Node-wise Diffusion for Scalable Graph Learning

Keke Huang
kkhuang@nus.edu.sg
National University of Singapore

Jing Tang
jingtang@ust.hk
The Hong Kong University of Science and Technology (Guangzhou)
The Hong Kong Uni. of Sci. and Tech.

Juncheng Liu
juncheng.liu@u.nus.edu
National University of Singapore

Renchi Yang
renchi@hkbu.edu.hk
Hong Kong Baptist University

Xiaokui Xiao
xkxiao@nus.edu.sg
National University of Singapore
CNRS@CREATE, Singapore

ABSTRACT

Graph Neural Networks (GNNs) have shown superior performance for semi-supervised learning of numerous web applications, such as classification on web services and pages, analysis of online social networks, and recommendation in e-commerce. The state of the art derives representations for *all* nodes in graphs following the *same* diffusion (message passing) model without discriminating their uniqueness. However, (i) labeled nodes involved in model training usually account for a small portion of graphs in the semi-supervised setting, and (ii) different nodes locate at different graph local contexts and it inevitably degrades the representation qualities if treating them undistinguishedly in diffusion.

To address the above issues, we develop NDM, a universal node-wise diffusion model, to capture the unique characteristics of each node in diffusion, by which NDM is able to yield high-quality node representations. In what follows, we customize NDM for semi-supervised learning and design the NIGCN model. In particular, NIGCN advances the efficiency significantly since it (i) produces representations for labeled nodes only and (ii) adopts well-designed neighbor sampling techniques tailored for node representation generation. Extensive experimental results on various types of web datasets, including citation, social and co-purchasing graphs, not only verify the state-of-the-art effectiveness of NIGCN but also strongly support the remarkable scalability of NIGCN. In particular, NIGCN completes representation generation and training within 10 seconds on the dataset with hundreds of millions of nodes and billions of edges, up to orders of magnitude speedups over the baselines, while achieving the highest F1-scores on classification.

CCS CONCEPTS

• **Computing methodologies** → **Semi-supervised learning**; **Neural networks**.

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04.
https://doi.org/10.1145/3543507.3583408

KEYWORDS

graph neural networks, scalability, semi-supervised classification

ACM Reference Format:
Keke Huang, Jing Tang, Juncheng Liu, Renchi Yang, and Xiaokui Xiao. 2023. Node-wise Diffusion for Scalable Graph Learning. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583408

1 INTRODUCTION

In recent years, Graph Neural Networks (GNNs) have gained increasing attention in both academia and industry due to their superior performance on numerous web applications, such as classification on web services and pages [15, 45], image search [1], web spam detection [2], e-commerce recommendations [13, 39, 42], and social analysis [24, 30, 31]. Various GNN models have been developed [3, 4, 8, 14, 22, 23, 35, 41, 46, 47] accordingly. Among them, *semi-supervised classification* is one of the most extensively studied problems due to the scarce labeled data in real-world applications [12, 20, 34].

Graph Convolutional Network (GCN) [21] is the seminal GNN model proposed for semi-supervised classification. GCN conducts *feature propagation* and *transformation* recursively on graphs and is trained in a full-batch manner, thus suffering from severe scalability issues [4, 5, 16, 36, 38, 44, 47]. Since then, there has been a large body of research on improving the efficiency. One line of work focuses on utilizing sampling and preprocessing techniques. Specifically, GraphSAGE [16] and FastGCN [4] sample a fixed number of neighbors for each layer. GraphSAINT [44] and ShaDow-GCN [43] randomly extract subgraphs with limited sizes as training graphs. Cluster-GCN [7] partitions graphs into different clusters and then randomly chooses a certain number of clusters as training graphs. Another line of research decouples feature propagation and transformation to ease feature aggregations. In particular, SGC [38] proposes to remove non-linearity in transformation and multiplies the feature matrix to the K-th power of the normalized adjacency matrix for feature aggregation. Subsequently, a plethora of decoupled models are developed to optimize the efficiency of feature aggregation by leveraging various graph techniques, including APPNP [22], GBP [6], AGP [36], and GRAND+ [14].

Despite the efficiency advances, current models either calculate node presentations for enormous unlabeled nodes or ignore the

unique topological structure of each labeled node during representation generation. Therefore, there is still room for improvement in efficiency and effectiveness. To explain, labeled nodes involved in model training in semi-supervised learning usually take up a small portion of graphs, especially on massive graphs, and computing representations for all nodes in graphs is unnecessarily inefficient. Meanwhile, different nodes reside in different graph locations with distinctive neighborhood contexts. Generating node representations without considering their topological uniqueness inevitably degrades the representation qualities.

To remedy the above deficiencies, we first develop a node-wise diffusion model NDM. Specifically, NDM calculates an individual diffusion length for each node by taking advantage of the unique topological characteristic for high-quality node representations. In the meantime, NDM employs a universal diffusion function GHD adaptive to various graphs. In particular, GHD is a *general heat diffusion* function that is capable of capturing different diffusion patterns on graphs with various densities. By taking NDM as the diffusion model for feature propagations, we design NIGCN (**N**ode-w**I**se GCN), a GCN model with superb scalability. In particular, NIGCN only computes representations for the labeled nodes for model training without calculating (hidden) representations for any other nodes. In addition, NIGCN adopts customized neighbor sampling techniques during diffusion. By eliminating those unimportant neighbors with noise features, our neighbor sampling techniques not only improve the performance of NIGCN for semi-supervised classification but also boost the efficiency significantly.

We evaluate NIGCN on 7 real-world datasets and compare with 13 baselines for transductive learning and 7 competitors for inductive learning. Experimental results not only verify the superior performance of NIGCN for semi-supervised classification but also prove the remarkable scalability of NIGCN. In particular, NIGCN completes feature aggregations and training within 10 seconds on the dataset with hundreds of millions of nodes and billions of edges, up to orders of magnitude speedups over the baselines, while achieving the highest F1-scores on classification.

In a nutshell, our contributions are summarized as follows.

- We propose a node-wise diffusion model NDM. NDM customizes each node with a unique diffusion scheme by utilizing the topological characteristics and provides a general heat diffusion function capable of capturing different diffusion patterns on graphs with various densities.
- We design a scalable GNN model NIGCN upon NDM. NIGCN calculates node representation for a small portion of labeled nodes without producing intermediate (hidden) representations for any other nodes. Meanwhile, neighbor sampling techniques adopted by NIGCN further boost its scalability significantly.
- We conduct comprehensive experiments to verify the state-of-the-art performance of NIGCN for semi-supervised classification and the remarkable scalability of NIGCN.

2 RELATED WORK

Kipf and Welling [21] propose the seminal Graph Convolutional Network (GCN) for semi-supervised classification. However, GCN suffers from severe scalability issues since it executes the feature propagation and transformation recursively and is trained in a full-batch manner. To alleviate the pain, two directions, i.e., decoupled models and sampling-based models, have been explored.

Decoupled Models. SGC proposed by Wu et al. [38] adopts the decoupling scheme by removing non-linearity in feature transformation and propagates features of neighbors within K hops directly, where K is an input parameter. Following SGC, a plethora of decoupled models have been developed. To consider node proximity, APPNP [22] utilizes personalized PageRank (PPR) [29, 32] as the diffusion model and takes PPR values of neighbors as aggregation weights. To improve the scalability, PPRGo [3] reduces the number of neighbors in aggregation by selecting neighbors with top-K PPR values after sorting them. Graph diffusion convolution (GDC) [23] considers various diffusion models, including both PPR and heat kernel PageRank (HKPR) to capture diverse node relationships. Later, Chen et al. [6] apply generalized PageRank model [25] and propose GBP that combines reverse push and random walk techniques to approximate feature propagation. Wang et al. [36] point out that GBP consumes a large amount of memory to store intermediate random walk matrices and propose AGP that devises a unified graph propagation model and employs forward push and random sampling to select subsets of unimportant neighborhoods so as to accelerate feature propagation. Zhang et al. [46] consider the number of neighbor hops before the aggregated feature gets smoothing. To this end, they design NDLS and calculate an individual local-smoothing iteration for each node on feature aggregation. Recently, Feng et al. [14] investigate the graph random neural network (GRAND) model. To improve the scalability, they devise GRAND+ by leveraging a generalized forward push to compute the propagation matrix for feature aggregation. In addition, GRAND+ only incorporates neighbors with top-K values for further scalability improvement.

Sampling-based Models. To avoid the recursive neighborhood over expansion, GraphSAGE [16] simply samples a fixed number of neighbors uniformly for each layer. Instead of uniform sampling, FastGCN [4] proposes importance sampling on neighbor selections to reduce sampling variance. Subsequently, AS-GCN [18] considers the correlations of sampled neighbors from upper layers and develops an adaptive layer-wise sampling method for explicit variance reduction. To guarantee the algorithm convergence, VR-GCN proposed by Chen et al. [5] exploits historical hidden representations as control variates and then reduces sampling variance via the control variate technique. Similar to AS-GCN, LADIES [47] also takes into account the layer constraint and devises a layer-wise, neighbor-dependent, and importance sampling manner, where two graph sampling methods are proposed as a consequence. Cluster-GCN [7] first applies graph cluster algorithms to partition graphs into multiple clusters, and then randomly takes several clusters as training graphs. Similarly, GraphSAINT [44] samples subgraphs as new training graphs, aiming to improve the training efficiency. Huang et al. [19] adopt the graph coarsening method developed by Loukas [27] to reshape the original graph into a smaller graph, aiming to boost the scalability of graph machine learning. Lately, Zeng et al. [43] propose to extract localized subgraphs with bounded scopes and then run a GNN of arbitrary depth on it. This principle of decoupling GNN scope and depth, named as ShaDow, can be applied to existing GNN models.

However, all the aforementioned methods either (i) generate node representations for *all* nodes in the graphs even though labeled nodes in training are scarce or (ii) overlook the topological uniqueness of each node during feature propagation. Ergo, there is still room for improvement in both efficiency and efficacy.

3 NODE-WISE DIFFUSION MODEL

In this section, we reveal the weakness in existing diffusion models and then design NDM, consisting of two core components, i.e., (i) the diffusion matrix and the diffusion length for each node, and (ii) the universal diffusion function generalized to various graphs.

3.1 Notations

For the convenience of expression, we first define the frequently used notations. We use calligraphic fonts, bold uppercase letters, and bold lowercase letters to represent sets (e.g., $\mathcal{N}$), matrices (e.g., $\mathbf{A}$), and vectors (e.g., $\mathbf{x}$), respectively. The i-th row (resp. column) of matrix $\mathbf{A}$ is represented by $\mathbf{A}[i,\cdot]$ (resp. $\mathbf{A}[\cdot,i]$).

Let $\mathbf{G}=(\mathcal{V},\mathcal{E},\mathbf{X})$ be an undirected graph where $\mathcal{V}$ is the node set with $|\mathcal{V}|=n$, $\mathcal{E}$ is the edge set with $|\mathcal{E}|=m$, and $\mathbf{X}\in\mathbb{R}^{n\times f}$ is the feature matrix. Each node $v\in\mathcal{V}$ is associated with a f-dimensional feature vector $\mathbf{x}_v\in\mathbf{X}$. For ease of exposition, node $u\in\mathcal{V}$ also indicates its index. Let $\mathcal{N}_u$ be the *direct* neighbor set and $d_u=|\mathcal{N}_u|$ be the degree of node u. Let $\mathbf{A}\in\mathbb{R}^{n\times n}$ be the adjacency matrix of $\mathbf{G}$, i.e., $\mathbf{A}[u,v]=1$ if $\langle u,v\rangle\in\mathcal{E}$; otherwise $\mathbf{A}[u,v]=0$, and $\mathbf{D}\in\mathbb{R}^{n\times n}$ be the diagonal degree matrix of $\mathbf{G}$, i.e., $\mathbf{D}[u,u]=d_u$. Following the convention [6, 36], we assume that $\mathbf{G}$ is a *self-looped* and connected graph.

3.2 Diffusion Matrix and Length

Diffusion Matrix. Numerous variants of Laplacian matrix are widely adopted as diffusion matrix in existing GNN models [6, 21, 22, 26, 38, 46]. Among them, the transition matrix $\mathbf{P}=\mathbf{D}^{-1}\mathbf{A}$ is intuitive and easy-explained. Let $1=\lambda_1\geq\lambda_2\geq\ldots\geq\lambda_n>-1$ be the eigenvalues of $\mathbf{P}$. During an infinite diffusion, any initial state $\pi_0\in\mathbb{R}^n$ of node set $\mathcal{V}$ converges to the stable state π, i.e., $\pi=\lim_{\ell\to\infty}\pi_0\mathbf{P}^\ell$ where $\pi(v)=\frac{d_v}{2m}$.

Diffusion Length. As stated, different nodes reside at different local contexts in the graphs, and the corresponding receptive fields for information aggregation differ. Therefore, it is rational that each node u owns a unique length ℓ_u of diffusion steps. As desired, node u aggregates informative signals from neighbors within the range of ℓ_u hops while obtaining limited marginal information out of the range due to over-smoothing issues. To better quantify the effective vicinity, we first define τ-distance as follows.

Definition 3.1 (τ-Distance). Given a positive constant τ and a graph $\mathbf{G}=(\mathcal{V},\mathcal{E})$ with diffusion matrix $\mathbf{P}$, a length ℓ is called τ-distance of node $u\in\mathcal{V}$ if it satisfies that for every $v\in\mathcal{V}$, $\frac{|\mathbf{P}^\ell[u,v]-\pi(v)|}{\pi(v)}\leq\tau$.

According to Definition 3.1, ℓ_u being τ-distance of u ensures that informative signals from neighbors are aggregated. On the other hand, to avoid over-smoothing, ℓ_u should not be too large. In the following, we provide an appropriate setting of ℓ_u fitting both criteria.

Theorem 3.2. *Given a positive constant τ and a graph $\mathbf{G}=(\mathcal{V},\mathcal{E})$ with diffusion matrix $\mathbf{P}$, $\ell_u:=\left\lceil\log_\lambda\frac{\tau\sqrt{d_{\min}d_u}}{2m}\right\rceil$ is τ-distance of node u, where $\lambda=\max\{\lambda_2,-\lambda_n\}$ and $d_{\min}=\min\{d_v\colon v\in\mathcal{V}\}$.*

Proof of Theorem 3.2. Let $\mathbf{e}_u\in\mathbb{R}^{1\times n}$ be a one-hot vector having 1 in coordinate $u\in\mathcal{V}$ and $\mathbf{1}_n\in\mathbb{R}^{1\times n}$ be the 1-vector of size n. Then, $\mathbf{P}^\ell[u,v]=\mathbf{e}_u\mathbf{P}^\ell\mathbf{e}_v^\top$. Let $\tilde{\mathbf{P}}=\mathbf{D}^{1/2}\mathbf{P}\mathbf{D}^{-1/2}=\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ and $\mathbf{u}_i^\top$ be the corresponding eigenvector of its i-th eigenvalue (sorted in descending order) of $\tilde{\mathbf{P}}$. For $\mathbf{e}_u$ and $\mathbf{e}_v$, we decompose

$$\mathbf{e}_u\mathbf{D}^{-1/2}=\sum_{i=1}^n\alpha_i\mathbf{u}_i,\ \text{and}\ \mathbf{e}_v\mathbf{D}^{1/2}=\sum_{i=1}^n\beta_i\mathbf{u}_i.$$

Note that $\{\mathbf{u}_1^\top,\ldots,\mathbf{u}_n^\top\}$ form the orthonormal basis and $\mathbf{u}_1=\frac{\mathbf{1}_n\mathbf{D}^{1/2}}{\sqrt{2m}}$. Thus, we have $\alpha_1=\mathbf{e}_u\mathbf{D}^{-1/2}\mathbf{u}_1^\top=\frac{1}{\sqrt{2m}}$ and $\beta_1=\mathbf{e}_v\mathbf{D}^{1/2}\mathbf{u}_1^\top=\frac{d_v}{\sqrt{2m}}$. Since $\tilde{\mathbf{P}}$ is the similar matrix of $\mathbf{P}$, they share the same eigenvalues. Therefore, we have

$$\begin{aligned}&\frac{\left|\mathbf{P}^\ell[u,v]-\pi(v)\right|}{\pi(v)}=\frac{\left|\mathbf{e}_u\mathbf{P}^\ell\mathbf{e}_v^\top-\pi(v)\right|}{\pi(v)}=\frac{\left|\mathbf{e}_u\mathbf{D}^{-1/2}\tilde{\mathbf{P}}^\ell\mathbf{D}^{1/2}\mathbf{e}_v^\top-\pi(v)\right|}{\pi(v)}\\&=\frac{\left|\sum_{i=1}^n\alpha_i\beta_i\lambda_i^\ell-\pi(v)\right|}{\pi(v)}=\frac{\left|\sum_{i=2}^n\alpha_i\beta_i\lambda_i^\ell\right|}{\pi(v)}\leq\lambda^\ell\cdot\frac{\sum_{i=2}^n|\alpha_i\beta_i|}{\pi(v)}\\&\leq\lambda^\ell\cdot\frac{\|\mathbf{e}_u\mathbf{D}^{-1/2}\|\|\mathbf{e}_v\mathbf{D}^{1/2}\|}{d_v/2m}=\frac{2m\lambda^\ell}{\sqrt{d_vd_u}},\end{aligned}$$

where the second inequality is by Cauchy–Schwarz inequality. Finally, setting $\ell:=\left\lceil\log_\lambda\frac{\tau\sqrt{d_{\min}d_u}}{2m}\right\rceil$ completes the proof. □

For the ℓ_u defined in Theorem 3.2, it is τ-distance of node u and in the meantime involves the topological uniqueness of node u. Moreover, the performance can be further improved by tuning the hyperparameter τ.

3.3 Universal Diffusion Function

As we know, the diffusion model defined by the symmetrically normalized Laplacian matrix $\mathbf{L}=\mathbf{I}-\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is derived from *Graph Heat Equation* [9, 37], i.e.,

$$\frac{\mathrm{d}\mathbf{H}_t}{\mathrm{d}t}=-\mathbf{L}\mathbf{H}_t,\ \text{and}\ \mathbf{H}_0=\mathbf{X},\tag{1}$$

where $\mathbf{H}_t$ is the node status of graph $\mathbf{G}$ at time t. By solving the above differential function, we have

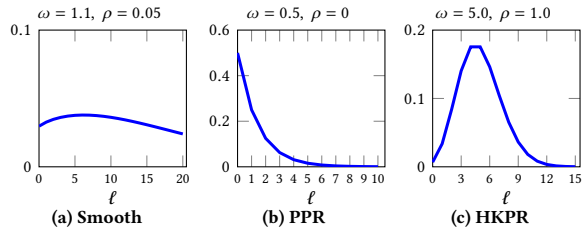
$$\mathbf{H}_t=\mathrm{e}^{-t\mathbf{L}}=\mathrm{e}^{-t(\mathbf{I}-\tilde{\mathbf{A}})}=\mathrm{e}^{-t}\sum_{\ell=0}^\infty\frac{t^\ell\tilde{\mathbf{A}}^\ell}{\ell!},\tag{2}$$

where $\tilde{\mathbf{A}}=\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. In this regard, the underlying diffusion follows the *Heat Kernel PageRank* (HKPR) function as

$$f(\omega,\ell)=\mathrm{e}^{-\omega}\frac{\omega^\ell}{\ell!},\tag{3}$$

where $\omega\in\mathbb{Z}^+$ is the parameter. However, $f(\omega,\ell)$ is neither expressive nor general enough to act as the universal diffusion function for real-world graphs, hinted by the following graph property.

Property 3.1 ([9]). *For graph $\mathbf{G}$ with average degree $d_{\mathbf{G}}$, we have $1-\Delta_\lambda=O(\frac{1}{\sqrt{d_{\mathbf{G}}}})$ where Δ_λ is the spectral gap of $\mathbf{G}$.*



(a) Smooth (b) PPR (c) HKPR

Figure 1: Three exemplary expansion tendency of GHD.

For the diffusion matrix $\mathbf{P}$ defined on $\mathbf{G}$, we have $\lambda = 1 - \Delta_\lambda$. Meanwhile, according to the analysis of Theorem 3.2, we know that $\mathbf{P}^\ell[u,v] - \pi(v) = \sum_{i=2}^{n} \alpha_i \beta_i \lambda_i^\ell$, representing the convergence, is (usually) dominated by λ^ℓ. As a result, diffusion on graphs with different densities, i.e., $d_\mathbf{G}$, converges at different paces. In particular, sparse graphs with small $d_\mathbf{G}$ incurring large λ tend to incorporate neighbors in a long range while dense graphs with large $d_\mathbf{G}$ incurring small λ are prone to aggregate neighbors not far away. In addition, it has been widely reported in the literature [14, 23] that different graphs ask for different diffusion functions, which is also verified by our experiments in Section 5.2.

To serve the universal purpose, a qualified diffusion function should be able to (i) expand smoothly in long ranges, (ii) decrease sharply in short intervals, and (iii) peak at specified hops, as required by various graphs accordingly. Clearly, the HKPR function in (3) fulfills the latter two requirements but fails the first one since it decreases exponentially when $\ell \geq \omega$. One may propose to consider *Personalized PageRank* (PPR). However, the PPR function is monotonically decreasing and thus cannot reach condition (iii).

Inspired by the above analysis, we try to ameliorate $f(\omega, \ell)$ to a universal diffusion function with a controllable change tendency for general purposes. To this end, we extend the graph heat diffusion Equation (3) by introducing an extra *power parameter* $\rho \in \mathbb{R}^+$ and devise our *General Heat Diffusion* (GHD) function as

$$U(\omega, \rho, \ell) = \frac{\omega^\ell}{(\ell!)^\rho \cdot C} \tag{4}$$

for the diffusion weight at the ℓ-th hop, where $\omega \in \mathbb{R}^+$ is the new heat parameter and $C = \sum_{\ell=0}^{\infty} \frac{\omega^\ell}{(\ell!)^\rho}$ is the normalization factor.

As desired, GHD can be regarded as a general extension of the graph heat diffusion model, and parameters ω and ρ together determine the expansion tendency. In particular, it is trivial to verify that GHD is degraded into HKPR when $\rho = 1$, and GHD becomes PPR when $\rho = 0$. As illustrated in Figure 1, by setting different ω and ρ combinations, GHD is able to exhibit smooth, exponential (i.e., PPR), or peak expansion (i.e., HKPR) tendency.

3.4 Diffusion Model Design

Upon τ-distance and diffusion function UDF, our node-wise diffusion model (NDM) can be concreted. Specifically, given a target set $\mathcal{T} \subseteq \mathcal{V}$, the representation $\mathbf{Z}_\mathcal{T}$ under NDM is calculated as

$$\mathbf{Z}_\mathcal{T} = \sum_{\ell=0}^{L} \mathbf{U}\Gamma\mathbf{P}^\ell\mathbf{X}, \tag{5}$$

Algorithm 1: Node-wise Diffusion Model

Input: Graph $\mathbf{G}$, feature matrix $\mathbf{X}$, target set $\mathcal{T}$, and hyperparameters τ, ω, ρ
Output: Representation $\mathbf{Z}_\mathcal{T}$

$d_{\max} \leftarrow \max_{u \in \mathcal{T}}\{d_u\}$;
$L \leftarrow \left\lceil \log_\lambda \frac{\tau\sqrt{d_{\min} d_{\max}}}{2m} \right\rceil$;
U is calculated according to (4);
$\Gamma \leftarrow \mathbf{I}[\mathcal{T}, \cdot]$, $\mathbf{Z}_\mathcal{T} \leftarrow \mathbf{0}^{|\mathcal{T}| \times f}$;
for $\ell \leftarrow 0$ **to** L **do**
  $\mathbf{U} \leftarrow \mathrm{Diag}\{U(\omega, \rho, \ell) : \forall u \in \mathcal{T}\}$;
  $\mathbf{Z}_\mathcal{T} \leftarrow \mathbf{Z}_\mathcal{T} + \mathbf{U}\Gamma$;
  $\Gamma \leftarrow \Gamma\mathbf{P}$, $\ell \leftarrow \ell + 1$;
$\mathbf{Z}_\mathcal{T} \leftarrow \mathbf{Z}_\mathcal{T}\mathbf{X}$;
return $\mathbf{Z}_\mathcal{T}$;

where $L = \max\{\ell_u : \forall u \in \mathcal{T}\}$, $\mathbf{U} = \mathrm{Diag}\{U(\omega, \rho, \ell) : \forall u \in \mathcal{T}\} \in \mathbb{R}^{|\mathcal{T}| \times |\mathcal{T}|}$ is a diagonal matrix, and $\Gamma = \mathbf{I}[\mathcal{T}, \cdot] \in \mathbb{R}^{|\mathcal{T}| \times n}$ is the indicator matrix, i.e., $\Gamma[i,u] = 1$ if $\mathcal{T}[i] = u$ and $\Gamma[i,u] = 0$ otherwise.

The pseudo-code of NDM is presented in Algorithm 1. NDM first finds the largest degree $d_{\max}$ for nodes $\mathcal{T}$, and computes the corresponding τ-distance as L. Then, NDM accumulates the weights of neighbors within L ranges for each node $u \in \mathcal{T}$, recorded as $\mathbf{Z}_\mathcal{T}$. Note that $\mathbf{U}[u,u] = 0$ if $\ell > L$. Finally, representation $\mathbf{Z}_\mathcal{T}$ is calculated by multiplying the feature matrix $\mathbf{X}$.

Time Complexity. It takes $O(m)$ time to calculate λ using the iterative methods [11], and hence computing L take $O(m + |\mathcal{T}|)$ time. Matrix multiplications $\mathbf{U}\Gamma$ and $\Gamma\mathbf{P}$ dominate the running time, which takes time complexities of $O(n|\mathcal{T}|)$ and $O(m|\mathcal{T}|)$, respectively. Therefore, as it takes $O(f|\mathcal{T}|)$ time to compute $Z_\mathcal{T}\mathbf{X}$, the total time complexity of NDM is $O((m+n)L|\mathcal{T}| + f|\mathcal{T}|)$.

4 OPTIMIZATION IN NODE REPRESENTATION LEARNING

Algorithm 1 in Section 3 presents a general node-wise diffusion model. However, it is yet optimal to be applied to reality. In this section, we aim to instantiate NDM in a practical manner and optimize the procedure of feature propagations.

4.1 Instantiation of NDM

Practical Implementation of τ-Distance. Calculating the τ-distance of each node is one of the critical steps in NDM, which requires the second largest eigenvalue λ of the diffusion matrix. However, it is computationally expensive to compute λ for large graphs. To circumvent the scenario, we employ property 3.1 to substitute λ without damaging the efficacy of NDM.

As we analyze in Section 3.3, according to Property 3.1, we borrow a correction factor $C_\mathbf{G}$ specific for graph $\mathbf{G}$ to ensure $\lambda = 1 - \Delta_\lambda = \frac{C_\mathbf{G}}{\sqrt{d_\mathbf{G}}}$. Meanwhile, for the sake of practicality, we could merge hyperparameter τ and $C_\mathbf{G}$ into one tunable parameter τ' to control the bound of τ-distance ℓ_u such that

$$\ell_u = \log_\lambda \frac{\tau\sqrt{d_{\min} d_u}}{2m} = \frac{\ln \frac{2m}{\sqrt{d_{\min} d_u}} - \ln \tau}{\ln \sqrt{d_\mathbf{G}} - \ln C_\mathbf{G}} := \tau' \frac{\ln \frac{2m}{\sqrt{d_{\min} d_u}}}{\ln \sqrt{d_\mathbf{G}}}. \tag{6}$$

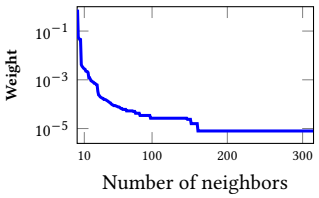


Figure 2: Weight Distribution of Neighbors.

Important Neighbor Identification and Selection. NDM in Algorithm 1 aggregates all neighbors during diffusion for each node, which, however, is neither effective nor efficient. The rationale is twofold.

First, it is trivial to see that the sum of weights in the ℓ-th hop is $\sum_{v\in\mathcal{V}} U(\omega,\rho,\ell)(\mathbf{D}^{-1}\mathbf{A})^{\ell}[u,v] = U(\omega,\rho,\ell)$. If n nodes are visited, the average weight is $\Theta(\frac{U(\omega,\rho,\ell)}{n})$, i.e., the majority of nodes contribute negligibly to feature aggregations and only a small portion of neighbors with large weights matters. Second, as found in [28, 40], input data contain not only the low-frequency ground truth but also noises that can originate from falsely labeled data or features. Consequently, incorporating features of those neighbors could potentially incur harmful noises. Therefore, it is a necessity to select important neighbors and filter out insignificant neighbors.

Based on the above analysis, we aim to identify important neighbors for target node u. For ease of exposition, we first define the weight function $\phi(\ell,u,v) = U(\omega,\rho,\ell)\mathbf{P}^{\ell}[u,v]$ to quantify the importance of neighbor node v to target node u, and then formalize the concept of *ε-importance neighbor* as follows.

Definition 4.1 (ε-Importance Neighbor). Given a target node u and threshold $\varepsilon \in (0,1)$, node v is called ε-importance neighbor of u if $\exists \ell \in \{0,1,\ldots,\ell_u\}$, we have $\phi(\ell,u,v) \geq \varepsilon$.

Thanks to the good characteristic of NDM, a sufficient number of random walks (RWs) are able to identify *all* such ε-importance neighbors with high probability, as proved in the following lemma.

Lemma 4.2. *Given a target node u, threshold $\varepsilon \in (0,1)$, and failure probability $\delta \in (0,1)$, assume $\phi(\ell,u,v) \geq \varepsilon$. Suppose $\theta = \lceil \frac{2\eta^2}{\varepsilon}\log(\frac{1}{\delta\varepsilon})\rceil$ RWs are generated from u and visit v for $\theta_v^{(\ell)}$ times at the ℓ-th step. For the weight estimation $\hat{\phi}(\ell,u,v) = \frac{U(\omega,\rho,\ell)\theta_v^{(\ell)}}{\theta}$, we have*

$$\Pr\left[\bigvee_{0\leq\ell\leq\ell_u}\left\{\bigvee_{\{v:\ \phi(\ell,u,v)\geq\varepsilon\}}\left(\hat{\phi}(\ell,u,v) \leq \frac{\phi(\ell,u,v)}{\eta}\right)\right\}\right] \leq \delta,$$

where $\eta > 1$ controls the approximation.

Lemma 4.2 affirms that sufficient RWs could capture ε-importance neighbors with high probability. However, there still contains deficiencies. In particular, along with those ε-important neighbors, many insignificant neighbors will be inevitably selected. For illustration, we randomly choose one target node on dataset Amazon and select its neighbors using RWs. We observe that 10.6% neighbors contribute to 99% weights, and the rest 89.4% neighbors share the left 1% weights, as shown in Figure 2. The amount of those insignificant neighbors could unavoidably impair the representation quality. To alleviate the deficiency, we propose to preserve the first-K neighbors with $K = \frac{1}{\varepsilon^2}$.[1]

Algorithm 2: NIGCN

Input: Graph $\mathbf{G}$, feature matrix $\mathbf{X}$, target set $\mathcal{T}$, parameters η and δ, hyperparameters τ', ω, ρ, ε
Output: Representation $\mathbf{Z}_{\mathcal{T}}$

1 $d_{\mathrm{G}} \leftarrow \frac{2m}{n}$, $\theta \leftarrow \lceil \frac{2\eta^2}{\varepsilon}\log(\frac{1}{\delta\varepsilon})\rceil$, $\mathbf{Z}_{\mathcal{T}} \leftarrow \mathbf{0}^{|\mathcal{T}|\times f}$;
2 **for** $u \in \mathcal{T}$ **do**
3 $\ell_u \leftarrow \left\lceil \tau' \frac{\ln \frac{2m}{\sqrt{d_{\min} d_u}}}{\ln \sqrt{\Delta_{\mathrm{G}}}} \right\rceil$;
4 **for** $i \leftarrow 1$ **to** θ **do**
5 Generate a random walk from node u with length ℓ_u;
6 **if** *v is visited at the ℓ-th step* **then**
7 $t_v \leftarrow t_v + \frac{U(\omega,\rho,\ell)}{\theta}$;
8 $\mathcal{S} \leftarrow \mathcal{S} \cup \{v\}$;
9 **if** $|\mathcal{S}| \geq \frac{1}{\varepsilon^2}$ **then break**;
10 **for** $v \in \mathcal{S}$ **do** $\mathbf{z}_u \leftarrow \mathbf{z}_u + t_v \cdot \mathbf{x}_v$;
11 **return** $\mathbf{Z}_{\mathcal{T}}$;

To explain, in the ℓ-th hop, each ε-important neighbor will be selected with probability at least $\frac{\varepsilon}{U(\omega,\rho,\ell)}$, and there are at most $\frac{U(\omega,\rho,\ell)}{\varepsilon}$ important neighbors. Thus ε-important neighbors from the ℓ-th hop will be picked after at most $\frac{U^2(\omega,\rho,\ell)}{\varepsilon^2}$ random selections in expectation. By summing up all ℓ_u hops, we have

$$\sum_{\ell=0}^{\ell_u}\frac{U^2(\omega,\rho,\ell)}{\varepsilon^2} \leq \sum_{\ell=0}^{\ell_u}\frac{U(\omega,\rho,\ell)}{\varepsilon^2} = \frac{1}{\varepsilon^2}.$$

Notice that neither RWs selection nor first-K selection is suitable to solely function as stop conditions. As stated, RW inevitably incurs substantial unimportant neighbors, while first-K selection alone is not bound to terminate when no sufficient neighbors exist. Hence, they compensate each other for better performance. As evaluated in Section 5.4, first-K selection further boosts the effectiveness notably.

4.2 Optimized Algorithm NIGCN

We propose NIGCN in Algorithm 2, the GCN model by instantiating NDM. We first initialize the number θ of RWs according to Lemma 4.2. Next, we generate length-ℓ_u RWs for each $u \in \mathcal{T}$. If neighbor v is visited at the ℓ-th step, we increase its weight t_v by $\frac{U(\omega,\rho,\ell)}{\theta}$ and store it into set $\mathcal{S}$. This procedure terminates if either the number of RWs reaches θ or the condition $|\mathcal{S}| \geq \frac{1}{\varepsilon^2}$ is met. Afterward, we update on $\mathbf{z}_u$ (Line 10). Eventually, the final representation $\mathbf{Z}_{\mathcal{T}}$ is returned once all $|\mathcal{T}|$ target nodes have been processed.

Accordingly, for a target node u in $\mathcal{T}$, NIGCN is formulated as

$$\mathbf{z}_u = \sum_{v\in\{|\mathrm{RW}(\mathcal{N}_u^{(0)}\cup\mathcal{N}_u^{(1)}\cup\cdots\cup\mathcal{N}_u^{(\ell_u)})|_{\leq K}\}}\sum_{\ell=0}^{\ell_u}\frac{U(\omega,\rho,\ell)\theta_v^{(\ell)}}{\theta}\cdot\mathbf{x}_v, \quad (7)$$

[1] One may propose to adopt top-K neighbors. However, top-K selection would incur enormous computation overheads since it requires sorting all neighbors by weights.

Table 1: Time complexity (b is the batch size, k is the sample size in one hop, L is the propagation length, and L' is the number of model layers).

GNN Method	Preprocessing	Training
GraphSAGE	-	$O(\lvert\mathcal{T}\rvert k^L f^2)$
GraphSAINT	-	$O(\frac{Lbmf}{n} + Lnf^2)$
FastGCN	-	$O(L\lvert\mathcal{T}\rvert kf + L\lvert\mathcal{T}\rvert f^2)$
ShaDow-GCN	-	$O(\lvert\mathcal{T}\rvert k^L + L'nf^2)$
APPNP	-	$O(Lmf + L'nf^2)$
GBP	$O(\frac{Lnf\sqrt{L\lvert\mathcal{T}\rvert\log(nL)m/n}}{\varepsilon})$	$O(L'\lvert\mathcal{T}\rvert f^2)$
AGP	$O(\frac{L^2nf}{\varepsilon})$	$O(L'\lvert\mathcal{T}\rvert f^2)$
NDLS	$O(Lmf)$	$O(L'\lvert\mathcal{T}\rvert f^2)$
GRAND+	$O(\frac{Ln}{\varepsilon})$	$O(knf + L'knf^2)$
NIGCN	$O(\frac{L\lvert\mathcal{T}\rvert f}{\varepsilon}\log\frac{1}{\delta\varepsilon})$	$O(L'\lvert\mathcal{T}\rvert f^2)$

where $\mathrm{RW}(\mathcal{N}_u^{(0)} \cup \mathcal{N}_u^{(1)} \cup \cdots \cup \mathcal{N}_u^{(\ell_u)})$ is the neighbor set identified by RWs within ℓ_u hops, $|\cdot|_{\le K}$ indicates the first-K neighbors, and $\sum_{\ell=0}^{\ell_u} \frac{U(\omega,\rho,\ell)\theta_v^{(\ell)}}{\theta}$ is the total estimated weights for selected neighbor v.

Time Complexity. For each node u, at most θ RWs of length ℓ_u are generated at the cost of $O(\theta\ell_u)$, and the total number of neighbors is bounded by $O(\theta\ell_u)$, which limits the cost on the feature update to $O(\theta\ell_u f)$. The total cost is $O((f+1)\theta\ell_u)$ for each target node. Let $L = \max\{\ell_u : \forall u \in \mathcal{T}\}$. By replacing $\theta = O(\frac{1}{\varepsilon}\log(\frac{1}{\delta\varepsilon}))$, the resulting time complexity of NIGCN is $O(\frac{L|\mathcal{T}|f}{\varepsilon}\log\frac{1}{\delta\varepsilon})$.

4.3 Time Complexity Comparison

ε-importance neighbors play a crucial role in feature aggregations. We assume that qualified representations incorporate all ε-importance neighbors with high probability. When capturing such ε-importance neighbors, we analyze and compare the sampling complexities of 9 representative models[2] with NIGCN, as summarized in Table 1.

GRAND+ [14] estimates the propagation matrix Π during preprocessing with error bounded by $r_{\max}$ at the cost of $O(\frac{(|\mathbf{U}'|+|\mathcal{T}|)L}{r_{\max}})$ where $\mathbf{U}'$ is a sample set of unlabeled node set $\mathbf{U} = \mathcal{V} \setminus \mathcal{T}$. To yield accurate estimations for ε-importance neighbors, $r_{\max}$ is set $r_{\max} = \Theta(\varepsilon)$ and the resulting time complexity of prepossessing is $O(\frac{Ln}{\varepsilon})$. According to [14], its training complexity is $O(knf + L'knf^2)$.

AGP [36] provides an accurate estimation for node u if $\pi(u) > \epsilon'$ for each dimension of feature matrix $\mathbf{X}$, denoted as $\mathbf{x} = \mathbf{X}[\cdot, i]$ for $i \in \{0, \ldots, f-1\}$ where $\|\mathbf{x}\|_1 \le 1$, $\pi = \sum_{\ell=0}^{L} w^{(\ell)}(\mathbf{D}^{-a}\mathbf{A}\mathbf{D}^{-b})^\ell\mathbf{x}$, and ϵ' is an input threshold. As proved in [36], the time cost of AGP on one feature dimension is $O(\frac{L^2}{\epsilon'}\sum_{\ell=0}^{L}\|(\sum_{i=\ell}^{\infty} w^{(i)})(\mathbf{D}^{-a}\mathbf{A}\mathbf{D}^{-b})^\ell\mathbf{x}\|_1)$. We consider a simple case that $a = 0$ and $b = 1$. Suppose that $\|\mathbf{x}\|_1 = \frac{1}{C}$ for some constant $C \ge 1$, and thus we have

$$\left\|\Big(\sum_{i=\ell}^{\infty} w^{(i)}\Big)(\mathbf{A}\mathbf{D}^{-1})^\ell\mathbf{x}\right\|_1 = \sum_{i=\ell}^{\infty} w^{(i)}\left\|(\mathbf{A}\mathbf{D}^{-1})^\ell\mathbf{x}\right\|_1 = \frac{1}{C}\sum_{i=\ell}^{\infty} w^{(i)}.$$

[2]We assume the models without explicit sampling process with sampling rate 100%.

Since $\sum_{\ell=0}^{L}\sum_{i=\ell}^{\infty} w^{(i)} \ge 1$, the time complexity of AGP is at least $\Theta(\frac{L^2}{C\epsilon'})$. To ensure nodes aggregating at least one ε-importance neighbor v are estimated accurately, $\varepsilon\mathbf{x}(v) = \Omega(\epsilon')$ is required. Since $\|\mathbf{x}\|_1 = \frac{1}{C}$ for some constant C and there are n nodes, it is reasonably to assume that $\mathbf{x}(v) = O(\frac{1}{n})$. Therefore, $\epsilon' = O(\frac{\varepsilon}{n})$. In this regard, the time cost of AGP to capture ε-importance neighbors for all f dimensions is $O(\frac{L^2nf}{\varepsilon})$.

GBP [6] derives representations for the i-th dimension as $\pi = \sum_{\ell=0}^{L} w^{(\ell)}\mathbf{D}^{\gamma}(\mathbf{D}^{-1}\mathbf{A})^\ell\mathbf{D}^{-\gamma}\cdot\mathbf{x}$, where $\mathbf{x} = \mathbf{X}[\cdot, i]$ for $i \in \{0, \ldots, f-1\}$ and $\mathbf{X}$ is the feature matrix with f dimensions. GBP ensures that the estimation $\hat{\pi}(u)$ of $\pi(u)$ for any $u \in \mathcal{T}$ is within an error of $d_u^{\gamma}\epsilon'$, i.e., $|\pi(u) - \hat{\pi}(u)| \le d_u^{\gamma}\epsilon'$, where the factor d_u^{γ} is due to the term $\mathbf{D}^{\gamma}$ and does not impact sampling errors. The final time complexity of GBP is $\frac{Lf\sqrt{L|\mathcal{T}|\log(nL)m/n}}{\epsilon'}$. As discussed above, we have $\varepsilon\mathbf{x}(u) = \Omega(\epsilon')$ and $\mathbf{x}(u) = O(\frac{1}{n})$, which indicates that $\epsilon' = O(\frac{\varepsilon}{n})$. Consequently, the time cost of GBP to capture ε-importance neighbors is $O(\frac{Lnf\sqrt{L|\mathcal{T}|\log(nL)m/n}}{\varepsilon})$.

For the rest models in Table 1, we borrow the time complexity from their official analyses since they either provide no sampling approximation guarantee or consider all neighbors without explicit sampling. As analyzed, time complexities of state of the art are linear in the size of the graph, while that of NIGCN is linear in the size of the target set $\mathcal{T}$. In semi-supervised classification with limited labels, we have $|\mathcal{T}| \ll n$, which confirms the theoretical efficiency superiority of NIGCN.

Parallelism. NIGCN derives the representation of every target node *independently* and does not rely on any intermediate representations of other nodes. This design makes NIGCN inherently parallelizable so as to be a promising solution to derive node representations for massive graphs since they can process all nodes simultaneously. Further, this enables NIGCN scalable for supervised learning as well.

5 EXPERIMENT

In this section, we evaluate the performance of NIGCN for semi-supervised classification in terms of effectiveness (micro F1-scores) and efficiency (running times).

5.1 Experimental Setting

Datasets. We use seven publicly available datasets across various sizes in our experiments. Specifically, we conduct *transductive learning* on the four citation networks, including three small citation networks [33] Cora, Citeseer, and Pubmed, and a web-scale citation network Papers100M [17]. We run *inductive learning* on three large datasets, i.e., citation network Ogbn-arxiv [17], social network Reddit [44], and co-purchasing network Amazon [44]. Table 6 in Appendix A.2 summarizes the statistics of those datasets. Among them, Papers100M is the largest dataset ever tested in the literature.

For semi-supervised classification with limited labels, we randomly sample 20 nodes per class for training, 500 nodes for validation, and 1000 nodes for testing. For each dataset, we randomly generate 10 instances and report the average performance of each tested method.

Baselines. For transductive learning, we evaluate NIGCN against 13 baselines. We categorize them into three types, i.e., (i) 2 *coupled*

Table 2: F1-score (%) of transductive learning.

	Methods	Cora	Citeseer	Pubmed	Papers100M
Coup*	GCN	78.91 ± 1.87	69.11 ± 1.46	78.05 ± 1.64	OOM
	GAT	80.22 ± 1.48	69.35 ± 0.93	78.82 ± 1.90	OOM
Sampling	GraphSAGE	76.67 ± 2.21	67.41 ± 1.77	76.92 ± 2.84	OOM
	GraphSAINT	74.76 ± 2.86	67.51 ± 4.76	78.65 ± 4.17	OOM
	ShaDow-GCN	73.83 ± 2.46	63.54 ± 1.11	71.79 ± 2.92	OOM
Decoupled	SGC	76.79 ± 1.82	70.49 ± 1.29	74.11 ± 2.55	48.59 ± 1.77
	APPNP	81.16 ± 0.77	69.83 ± 1.27	80.21 ± 1.79	OOM
	PPRGo	79.01 ± 1.88	68.92 ± 1.72	78.20 ± 1.96	OOM
	GDC	80.69 ± 1.99	69.69 ± 1.42	77.67 ± 1.65	OOM
	GBP	81.17 ± 1.60	70.18 ± 1.90	80.09 ± 1.51	44.91 ± 1.23
	AGP	77.70 ± 2.04	67.15 ± 2.04	78.97 ± 1.33	46.71 ± 1.99
	NDLS	81.39 ± 1.55	69.63 ± 1.69	80.38 ± 1.41	OOM
	GRAND+	**83.48 ± 1.18**	**71.42 ± 1.89**	79.18 ± 1.93	OOM
	NIGCN	82.13 ± 1.08	71.35 ± 0.82	**80.90 ± 2.02**	**49.81 ± 1.10**

Table 3: F1-score (%) of inductive learning.

Methods	Ogbn-arxiv	Reddit	Amazon
GraphSAGE	51.79 ± 2.16	89.16 ± 1.16	47.71 ± 1.07
FastGCN	56.45 ± 1.69	92.43 ± 1.00	OOM
SGC	56.03 ± 1.96	92.64 ± 1.02	41.32 ± 1.10
PPRGo	52.12 ± 3.22	78.21 ± 3.07	60.10 ± 1.17
GBP	54.01 ± 2.55	76.09 ± 1.75	60.78 ± 1.04
AGP	55.89 ± 1.47	92.18 ± 0.88	55.72 ± 1.68
NDLS	54.23 ± 2.49	85.25 ± 1.24	50.10 ± 2.09
NIGCN	**57.38 ± 1.31**	**93.42 ± 0.48**	**63.45 ± 0.70**

GNN methods GCN [21] and GAT [35], (ii) 3 *sampling-based* methods GraphSAGE [16], GraphSAINT [44], and ShaDow-GCN [43], and (iii) 8 *decoupled* GNN methods SGC [38], APPNP [22], and its improvement PPRGo [3], GDC [23], GBP [6], AGP [36], and two recently proposed NDLS [46], and GRAND+ [14].

For inductive learning, we compare NIGCN with 7 baselines. Among the 13 methods tested in transductive learning, 7 of them are not suitable for semi-supervised inductive learning and thus are omitted, as explained in Section A.2. In addition, we include an extra method FastGCN [4] designed for inductive learning. Details for the implementations are provided in Appendix A.2.

Parameter Settings. For NIGCN, we fix $\eta = 2$, $\delta = 0.01$ and tune the four hyperparameters τ', ω, ρ, and ε. Appendix A.2 provides the principal on how they are tuned and values selected for all datasets. As with baselines, we either adopt their suggested parameter settings or tune the parameters following the same principle as NIGCN to reach their best possible performance.

All methods are evaluated in terms of *micro F1-scores* on node classification and *running times* including preprocessing times (if applicable) and training times. One method is omitted on certain datasets if it (i) is not suitable for inductive semi-supervised learning or (ii) runs out of memory (OOM), either GPU memory or RAM.

5.2 Performance Results

Table 2 and Table 3 present the averaged F1-scores associated with the standard deviations in *transductive learning* on Cora, Citeseer, Pubmed, and Papers100M and *inductive learning* on Ogbn-arxiv, Reddit, and Amazon respectively. For ease of demonstration, we highlight the *largest* score in bold and underline the *second largest* score for each dataset.

Table 2 shows that NIGCN achieves the highest F1-scores on datasets Pubmed and Papers100M and the second highest scores on Cora and Citeseer. Meanwhile, NIGCN obtains the largest F1-scores on the three datasets Ogbn-arxiv, Reddit, and Amazon, as displayed in Table 3. In particular, the improvement margins over the second best on the three datasets are 0.93%, 0.78%, and 2.67% respectively. These observations indicate that NIGCN performs better on relatively large graphs. Intuitively, nodes in large graphs are prone to reside in various structure contexts and contain neighbors of mixed quality, and the NDM diffusion model and the sampling techniques (important neighbor identification and selection) utilized by NIGCN are able to take advantage of such node-wise characteristics.

The most competitive method, GRAND+ achieves the best on datasets Cora and Citeseer. Nonetheless, as shown in Figure 6 (Section A.3), GRAND+ runs significantly slower than NIGCN does. For the three sampling-based methods, i.e., GraphSAGE, GraphSAINT, and ShaDow-GCN, they acquire noticeably lower F1-scores than NIGCN does. This is due to that they sample neighbors and nodes randomly without customizing the sampling strategy towards target nodes, as introduced in Section 2. Meanwhile, the clear performance improvement of NIGCN over GBP and AGP clearly supports the superiority of our general heat diffusion function GHD over the diffusion models used in GBP and AGP (i.e., PPR and HKPR), as well as the efficacy of our diffusion model NDM.

Overall, it is crucial to consider the unique structure characteristic of each individual node in the design of both the diffusion model and neighbor sampling techniques for node classifications.

5.3 Scalability Evaluation on Large Graphs

In this section, we evaluate the scalability of tested methods by comparing their running times on the four large datasets, Ogbn-arxiv, Reddit, Amazon, and Papers100M. In particular, the running times include preprocessing times (if applicable) and training times. For a comprehensive evaluation, we also report the corresponding running times on the three small datasets Cora, Citeseer, and Pubmed in Appendix A.3.

As shown in Figure 3, NIGCN ranks third with negligible lags on dataset Ogbn-arxiv and dominates other methods noticeably on datasets Reddit, Amazon, and Papers100M. Meanwhile, its efficiency advantage expands larger as datasets grow. Specifically, on dataset Ogbn-arxiv, NIGCN, SGC, and AGP all can finish running within 1 second. The speedups of NIGCN against the second best on datasets Reddit, Amazon, and Papers100M are up to 4.12×, 8.90×, and 441.61× respectively. In particular, on the largest dataset Papers100M, NIGCN is able to complete preprocessing and model training within 10 seconds. The remarkable scalability of NIGCN lies in that (i) NIGCN only generates node representations for a small portion of labeled nodes involved in model training, and (ii) neighbor sampling techniques in NIGCN significantly reduce the number of neighbors for each labeled node in feature aggregation, as detailed analyzed in Section 4.1. This observation strongly supports the outstanding scalability of NIGCN and its capability to handle web-scale graphs.

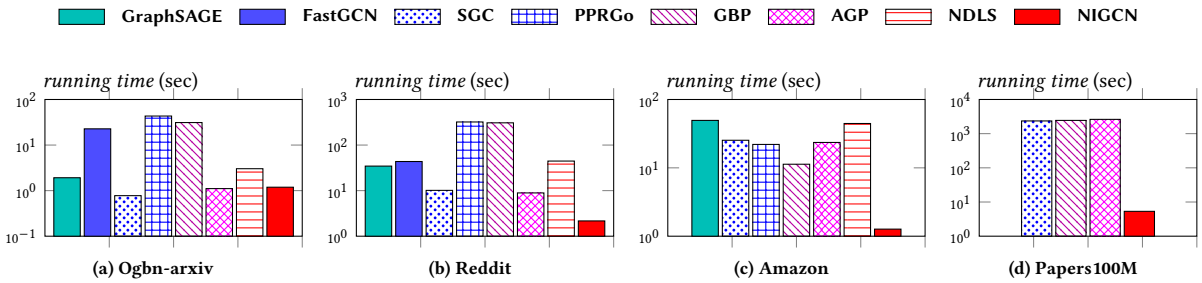


Figure 3: Running times on large graphs (best viewed in color).

Table 4: Performance of NIGCN variants.

Variants	$\text{NIGCN}_{\text{HKPR}}$	$\text{NIGCN}_{\text{UDL}}$	$\text{NIGCN}_{\text{NFK}}$	NIGCN
F1-score (%)	61.60 ± 0.82	61.32 ± 1.32	52.76 ± 1.14	63.45 ± 0.70
Disparity (%)	-1.85	-2.13	-10.69	0

Table 5: F1-score (%) vs. label percentages.

Methods	4‰	8‰	1%	2%	5%
PPRGo	63.68 ± 1.13	66.02 ± 0.89	66.62 ± 0.86	67.54 ± 0.67	OOM
GBP	64.57 ± 0.66	68.31 ± 0.86	69.22 ± 0.71	71.56 ± 0.54	73.57 ± 0.45
NIGCN	**67.24 ± 0.60**	**70.93 ± 0.75**	**71.49 ± 0.91**	**73.19 ± 0.52**	**74.02 ± 0.31**

5.4 Ablation Study

Variants of NIGCN. To validate the effects of diffusion model NDM and the sampling techniques in NIGCN, we design three variants of NIGCN, i.e., $\text{NIGCN}_{\text{HKPR}}$, $\text{NIGCN}_{\text{UDL}}$, and $\text{NIGCN}_{\text{NFK}}$. Specifically, (i) $\text{NIGCN}_{\text{HKPR}}$ adopts heat kernel PageRank (HKPR) instead of general heat diffusion GHD in NDM as the diffusion function, (ii) $\text{NIGCN}_{\text{UDL}}$ unifies the diffusion length for all labeled nodes in contrast with the node-wise diffusion length in NDM, and (iii) $\text{NIGCN}_{\text{NFK}}$ removes the first-K limitation on the number of neighbors. We test all variants on Amazon and Table 4 reports the corresponding F1-scores. For clarification, we also present the F1-score disparity from that of NIGCN.

First of all, we observe that the F1-score of $\text{NIGCN}_{\text{HKPR}}$ is 1.04% smaller than that of NIGCN. This verifies that HKPR is not capable of capturing the structure characteristics of Amazon and NDM offers better generality. Second, $\text{NIGCN}_{\text{UDL}}$ acquires 1.32% less F1-scores compared with NIGCN. This suggests that diffusion with customized length leverages the individual structure property of each target node, which benefits the node classification. Last, $\text{NIGCN}_{\text{NFK}}$ achieves 9.88% smaller F1-score than NIGCN does, which reveals the potential noise signals from neighbors and recognizes the importance and necessity of important neighbor selection.

Label Percentages Varying. To evaluate the robustness of NIGCN towards the portion of labeled nodes, we test NIGCN by varying the label percentages in {4‰, 8‰, 1%, 2%, 5%} on Amazon[3] and compare it with two competitive baselines PPRGo and GBP. Results in Table 5 and Figure 4 report the F1-scores and running times respectively.

As displayed in table 5, NIGCN achieves the highest F1-score with average 1.93% advantages over the second highest scores across tested percentage ranges. Moreover, Figure 4 shows that

[3]Papers100M contains only 1.4% labeled nodes which is insufficient for testing.

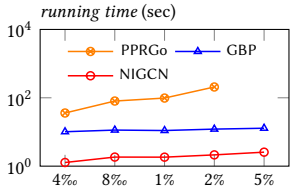


Figure 4: Running time.

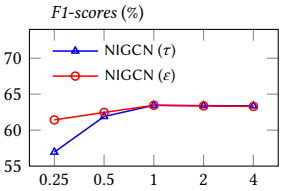


Figure 5: Performance.

We evaluate NIGCN by varying two parameters so as to test its sensitivity towards the change of the two parameters. The results present that NIGCN is more sensitive to aggregation length than the significance metric.]

NIGCN notably dominates the other two competitors in efficiency. In particular, NIGCN completes execution within 3 seconds and runs up to 5× to 20× faster than GBP and PPRGo in *all* settings respectively. These findings validate the robustness and outstanding performance of NIGCN for semi-supervised classification.

Parameter Analysis. The performance gap between $\text{NIGCN}_{\text{HKPR}}$ and NIGCN has shown that inappropriate combination of ω and ρ degrades the performance significantly. Here we test the effects of hyperparameters τ and ε in control of the diffusion length ℓ_u and the number of neighbors, respectively.

On Amazon, we select $\tau = 1.5$, denoted as τ_o and $\varepsilon = 0.05$, denoted as ε_o. We then test $\tau \in \{0.25\tau_o, 0.5\tau_o, 2\tau_o, 4\tau_o\}$ and $\varepsilon \in \{0.25\varepsilon_o, 0.5\varepsilon_o, 2\varepsilon_o, 4\varepsilon_o\}$ and plot the results in Figure 5. As shown, F1-score improves along with the increase of τ until $\tau = \tau_o$ and then decreases slightly as expected. Similar patterns are also observed in the case of ε. Specifically, NIGCN exhibits more sensitivity towards the change of τ than that of ε. This is because NIGCN is able to capture the most important neighbors within the right τ-distance with high probability when changing the threshold of ε-importance neighbors, which, however, is not guaranteed when altering the bound of τ-distance.

6 CONCLUSION

In this paper, we propose NIGCN, a scalable graph neural network built upon the node-wise diffusion model NDM, which achieves orders of magnitude speedups over representative baselines on massive graphs and offers the highest F1-score on semi-supervised classification. In particular, NDM (i) utilizes the individual topological characteristic and yields a unique diffusion scheme for each

target node and (ii) adopts a general heat diffusion function GHD that adapts well to various graphs. Meanwhile, to optimize the efficiency of feature aggregations, NIGCN computes representations for target nodes only and leverages advanced neighbor sampling techniques to identify and select important neighbors, which not only improves the performance but also boosts the efficiency significantly. Extensive experimental results strongly support the state-of-the-art performance of NIGCN for semi-supervised classification and the remarkable scalability of NIGCN.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments. This research is part of the programme DesCartes and is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. This work is also supported by the National Natural Science Foundation of China (NSFC) under Grant No. U22B2060 and by HKUST(GZ) under a Startup Grant.

REFERENCES

[1] 2020. https://europe.naverlabs.com/blog/web-image-search-gets-better-with-graph-neural-networks/.
[2] Anas Belahcen, Monica Bianchini, and Franco Scarselli. 2015. Web Spam Detection Using Transductive(Inductive Graph Neural Networks. In *Advances in Neural Networks: Computational and Theoretical Issues*. Vol. 37. 83–91.
[3] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling Graph Neural Networks with Approximate PageRank. In *SIGKDD*. 2464–2473.
[4] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR*.
[5] Jianfei Chen, Jun Zhu, and Le Song. 2018. Stochastic Training of Graph Convolutional Networks with Variance Reduction. In *ICML*, Vol. 80. PMLR, 941–949.
[6] Ming Chen, Zhewei Wei, Bolin Ding, Yaliang Li, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. 2020. Scalable Graph Neural Networks via Bidirectional Propagation. In *NeurIPS*.
[7] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *SIGKDD*. 257–266.
[8] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*.
[9] Fan RK Chung and Fan Chung Graham. 1997. *Spectral graph theory*. Number 92. American Mathematical Soc.
[10] Fan R. K. Chung and Lincoln Lu. 2006. Survey: Concentration Inequalities and Martingale Inequalities: A Survey. *Internet Math.* 3, 1 (2006), 79–127.
[11] James Demmel. 1997. *Applied Numerical Linear Algebra*. SIAM.
[12] Kaize Ding, Jianling Wang, Jundong Li, Kai Shu, Chenghao Liu, and Huan Liu. 2020. Graph Prototypical Networks for Few-shot Learning on Attributed Networks. In *CIKM*. 295–304.
[13] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *WWW*. 417–426.
[14] Wenzheng Feng, Yuxiao Dong, Tinglin Huang, Ziqi Yin, Xu Cheng, Evgeny Kharlamov, and Jie Tang. 2022. GRAND+: Scalable Graph Random Neural Networks. In *WWW*. 3248–3258.
[15] Tao Guo and Baojiang Cui. 2021. Web Page Classification Based on Graph Neural Network. In *IMIS*, Vol. 279. Springer, 188–198.
[16] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Neurips*. 1024–1034.
[17] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *arXiv:2005.00687* (2020).
[18] Wen-bing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. 2018. Adaptive Sampling Towards Fast Graph Representation Learning. In *NeurIPS*. 4563–4572.
[19] Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. 2021. Scaling Up Graph Neural Networks Via Graph Coarsening. In *SIGKDD*. 675–684.
[20] Lukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to Remember Rare Events. In *ICLR*.
[21] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[22] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.
[23] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion Improves Graph Learning. In *NeurIPS*. 13333–13345.
[24] Chang Li and Dan Goldwasser. 2019. Encoding Social Information with Graph Convolutional Networks forPolitical Perspective Detection in News Media. In *ACL*, Anna Korhonen, David R. Traum, and Lluís Màrquez (Eds.). 2594–2604.
[25] Pan Li, I (Eli) Chien, and Olgica Milenkovic. 2019. Optimizing Generalized PageRank Methods for Seed-Expansion Community Detection. In *NeurIPS*. 11705–11716.
[26] Zemin Liu, Yuan Fang, Chenghao Liu, and Steven C. H. Hoi. 2021. Node-wise Localization of Graph Neural Networks. In *IJCAI*. 1520–1526.
[27] Andreas Loukas. 2019. Graph Reduction with Spectral and Cut Guarantees. *J. Mach. Learn. Res.* 20 (2019), 116:1–116:42.
[28] Hoang Nt and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv:1905.09550* (2019).
[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] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. DeepInf: Social Influence Prediction with Deep Learning. In *SIGKDD*. 2110–2119.
[31] Aravind Sankar, Yozen Liu, Jun Yu, and Neil Shah. 2021. Graph Neural Networks for Friend Ranking in Large-scale Social Platforms. In *WWW*. 2535–2546.
[32] Franco Scarselli, Ah Chung Tsoi, and Markus Hagenbuchner. 2004. Computing personalized pageranks. In *WWW*. 382–383.
[33] 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.
[34] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. 2017. A Meta-Learning Perspective on Cold-Start Recommendations for Items. In *NeurIPS*. 6904–6914.
[35] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[36] Hanzhi Wang, Mingguo He, Zhewei Wei, Sibo Wang, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. 2021. Approximate Graph Propagation. In *SIGKDD*. 1686–1696.
[37] Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. 2021. Dissecting the diffusion process in linear graph convolutional networks. (2021).
[38] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *ICML*, Vol. 97. 6861–6871.
[39] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Peng He, Paul Weng, Han Gao, and Guihai Chen. 2019. Dual Graph Attention Networks for Deep Latent Representation of Multifaceted Social Effects in Recommender Systems. In *WWW*. 2091–2102.
[40] Yiqing Xie, Sha Li, Carl Yang, Raymond Chi-Wing Wong, and Jiawei Han. 2020. When Do GNNs Work: Understanding and Improving Neighborhood Aggregation. In *IJCAI*. 1303–1309.
[41] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*, Vol. 80. PMLR, 5449–5458.
[42] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *SIGKDD*. 974–983.
[43] Hanqing Zeng, Muhan Zhang, Yinglong Xia, Ajitesh Srivastava, Andrey Malevich, Rajgopal Kannan, Viktor K. Prasanna, Long Jin, and Ren Chen. 2021. Decoupling the Depth and Scope of Graph Neural Networks. In *NeurIPS*. 19665–19679.
[44] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. 2020. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *ICLR*.
[45] Lulu Zhang, Buqing Cao, Mi Peng, Yueying Qing, Guosheng Kang, Jianxun Liu, and Kenneth K Fletcher. 2021. Bilinear Graph Neural Network-Enhanced Web Services Classification. In *HPCC*. 189–196.
[46] Wentao Zhang, Mingyu Yang, Zeang Sheng, Yang Li, Wen Ouyang, Yangyu Tao, Zhi Yang, and Bin Cui. 2021. Node Dependent Local Smoothing for Scalable Graph Learning. In *NeurIPS*. 20321–20332.
[47] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. 2019. Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks. In *NeurIPS*. 11247–11256.

A APPENDIX

A.1 Proofs

Proof of Lemma 4.2. Before the proof, we first introduce *Chernoff Bound* [10] as follows. □

Lemma A.1 (Chernoff Bound [10]). *Let X_i be an independent variable such that for each $1 \le i \le \theta$, $X_i \in [0,1]$. Let $X = \frac{1}{\theta}\sum_i^{\theta} X_i$. Given $\zeta \in (0,1)$, we have*

$$\Pr[\mathbb{E}[X] - X \ge \zeta] \le \mathrm{e}^{-\theta\zeta^2/2\mathbb{E}[X]}. \tag{8}$$

Let v be an ε-importance neighbor of u in the ℓ-th hop. Suppose v has been visited $\theta_v^{(\ell)}$ times after $\theta = \frac{2\eta^2}{\varepsilon}\log(\frac{W_s}{\delta\varepsilon})$ WRWs are generated. Let $\hat{\phi}(\ell,u,v) = \frac{U(\omega,\rho,\ell)\theta_v^{(\ell)}}{\Theta}$ be the weight estimation of $\phi(\ell,u,v)$. As Chernoff indicates, we have

$$\begin{aligned}
&\Pr\left[\phi(\ell,u,v) - \hat{\phi}(\ell,u,v) \ge \frac{\phi(\ell,u,v)}{\eta}\right] \\
&\le \mathrm{e}^{-\frac{\theta\phi(\ell,u,v)^2}{\eta^2}/(2\phi(\ell,u,v))} \le \mathrm{e}^{-\frac{\theta\varepsilon}{2\eta^2}} \\
&\le \mathrm{e}^{-\frac{2\eta^2}{\varepsilon}\log(\frac{W_s}{\delta\varepsilon})\cdot\frac{\varepsilon}{2\eta^2}} = \frac{\delta\varepsilon}{W_s}.
\end{aligned}$$

Thus $\hat{\phi}(\ell,u,v) \ge \frac{\phi(\ell,u,v)}{\eta}$ holds with probability at least $1 - \frac{\delta\varepsilon}{W_s}$. Within L hops, the total weight is

$$\sum_{\ell=0}^{L} w^{(\ell)} \sum_{v\in\mathcal{V}} (\mathbf{D}^{-1}\mathbf{A})^{\ell}[u,v] = \sum_{\ell=0}^{L} w^{(\ell)} = W_s.$$

Thus, the total number of ε-importance neighbors of u is bounded by $\frac{W_s}{\varepsilon}$. By union bound, the total failure probability within L hops is no more than $\frac{\delta\varepsilon}{W_s}\frac{W_s}{\varepsilon} = \delta$, which completes the proof. □

A.2 Experimental Settings

Dataset Statistics. Table 6 presents the detailed statistics of the 7 tested datasets.

Running Environment. All experiments are conducted on a Linux machine with an NVIDIA RTX2080 GPU (10.76GB memory), Intel Xeon(R) CPU (2.60GHz), and 377GB RAM.

Implementation Details. By following the state of the art [6, 14, 36], we implement and NIGCN in PyTorch and C++. The implementations of GCN and APPNP are obtained from Pytorch Geometric[4], and the other five baselines are obtained from their official releases.

Parameter Settings. We mainly tune ω, ρ, τ, and ε for NIGCN. Table 7 reports the parameter settings adopted for each dataset. According to our analysis in Section 3.2, we tune ω in $[0.9, 1.5]$ and ρ in $[0.01, 0.1]$ for sparse datasets, i.e., Cora, Citeseer, and Pubmed to capture long-range dependency by a smooth expansion tendency; we tune ω in $[1, 10]$ and ρ in $[0.5, 1.5]$ to provide refined HKPR-alike properties, i.e., reaching peak at certain hop. For extremely dense dataset Amazon, we tune ω in $[0.8, 1.2]$ and ρ in $[1, 1.5]$ to realize optimized PPR-alike properties, i.e., exponentially decrease in short distance. For τ and ε, we search τ in $[0.5, 2]$ to find the best scaling factor for each dataset and tune ε in the range of $[0.005, 0.05]$. As stated, we fix $\eta = 2$ and $\delta = 0.01$.

[4] https://github.com/pyg-team/pytorch_geometric

Table 6: Dataset details.

Dataset	#Nodes (n)	#Edges (m)	#Features (f)	#Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3
Ogbn-arxiv	169,343	1,166,243	128	40
Reddit	232,965	114,615,892	602	41
Amazon	1,569,960	132,169,734	200	107
Papers100M	111,059,956	1,615,685,872	128	172

Table 7: Hyper-parameters of NIGCN.

Dataset	ω	ρ	τ	ε	η	δ
Cora	1.15	0.06	1.7	0.02	2	0.01
Citeseer	1.1	0.04	1.2	0.03	2	0.01
Pubmed	1.15	0.1	1.9	0.01	2	0.01
Ogbn-arxiv	9.0	0.85	1.5	0.008	2	0.01
Reddit	4.0	1.1	0.8	0.008	2	0.01
Amazon	0.9	1.15	1.5	0.05	2	0.01
Papers100M	7.4	1.3	0.6	0.01	2	0.01

Baselines for Inductive Learning. As stated, several baselines tested in transductive learning are not suitable for semi-supervised inductive learning. For example, GraphSAINT samples a random subgraph as the training graph and demands each node in the subgraph owns a label [44]. Nonetheless, the percentages of labeled nodes are small due to semi-supervised setting on large graphs (Ogbn-arxiv, Reddit, and Amazon), which degrades the performance of GraphSAINT notably. GRAND+ needs to sample a subset test nodes for loss calculation during training, which, however, is conflict with the setting of inductive learning.

A.3 Additional Experiments

Figure 6 presents the running times of the 13 tested methods in transductive learning. As shown, the efficiency of the tested methods on the three small datasets varies and there is no clear winners. In particular, AGP (resp. GCN) outperforms other methods on Cora and pubmed (resp. Citeseer). However, the F1-scores of AGP and GCN fall behind those of other models with a clear performance gap, as shown in Table 2. Recall that GRAND+ achieves the highest F1-scores on Cora and Citeseer, and our model NIGCN performs best on Pubmed. Nonetheless, GRAND+ runs up to $10\times \sim 20\times$ slower than NIGCN does.

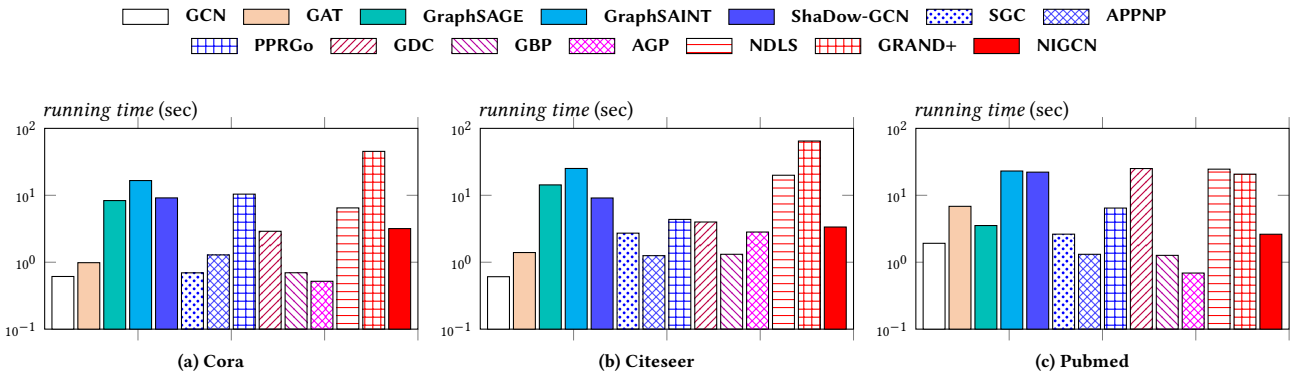


Figure 6: Running times on small graphs (best viewed in color).

Fully described in Appendix A.3.]