best labeling. For inference, we also assign multiple labelings and make a prediction according to the best one. In this way, Preferential Labeling asymptotically achieves our generalized equivariance property, and works well for multi-output equivariant node classification.

We evaluated our approach on two unattributed node classification applications, maximum independent set solving (MIS) and propositional satisfiability solving (SAT)[1]. Experimental results show that our approach successfully alleviates the limitations of existing GNNs when encoding unattributed graphs, where the number of errors drops by 39% in the MIS problem and 76% in the SAT problem.

2 Methodology

In this section, we first present the problem formulation and analyze the equivariance property on unattributed graphs. Then, we present our Preferential Labeling approach to address equivariant node classification.

2.1 Problem Formulation

A problem on unattributed graphs can be formalized as predicting output Y given a graph X. A predicate function $H(X, Y)$, specific to a task, determines if Y is appropriate for a given X. The predicate is true if and only if Y is an appropriate solution for X.

For an unattributed graph $G = \langle V, E \rangle$, the input can be fully represented by an adjacency matrix $X \in \{0,1\}^{n\times n}$, where n is the number of nodes. In a node classification task, the output is a matrix $Y \in \mathbb{R}^{n\times k}$ for n nodes and k classes.

To analyze how node indexes affect (or shall not affect) a GNN, we introduce the notation S_n to represent the permutation group for $[n]$. Given $\pi \in S_n$, the action of π on an unattributed graph $X \in \{0,1\}^{n\times n}$ is defined as $(\pi(X))_{i,j} = X_{(\pi^{-1}(i)),(\pi^{-1}(j))}$, and its corresponding action on $Y \in \mathbb{R}^{n\times k}$ is given by $(\pi(Y))_{i,c} = Y_{(\pi^{-1}(i)),c}$, i.e., π denotes the same shuffle on the rows and columns of X, as well as the rows of Y. Here, π is the mapping from node indexes to permuted indexes. Thus, π^{-1} is retrieving the original node indexes in X and Y from the permuted indexes i and j in $\pi(X)$ and $\pi(Y)$.

[1]The code and data are available at
https://github.com/zysszy/Preferential-Labeling

Equivariance. We now formulate the *equivariance property* of node classification tasks. It essentially asserts that for any permutation $\pi \in S_n$,

$$H(X, Y) \quad \text{implies} \quad H(\pi(X), \pi(Y)) \tag{1}$$

That is to say, if we permute the order of nodes, the solution should be changed correspondingly.

Suppose for every X there exists a unique Y satisfying $H(X, Y)$, the mapping from X to Y can be modeled by a function h and the equivariance property becomes the form that we commonly see

$$h(\pi(X)) = \pi(h(X)) \tag{2}$$

for every permutation $\pi \in S_n$.

In the above formulation, we define the equivariance property of a node classification task. In fact, equivariance can also be said in terms of GNN output $f(X)$, given by

$$f(\pi(X)) = \pi(f(X)) \tag{3}$$

2.2 Limitations of Existing GNNs on Unattributed Graphs

We analyze the limitations of existing GNNs on unattributed graphs. As mentioned in Section 1, previous approaches for unattributed graph either assign random labels to nodes (Allamanis, Brockschmidt, and Khademi 2018; Wei et al. 2020) or assign the same embedding to all nodes (Li, Chen, and Koltun 2018; Selsam et al. 2019). When they are applied to unattributed node classifications, they suffer from at least one of the two limitations: 1) node distinction and 2) equivariance property.

Node Distinction. We first consider distinguishing different nodes in a graph. The state-of-the-art approaches (Li, Chen, and Koltun 2018; Selsam et al. 2019; Zhang et al. 2020) assign all nodes with the same embedding, and thus, the model cannot distinguish different nodes effectively. Consider a common graph convolutional network (GCN), which learns the hidden representation for a node by encoding the node vector with the neighbors via a set of fully-connected layers. In the example given by Figure 2, all four nodes will have the same hidden representation, because every node is represented by the same embedding and all nodes have the same neighboring information.

Equivariance Property. We now consider the equivariance property of node classification for unattributed graphs, which is believed to be important for various GNN applications (Chen, Li, and Bruna 2018; Azizian et al. 2020).

In node classification for unattributed graphs, if the node index changes, the output would change accordingly, shown in Eqn (1). Thus, it is tempting to design an equivariant GNN satisfying Eqn (3) for node classification tasks, as suggested by Wu et al. (2021). Otherwise, the GNN would be sensitive to labeling artefacts (Allamanis, Brockschmidt, and Khademi 2018; Wei et al. 2020), if it does not satisfy some form of equivariance. Previous work achieves the equivariance property (3) by using the same embeddings for all nodes (Li, Chen, and Koltun 2018; Selsam et al. 2019; Zhang et al. 2020).

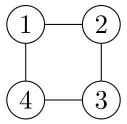

Figure 2: Graph C_4, a circle of length 4. This graph is auto-isomorhpic under $\pi : 1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 4, 4 \mapsto 1$.

However, we hereby show that an equivariant GNN satisfying (3) will fail on node classification problems, where multiple outputs are appropriate. In other words, the mapping from X to Y is not a function, and given an input X, there exists multiple Y such that $H(X, Y)$ holds. Usually, GNN predicts one appropriate Y by a function $Y = f(X)$.

We show the drawback of equivariant GNNs with an example of a non-trivial auto-isomorphic graph, i.e., there exists a non-identity permutation π such that $\pi(X) = X$. If (3) holds, then $\pi(X) = X$ implies $f(X) = \pi(f(X))$. This means that GNN output must be the same for all corresponding nodes shuffled by π.

This, unfortunately, may be a bad solution for various tasks. Consider the maximum independent set (MIS) problem that selects the largest number of vertices that are not directly connected. In Figure 2, for example, $\{1, 3\}$ is an MIS and $\{2, 4\}$ is also an MIS. However, an equivariant GNN cannot predict any MIS in this example, because there exists a permutation π (e.g., $\pi : 1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 4, 4 \mapsto 1$) essentially tying the output of all nodes.

It should be mentioned that we show the limitation by an example of non-trivial auto-isomorphism. Our analysis is suggestive to real applications, where graphs often have similar local structures.

2.3 Our Solution

We start with a naïve attempt to address both node distinction and equivariance in GNNs. With further analysis, we propose a generalized equivariance property and our Preferential Labeling approach.

A Naïve Attempt. To address the limitation of the node distinction, a naïve idea is still to assign node embeddings by randomly labeling the nodes, but to use an ensemble of different node labelings to eliminate artefacts (Murphy et al. 2019; Sato, Yamada, and Kashima 2021). For training, node labels are assigned randomly; this serves as a way of data augmentation, and can be thought of as training an ensemble over epochs. During inference, it assigns multiple random labels and uses an average ensemble for prediction.

However, it is not appropriate if we directly apply such a naïve idea to equivariant node classification. The standard cross-entropy training is essentially to

$$\underset{\omega}{\text{minimize}} \sum_{(X,Y)\in\mathcal{D}} \sum_{\pi\in S_n} \sum_{i=1}^{n} D_{\text{KL}}\left((\pi(Y))_i \mid\mid f_i(\pi(X);\omega)\right),$$

where ω denotes the trainable parameters in a GNN, $\mathcal{D}$ is the training set, and D_{KL} is the Kullback–Leibler divergence between the predictions and the ground truth. $(\pi(Y))_i$ and $f_i(\pi(X))$ are the ith row in a matrix, representing the target and predicted distributions of a node. However, the training objective will enforce the GNN to learn the same prediction of nodes under auto-isomorphism, because for every $\pi \in S_n$, the KL objective requires $\pi(Y) = f(\pi(X))$. For some auto-isomorphic permutation τ, i.e., $\tau(X) = X$, this implies $f(X) = f(\tau(X)) = \tau(Y)$. Since $Y = f(X)$, we will have $f(X) = \tau(f(X))$. This is precisely the limitation that we have analyzed in Section 2.2, namely, auto-isomorphism tying the prediction of a GNN.

To address this issue, we propose a desired generalized equivariance property.

Generalized Equivariance Property. An equivariant GNN satisfying (3) fails for equivariant node classification, because it unreasonably assumes the output is a function of input, i.e., approximating (1) by (2).

We relax this constraint for multi-output node classification and analyze the desired form of equivariance in this setting. We denote $\mathcal{H}(X) = \{Y : H(X, Y)\}$ be the set of all correct outputs given a graph X. The training set typically provides one groundtruth solution $Y_* \in \mathcal{H}(X_*)$ for a specific graph X_*, as usually one solution suffices in real applications (and this coincides with GNNs whose output is a function of input).

We would define $\mathcal{H}_*(\cdot)|_{\mathcal{D}}$ as a minimal equivariant subset of $\mathcal{H}(\cdot)$ such that $Y_* \in \mathcal{H}_*(X_*)$, with the domain restricted to $\mathcal{D} = \{X : X = \pi(X_*) \text{ for some } \pi \in S_n\}$.

This starts from defining

$$\mathcal{H}_*(X_*) = \{\gamma(Y_*) : \gamma \in S_n \text{ and } \gamma(X_*) = X_*\}, \quad (4)$$

which essentially endorses multiple correct outputs other than the given Y_* due to self-isomorphism γ.

Then, the equivariance property suggests $\mathcal{H}_*(X) = \pi(\mathcal{H}_*(X_*))$, if $X = \pi(X_*)$ for some π. Here, we abuse the notation as $\pi(\mathcal{H}_*(X_*)) \triangleq \{\pi(Y) : Y \in \mathcal{H}_*(X_*)\}$.

We would like to design a neural network predicting a correct solution, i.e.,

$$f(X_*) \in \mathcal{H}_*(X_*). \quad (5)$$

Due to the equivariance of $\mathcal{H}_*$, we have

$$f(\pi(X_*)) \in \mathcal{H}_*(\pi(X_*)) = \pi(\mathcal{H}_*(X_*)). \quad (6)$$

By the definition of $\mathcal{H}_*$ in (4), Eqn (5) implies that there exists $\gamma_1 \in S_n$ such that $\gamma_1(X_*) = X_*$ and $f(X_*) = \gamma_1(Y_*)$. Likewise, Eqn (6) implies that there exists $\gamma_2 \in S_n$ such that $\gamma_2(X_*) = X_*$ and $f(\pi(X_*)) = \pi\gamma_2(Y_*)$. Combining these and denoting $\gamma_2\gamma_1^{-1}$ by γ, we see that there exists $\gamma \in S_n$ such that

$$\gamma(X_*) = X_* \quad \text{and} \quad f(\pi(X_*)) = \pi\gamma(f(X_*)). \quad (7)$$

We call (7) the *generalized equivariance property*. In fact, (3) is a special case of (7), where γ is an identity permutation. However, we relax (3) by allowing an additional auto-isomorphic permutation γ in the solution space, and thus, it does not suffer from the limitation in Section 2.2.

The analysis shows that an ideal GNN for unattributed node classification should satisfy (7) rather than (2).

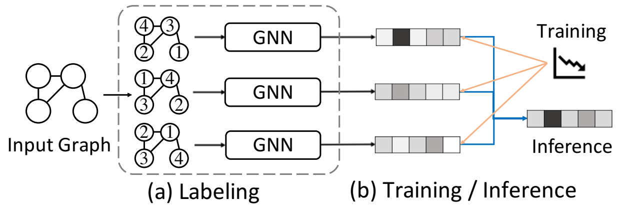

Figure 3: An overview of our Preferential Labeling approach.

Preferential Labeling. Inspired by the above analysis, we propose a simple yet effective approach, Preferential Labeling, which asymptotically satisfies (7). The overview of our approach is shown in Figure 3.

For training, Preferential Labeling assigns nodes with a random permutation of labels. A node is represented by a table-lookup embedding based on the assigned label. To satisfy (7), we dynamically sample multiple label assignments for an input graph in each epoch, but only train the GNN with the best labeling (i.e., having the lowest loss). When processing the graph in the next epoch, we re-assign node labels and lookup for a (possibly) different preferred embedding by other random permutations.

Formally, we allocate $\boldsymbol{e}_1, \cdots, \boldsymbol{e}_N$ as embedding parameters in our GNN model, where N is the total number of embeddings; these embeddings have not been associated with any graph or node. In each epoch when processing a graph that has nodes $V = \{v_1, \cdots, v_n\}$ with $n \le N$, we randomly sample a permutation $\pi \in S_n$. By the convention of our paper, π operates on adjacency matrices; consequently, a node v_i is now represented by $\boldsymbol{e}_{\pi^{-1}(i)}$ for GNN processing.

We repeat this sampling process K times, and compute the loss of these permutations when fed to GNN. Finally, we select the permutation that has the lowest loss as the final permutation for training. In different training epochs, these permutations are re-sampled even for the same data sample.

The training process can be described as

$$\underset{\omega}{\text{minimize}} \sum_{(X,Y)\in\mathcal{D}} \min_{\pi\in S_n} \sum_{i=1}^{n} D_{\text{KL}}\left((\pi(Y))_i \;||\; f_i(\pi(X);\omega)\right), \tag{8}$$

where we train the model with the best labeling π, which is a permutation of both X and Y matrices (Section 2.1).

Consider the inference of a k-way classification problem. The GNN with permutation π_m outputs a probability as $\left(p_1^{(m)}, \cdots, p_k^{(m)}\right)$. We pick the prediction that has the highest joint predicted probability (product of node probabilities). Eventually, our Preferential Labeling approach predicts a label by $c = \text{argmax}_i\{p_i\}$.

Our Preferential Labeling does not suffer from the limitation of equivariance, because our network is not an equivariant function as in (2). However, we will achieve the generalized equivariance property (7) asymptotically, because the labeling of X is optimized out (except for auto-isomorphism) during training and inference by taking the preferred π, detailed below.

Theoretical Analysis. We show that the inference of our Preferential Labeling asymptotically satisfies the generalized equivariance property for node classification, if we have enough sampled permutations. We also draw a connection with Expectation–Maximization (EM) algorithms.

During our inference, we assign multiple labelings to a graph, and pick the prediction that has the highest predicted probability as our output. Formally, we consider the joint predicted probability of a graph X with labeling τ as

$$s(X,\tau) = \prod_{i=1}^{n} \max_{j=1,\cdots,k} f_{ij}(\tau(X)), \tag{9}$$

where f denotes a GNN function, outputing an $n \times k$ matrix (n: the number of nodes, k: the number of category). The ith row is the predicted distribution of the ith node.

During inference, we have multiple labelings τ. We pick the best one that maximizes $s(X,\tau)$, given by

$$\tau_*^{(X)} = \underset{\tau\in S_n}{\text{argmax}}\, s(X,\tau). \tag{10}$$

The prediction of our Preferential Labeling is

$$\hat{Y}(X) = \left(\tau_*^{(X)}\right)^{-1}\left(f\left(\tau_*^{(X)}(X)\right)\right). \tag{11}$$

The formula follows our convention of adjacency matrix representations. When we perform node labeling, we permute the indexes of both X and Y by $\tau_*^{(X)}$. After GNN processing, we need an inverse permutation $\left(\tau_*^{(X)}\right)^{-1}$ to obtain the predictions for X, because our prediction should be corresponding to the original graph X, rather than $\tau_*^{(X)}(X)$.

Theorem 1. *$\hat{Y}(\cdot)$ achieves generalized equivariance, i.e., for any graph X and permutation $\pi \in S_n$, there exists $\gamma \in S_n$ such that $\gamma(X) = X$ and $\hat{Y}(\pi(X)) = \pi\gamma(\hat{Y}(X))$.*

Proof. Consider any graph X and permutation π. Replacing X by $\pi(X)$ in Eqn (10), we have

$$\tau_*^{(\pi(X))} = \underset{\tau\in S_n}{\text{argmax}}\, s(\pi(X),\tau). \tag{12}$$

Notice that Eqns (12) and Eqn (10) are essentailly the same problem, and that their optima should be achieved by the same element, i.e., $\tau_*^{(X)}(X) = \tau_*^{(\pi(X))}\pi(X)$.

This essentially means that the two permutations $\tau_*^{(X)}$ and $\tau_*^{(\pi(X))}\pi$ yield the same result on X, implying that they are the same, except for an auto-isomorphic permutation. In other words, there exists γ such that $\gamma(X) = X$ and $\tau_*^{(X)} = \tau_*^{(\pi(X))}\pi\gamma$, which can be rearranged as

$$\tau_*^{(\pi(X))} = \tau_*^{(X)}\gamma^{-1}\pi^{-1}. \tag{13}$$

Replacing X by $\pi(X)$ in Eqn (11), we have

$$\begin{aligned}
\hat{Y}(\pi(X)) &= (\tau_*^{(\pi(X))})^{-1}(f(\tau_*^{(\pi(X))}\pi(X))) && (14)\\
&= \pi\gamma(\tau_*^{(X)})^{-1}(f(\tau_*^{(X)}\gamma^{-1}\pi^{-1}\pi(X))) && (15)\\
&= \pi\gamma(\tau_*^{(X)})^{-1}(f(\tau_*^{(X)}(X))) && (16)\\
&= \pi\gamma(\hat{Y}(X)), && (17)
\end{aligned}$$

where (15) is due to the substitution with (13); (16) is due to the cancellation of $\pi^{-1}\pi$ and the auto-isomorphism of γ, i.e., $\gamma(X) = X$; and (17) is due to the definition of $\hat{Y}$ in (11). □

Our Preferential Labeling is also related to EM algorithms.

Theorem 2. *The training of Preferential Labeling in (8) is a hard Expecatation–Maximization algorithm with a uniform prior on S_n.*

Proof. The labeling π can be thought of as a latent variable in the task of mapping a graph X to output Y. The $\min$ operator in (8) is to seek $\pi \in S_n$ maximizing the likelihood of output given input and the latent labeling, denoted by $P(Y|X, \pi)$. With the assumption of uniform prior $P(\pi)$ for $\pi \in S_n$, this is equivalent to cross-entropy training with one latent labeling π that maximizes the posterior $P(\pi|X, Y) \propto P(Y|X, \pi)P(\pi)$, known as hard EM (Samdani, Chang, and Roth 2012). □

This easy theorem provides further intuition on our Preferential Labeling approach. EM algorithms are known for handling multi-modal mixtures of distributions, similar to multi-output node classification. The training of our Preferential Labeling is also analogous to the E-step, which determines the fitness of the sample to a mixture component. Our approach adopts a hard EM variant that selects a single best permutation, because full marginalization of S_n is intractable. Also, we assume a uniform prior for S_n, which is particularly suitable for eliminating labeling artefacts.

3 Experiments

We conducted experiments on two node classification tasks. We chose state-of-the-art or standard GNN architectures, but compare our approach with various embedding strategies.

3.1 Competing Methods

Static Labeling. The static labeling assigns an embedding based on the identity of a node (e.g., x_1 and c_1 in Figure 1), although such identity does not represent meaningful semantics in different samples. Static learning is widely applied in previous work (Boldi et al. 2012; Allamanis, Brockschmidt, and Khademi 2018).

Same Embedding. This baseline assigns all nodes in the unattributed graph with the same embedding. This is adopted in previous work (Li, Chen, and Koltun 2018; Zhang et al. 2020) to eliminate the artefacts of node labeling.

Random Labeling. The random labeling assigns an embedding randomly during training and inference. This is a special case of our approach with $K = 1$, and no actual preferential training is performed.

Degree Feature. An intuitive way to encode a node without labeling artefact is by its degree information, which captures some local information of the node. In this baseline, we use $1/(d_v + 1)$ as a one-dimensional, non-learnable embedding feature, where d_v is the degree of node v. We use the above formula so that the feature is in $(0, 1]$.

Row #	GCN (Li, Chen, and Koltun 2018)	Accuracy
1	Same	75.59%
2	Degree Feature	73.22%
3	Degree Ranking Embedding	71.58%
4	Static Labeling	74.57%
5	Random Labeling	75.28%
6	Preferential Labeling-10	**85.04**%

Table 1: The results for MIS solving. "Preferential Labeling-10" indicates 10 random labelings in both training and inference.

Degree Ranking Embedding. The drawbacks of using degree information as a single numerical feature are its low-dimensionality and non-learnability. In this baseline, we extend the idea by embedding the ranking of node degrees. Specifically, we sort all nodes by the degrees in descending order, and a node having ith largest degree is encoded by ith embedding vector $\boldsymbol{e}_i$.

3.2 Experiment I: MIS Solving

We first evaluate our Preferential Labeling on solving the maximum independent set (MIS). In graph theory, an *independent set* is a set of nodes without any edge. An independent set is *maximum*, if it has the largest number of nodes among all independent sets. MIS solving is an NP-hard problem that aims to find out a maximum independent set from a graph.

For an input graph, the goal of the GNN in this task is to predict a binary label for each node, deciding whether a node is in the MIS. To induce an MIS from model predictions, we use a simple search algorithm. We first sort all nodes in descending order based on the predicted probability that the node is in the MIS. Then, we iterate over nodes in order and select the top node into the MIS; its neighbors are removed from the list. The process is iterated until we have processed the entire node list. In this way, the selected nodes are guaranteed to be an independent set. Our evaluation determines whether it is maximum.

Model. In this experiment, we adopt the state-of-the-art model (GCN; Li, Chen, and Koltun 2018) for MIS solving. Li, Chen, and Koltun (2018) use the same embedding for all nodes. The model contains 20 graph convolutional layers, which are regularized by dropout with rate of 0.1. For the hidden size of all layers used in this model, we set it to 128. For training, we use Adam (Kingma and Ba 2015) to train the model with learning rate 10^{-4} on a single Titan RTX.

Dataset. We follow the data synthesis process in previous work (Li, Chen, and Koltun 2018) and generate 173,751, 20,000, 20,000 graphs for training, development, and test, respectively. The number of nodes in a graph is generated from a uniform distribution $U[100, 150]$.

Results. Table 1 shows the results for MIS solving. Since our post-processing ensures the output is an independent set, the performance evaluation focuses on whether it is maximum. If the predicted set has the same number of nodes as the groundtruth MIS, we say the model solves this MIS correctly; otherwise, the model makes an error.

		Error Rate				
Row #	NLocalSAT (Zhang et al. 2020)	Test-5	Test-10	Test-20	Test-40	Avg.
1	Same	5.26%	8.17%	15.03%	27.62%	14.02%
2	Degree Feature	5.31%	8.37%	14.25%	24.94%	13.22%
3	Degree Ranking Embedding	5.45%	10.23%	16.17%	28.04%	14.97%
4	Static	6.11%	9.86%	16.89%	28.88%	15.44%
5	Static & Inference-10 (Averaging)	5.00%	8.77%	15.74%	29.70%	14.80%
6	Static & Inference-10 (Max Prob.)	1.77%	3.65%	7.86%	16.22%	7.38%
7	Random	3.38%	6.17%	12.70%	23.66%	11.48%
8	Random & Inference-10 (Averaging)	3.39%	6.07%	12.42%	23.34%	11.31%
9	Random & Inference-10 (Max Prob.)	2.72%	5.03%	11.37%	22.06%	10.30%
10	Preferential Labeling-10 (Max Prob.)	**1.13**%	**1.68**%	**1.81**%	**5.24**%	**2.47**%

Table 2: The results for SAT solving. "Test-k" indicates a test set where each sample has k variables. "Inference-m" indicates m random labelings during inference.

We observe that Static Labeling (Row 4) has low performance as it introduces labeling artefacts. Same and Random Labelings (Rows 1 and 5) eliminate such artefacts and perform better. The Degree Feature (Row 2) and Degree Ranking Embedding (Row 3) suffer from the limitation of the equivariance property mentioned in Section 2.2, and perform worse than other baselines.

By contrast, Preferential Labeling (Row 6) is able to eliminate labeling artefacts, and at the same time, achieve the desired generalized equivariance property in Eqn (7). Its performance is higher than all competing approaches, with the number of errors dropping by 39% from the best baseline.

3.3 Experiment II: SAT Solving

We further evaluate Preferential Labeling on the SAT solving problem. The propositional satisfiability problem (SAT) is one of the most fundamental problems in computer science. A propositional formula is said to be *satisfiable*, if there exists an assignment of propositional variables to either True or False that makes the formula True; such assignment is known as a *certificate*.

We consider a specific setting of SAT solving, where the given formula is known to be satisfiable, and the goal is to predict a certificate, i.e., whether a variable should be assigned with True or False. This is a key step in SAT solvers.

Model. The GNN model and settings are generally adopted from the state-of-the-art NLocalSAT (Zhang et al. 2020).

A SAT formula is represented as a bipartite graph, where a node is either a clause or a literal (see Figure 1 for an example). The nodes are represented by identifiers, which are labeling artefacts. In our experiment, we applied a convolution-based NLocalSAT model (Zhang et al. 2020), which achieves state-of-the-art performance for SAT solving. Zhang et al. (2020) use the same embedding for all clause/literal nodes. The model has 16 convolutional layers, regularized by a dropout rate of 0.1. In our model, we perform Preferential Labeling for literals and clauses from respective candidate labelings/embeddings.

Dataset. We used the SAT dataset in Zhang et al. (2020). The training and development sets contain 500K and 396K SAT formulas, respectively. The number of variables in a formula is generated from a uniform distribution $U[10, 40]$, whereas the number of clauses is generated from $U[2, 6]$. For testing, the dataset contains four sets of different levels of difficulty. Specifically, the number of variables in a formula is 5, 10, 20, or 40 in each test set, denoted by Test-5, Test-10, Test-20, or Test-40, each containing 40K, 20K, 10K, or 5K test formulas.

Results. Table 2 shows the results for SAT solving, where the performance of a model is evaluated by the formula-level error rate, i.e., if the predicted assignment does not make the formula true, we say the model makes an error.

As mentioned, Static Labeling (Row 4) introduces artefacts of node identities, whereas using the same embedding (Row 1) is unable to distinguish different nodes well. The Degree Feature baseline alleviates these issues and performs better than Rows 1 and 4 in this task. However, they do not perform well in general.

We analyze the performance of an equivariant GNN that satisfies Eqn (2). This can be achieved by Random Labeling for training (Rows 8 and 9), by explicitly introducing averaging ensembles during inference (Rows 5 and 8) as the naïve attempt introduced in Section 2.3, or by using the same embedding (Row 1) or the degree embedding (Rows 2 and 3). Their performance, although better than Static Labeling (Row 4), appears inadequate.

We then evaluate the effect of Preferential Labeling in the inference stage, applied to different baseline models. This relaxes (3) but satisfies generalized equivariance (7) during inference. We see the error rates (Rows 6 and 9) are considerably lower than the GNN as an equivariant function.

Moreover, Preferential Labeling explicitly reduces labeling artefacts during training. With the inference algorithm controlled, our approach largely outperforms training with Static and Random Labelings (Row 10 vs. Rows 6 and 9).

We analyze how the number of random labelings affects model performance during inference, shown in Figure 4. We observe that all models achieve higher performance with more labelings. However, the improvement for Random Labeling is marginal, as it suffers from the limitation of an equivariant function in a fundamental way, regardless of the

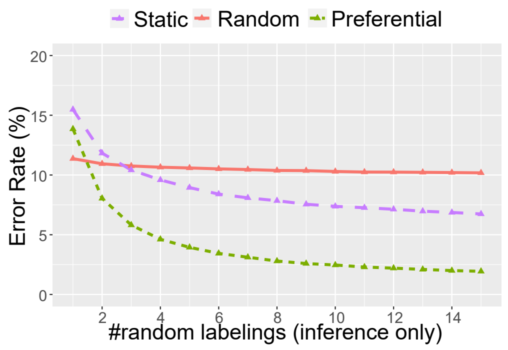

Figure 4: Error rate versus the number of random labelings during inference. We compare the embedding strategies for training, and all variants use the labeling with the maximum predicted probability for inference.

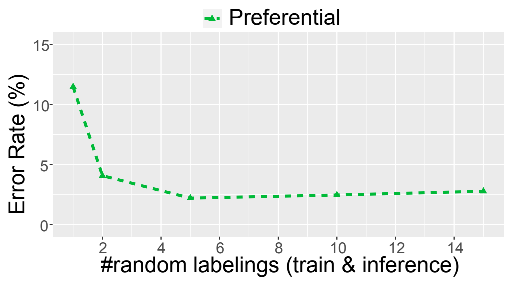

Figure 5: Error rate versus the number of random labelings during both training and inference.

number of labelings.

Static Labeling and Preferential Labeling do not achieve good performance if the number of labelings is small (e.g., ≤ 2). A plausible explanation is that the few labelings during inference may not comply with the training, resulting in high variance. However, the performance improves largely when we have more labelings, as these models are able to relax the function equivariance but achieve generalized equivariance asymptotically. Specifically, our proposed Preferential Labeling is consistently better than Static Labeling by a large margin, as our model is explicitly trained with the best permutation in a hard EM fashion.

Finally, we analyze in Figure 5 how the number of both training and inference labelings affect the performance of Preferential Labeling. In this figure, we use the average error rate (the results on other settings are available at the link in Footnote 1). Results show that if the number of labeling is one, it reduces to random labeling, yielding poor performance. However, the error rate drops significantly when the number increases, and becomes stable when the number is great than or equal to 5. This shows that our approach could still be applied when computational resources are restricted.

4 Related Work

Graph neural networks (GNNs) have been widely researched in recent years (Scarselli et al. 2009; Battaglia et al. 2018). GNNs have a variety of applications in different domains, ranging from social networks (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017), knowledge graphs (Hamaguchi et al. 2017), and programming source code (Mou et al. 2016; Mou and Jin 2018).

A common GNN architecture is the graph convolutional network (GCN, Kipf and Welling, 2017). Recently, researchers have designed various GNN architectures, such as gated graph neural network (Li et al. 2016), graph attention networks (Velickovic et al. 2018), and Transformer-based GNN (Cai and Lam 2020). However, the focus of this paper is not the architecture design. Rather, we focus on node representations in unattributed graphs.

To represent a node in graphs, DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) learns the node presentation by predicting the neighbors in an unsupervised way. Such pretraining-style node embedding does not generalize to new graphs. In certain applications, researchers use domain-specific information as labels. For example, a node in a knowledge graph is represented by text (Lin, Liu, and Sun 2016) and a variable in code analysis/generation tasks is often denoted by its name or subtokens (Allamanis, Brockschmidt, and Khademi 2018; Sun et al. 2020; Xiong and Wang 2021).

The embedding of nodes in unattributed graphs is not extensively addressed in previous literature. Existing work generally applies either an arbitrary labeling (Allamanis, Brockschmidt, and Khademi 2018) or the same embedding (Li, Chen, and Koltun 2018; Selsam et al. 2019; Zhang et al. 2020), which suffer from several limitations as discussed in this paper. To address this, we propose Preferential Labeling for unattributed node classification tasks.

Our Preferential Labeling is also related to, but different from Xu et al. (2019) and Garg, Jegelka, and Jaakkola (2020), where they show that GNNs are limited in determining graph isomorphism. We instead showed equivariant GNNs are limited in solving one-to-many equivariant problems, and further proposed the desired generalized equivariant property. Moreover, our Preferential Labeling does not suffer from the above limitation, because it assumes nodes are unlabeled, but we have (preferential) labelings.

5 Conclusion

In this paper, we address the task of node classification of unattributed graphs. We analyze the limitations of existing GNNs, showing that an equivariant GNN may not solve an equivariant node classification task, when multiple outputs are correct. We propose a generalized equivariance property, which allows an additional auto-isomorphic permutation. Based on our analysis, we further propose Preferential Labeling that samples multiple permutations and uses the best one for training and inference; theoretical analysis shows that our Preferential Labeling achieves the desired generalized equivariance property asymptotically. We conducted extensive experiments on MIS solving and SAT solving tasks to demonstrate the effectiveness and generality of our approach.

Acknowledgments

The work is supported in part by the National Key Research and Development Program of China under Grant No. 2019YFE0198100, the Innovation and Technology Commission of HKSAR under Grant No. MHP/055/19, National Natural Science Foundation of China under Grant No. 61922003, and the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant No. RGPIN2020-04465. Lili Mou is supported in part by the Amii Fellow Program, the Canada CIFAR AI Chair Program, a UAHJIC project, and a donation from DeepMind. This research is also supported by Compute Canada (www.computecanada.ca).

References

Allamanis, M.; Brockschmidt, M.; and Khademi, M. 2018. Learning to represent programs with graphs. In *International Conference on Learning Representations*.

Azizian, W.; et al. 2020. Expressive power of invariant and equivariant graph neural networks. In *International Conference on Learning Representations*.

Backstrom, L.; Dwork, C.; and Kleinberg, J. M. 2007. Wherefore art thou r3579x?: Anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th International Conference on World Wide Web*, 181–190.

Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint*.

Boldi, P.; Bonchi, F.; Gionis, A.; and Tassa, T. 2012. Injecting uncertainty in graphs for identity obfuscation. *Proceedings of the VLDB Endowment*, 1376–1387.

Cai, D.; and Lam, W. 2020. Graph transformer for graph-to-sequence learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 7464–7471.

Chen, Z.; Li, L.; and Bruna, J. 2018. Supervised community detection with line graph neural networks. In *International Conference on Learning Representations*.

Garg, V.; Jegelka, S.; and Jaakkola, T. 2020. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, 3419–3430.

Hamaguchi, T.; Oiwa, H.; Shimbo, M.; and Matsumoto, Y. 2017. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 1802–1808.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1024–1034.

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. S. 2016. Gated graph sequence neural networks. In *International Conference on Learning Representations*.

Li, Z.; Chen, Q.; and Koltun, V. 2018. Combinatorial optimization with graph convolutional networks and guided tree search. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 537–546.

Lin, Y.; Liu, Z.; and Sun, M. 2016. Knowledge representation learning with entities, attributes and relations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2866–2872.

Mou, L.; and Jin, Z. 2018. *Tree-Based Convolutional Neural Networks: Principles and Applications*. Springer.

Mou, L.; Li, G.; Zhang, L.; Wang, T.; and Jin, Z. 2016. Convolutional neural networks over tree structures for programming language processing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 1287–1293.

Murphy, R.; Srinivasan, B.; Rao, V.; and Ribeiro, B. 2019. Relational pooling for graph representations. In *International Conference on Machine Learning*, 4663–4673.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge discovery and data mining*, 701–710.

Samdani, R.; Chang, M.-W.; and Roth, D. 2012. Unified expectation maximization. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 688–698.

Sato, R.; Yamada, M.; and Kashima, H. 2021. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining*, 333–341.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1): 61–80.

Selsam, D.; Lamm, M.; Bünz, B.; Liang, P.; de Moura, L.; and Dill, D. L. 2019. Learning a SAT solver from single-Bit supervision. In *International Conference on Learning Representations*.

Sun, Z.; Zhu, Q.; Xiong, Y.; Sun, Y.; Mou, L.; and Zhang, L. 2020. TreeGen: A tree-based transformer architecture for code generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 8984–8991.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Wang, X.; Ye, Y.; and Gupta, A. 2018. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6857–6866.

Wei, J.; Goyal, M.; Durrett, G.; and Dillig, I. 2020. LambdaNet: Probabilistic type inference using graph neural networks. In *International Conference on Learning Representations*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2021. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1): 4–24.

Xiong, Y.; and Wang, B. 2021. L2S: A framework for synthesizing the most probable program under a specification. *TOSEM: ACM Transactions on Software Engineering and Methodology*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How powerful are graph neural networks? In *International Conference on Learning Representations*.

Zhang, W.; Sun, Z.; Zhu, Q.; Li, G.; Cai, S.; Xiong, Y.; and Zhang, L. 2020. NLocalSAT: Boosting local search with solution prediction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 1177–1183.