MAPN: Enhancing Heterogeneous Sparse Graph Representation by Mamba-based Asynchronous Aggregation

Xuqi Mao[1,3], Zhenying He[1,3], and X. Sean Wang[1,2,3](✉)

[1] School of Computer Science, Fudan University, Shanghai, China
[2] School of Software, Fudan University, Shanghai, China
[3] Shanghai Key Laboratory of Data Science, Shanghai, China
{xqmao17,zhenying,xywangCS}@fudan.edu.cn

Abstract. Graph neural networks (GNNs) have become the state of the art for various graph-related tasks and are particularly prominent in heterogeneous graphs (HetGs). However, several issues plague this paradigm: first, the difficulty in fully utilizing long-range information, known as over-squashing; second, the tendency for excessive message-passing layers to produce indistinguishable representations, referred to as over-smoothing; and finally, the inadequacy of conventional MPNNs to train effectively on large sparse graphs. To address these challenges in deep neural networks for large-scale heterogeneous graphs, this paper introduces the Mamba-based Asynchronous Propagation Network (MAPN), which enhances the representation of heterogeneous sparse graphs. MAPN consists of two primary components: node sequence generation and semantic information aggregation. Node sequences are initially generated based on meta-paths through random walks, which serve as the foundation for a spatial state model that extracts essential information from nodes at various distances. It then asynchronously aggregates semantic information across multiple hops and layers, effectively preserving unique node characteristics and mitigating issues related to deep network degradation. Extensive experiments across diverse datasets demonstrate the effectiveness of MAPN in graph embeddings for various downstream tasks underscoring its substantial benefits for graph representation in large sparse heterogeneous graphs.

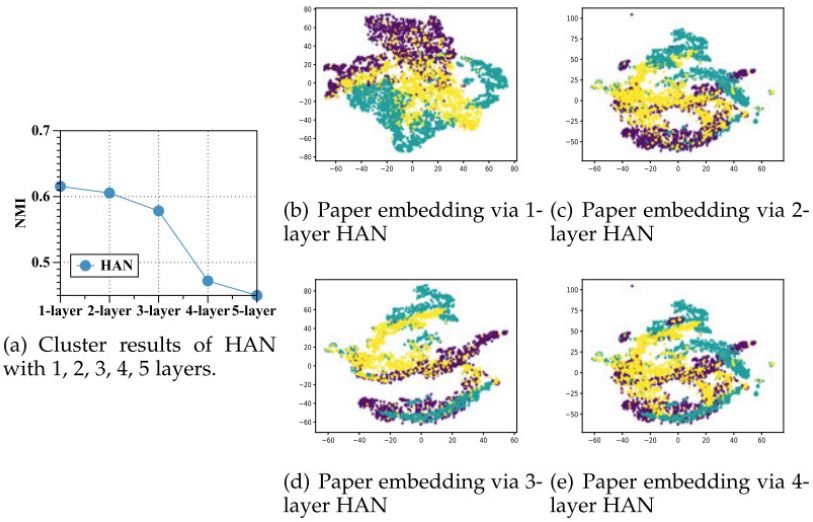
Keywords: heterogeneous graph · graph neural network · sparse graph.

1 Introduction

Heterogeneous Graphs (HetGs) can model numerous real-world datasets and represent diverse objects and relationships through various types of nodes and edges [12, 30, 2, 1, 22, 45, 27, 25, 24, 28]. The complexity and rich properties of HetGs have spurred a substantial body of research, especially in the fields of traditional graph representation learning and heterogeneous graph neural networks (HGNNs) [48, 36, 13].

As Graph Neural Networks (GNNs) continue to evolve, a variety of models based on Message-Passing Neural Networks (MPNNs) have emerged [43, 31]. Yet, these MPNNs face significant challenges. They often underperform on large sprse graphs where neighboring features vary greatly [51], and excessive message-passing layers can lead to indistinguishable embeddings, known as over-smoothing [6]. Additionally, the problem of over-squashing has been recognized, which underscores the difficulty MPNNs have in learning from long-range neighbors [39].

Increasing the depth of graph neural networks (GNNs) might improve the ability to capture information in graphs with more uniform node degree distributions. However, for extremely imbalanced graph structures, simply adding more layers to the network is not always an effective solution. In practice, performance degradation is often observed as the number of layers in heterogeneous GNNs increases. As illustrated in Figure 1, experiments on the ACM dataset show that the ability of HAN [42] to learning distinguishable paper representations diminishes significantly as the number of layers increases from one to five. This performance degradation primarily stems from the loss or dilution of information during the transmission process in deeper layers, leading to the homogenization of node features, which impairs the model's ability to distinguish between nodes and reduces training efficiency. Simply increasing the number of layers often leads to excessive computational resource consumption and overfitting, failing to capture the true structure of the graph.



(a) Cluster results of HAN with 1, 2, 3, 4, 5 layers.

(b) Paper embedding via 1-layer HAN

(c) Paper embedding via 2-layer HAN

(d) Paper embedding via 3-layer HAN

(e) Paper embedding via 4-layer HAN

Fig. 1. Degration of deep neural networks.

To bridge this gap, this study delves into the reasons behind model depth degradation and identifies deficiencies in current approaches to network propagation and information aggregation. We introduce the Mamba-based Asynchronous Propagation Network (MAPN), a novel approach designed to mitigate deep degradation in large, sparse heterogeneous graph representation through a state space model and a dual asynchronous aggregation mechanism. MAPN features two main components: node sequence generation and semantic information aggregation. The node sequence generation module efficiently extracts vital information from distant nodes based-on meta-path, and the semantic information aggregation module aggregates neighbor information within and across meta-paths, preserving node uniqueness during multi-hop aggregation across layers. This design ensures that the unique characteristics of each node are maintained in deep network structures, addressing the issue of deep degradation.

2 Preliminary

Definition 1 Heterogeneous Graph. A heterogeneous graph is a graph $G(V, E)$, where each node $v \in V$ is mapped to a specific node type $\theta(v) \in \mathcal{A}$ by $\theta : V \to \mathcal{A}$ and each edge $e \in E$ is mapped to a specific relation type $\kappa(e) \in \mathcal{R}$ by $\kappa : E \to \mathcal{R}$. In a heterogeneous graph, the number of distinct node types $|\mathcal{A}|$ is greater than 1, or the number of distinct relation types $|\mathcal{R}|$ is greater than 1.

Definition 2 Meta-path. A meta-path ϕ is a type sequence of nodes and edges defining a composite relation. It is typically represented by the notation of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$, where A_1 denotes the starting node type, A_{l+1} denotes the ending node type, and each intermediate relation R_i denotes a connection between node type A_i and A_{i+1}.

Definition 3 In a simple graph $G = (V, E)$, the l-th layer of the Semantic Propagation Network (MPNN) can be formalized as:

$$\begin{aligned} m_a^{(l)} &= \text{AGGREGATE}^{(l)}\left(\left\{H_{a'}^{(l-1)} : a' \in S^{(l)}(a)\right\}\right) \\ H_a^{(l)} &= \text{UPDATE}^{(l)}\left(H_a^{(l-1)}, m_a^{(l)}\right) \end{aligned} \tag{1}$$

where $H_a^{(0)}$ is initialized as the feature vector x_a of node a. The set $S^{(l)}(a)$ includes the nodes from which node a directly aggregates information in the l-th layer.

Definition 4 Synchronous MPNN. An L-layer MPNN is synchronous if and only if for any two layers $0 \le l' \le l \le L$ and any two nodes u and v, there is:

$$\frac{\partial H_a^{(l)}}{\partial H_b^{(l')}} \le f_{\theta^{(L)}}\left(\frac{\partial H_{a'}^{(l-1)}}{\partial H_b^{(l')}}\right), H_{a'}^{(l-1)} | a' \in \mathcal{B}_1(a). \tag{2}$$

where $\theta^{(L)}$ is the parameter of the MPNN at layer L.

Definition 5 State Space Model (SSM). The SSM represents dynamic systems by their states at each time step, as described by the following two equations

[16]: $\mathbf{h}'(t) = \mathbf{A}\mathbf{h}(t) + \mathbf{B}x(t)$ and $y(t) = \mathbf{C}\mathbf{h}(t) + \mathbf{D}x(t)$. Here, $\mathbf{h}(t) \in \mathbb{R}^n$ denotes the latent state of the system, $x(t) \in \mathbb{R}$ represents the input signal, $y(t) \in \mathbb{R}$ is the output signal. The parameters $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times 1}$, $\mathbf{C} \in \mathbb{R}^{m\times n}$, and $\mathbf{D} \in \mathbb{R}$ learnable. The SSM learns how to transform the input signal $x(t)$ into the latent state $\mathbf{h}(t)$, which is then used to model the system dynamics and predict its output $y(t)$.

Problem Definition. Given a sparse large heterogeneous graph $G = (V, E)$ and a set of relational constraints represented by a meta-path ψ, the heterogeneous graph representation learning maps each node a to a low-dimensional representation in the space defined by ψ. Specifically, the goal is to generate a representation for each node through a function $f : a \to \mathbf{a}^d, d \ll |V|$.

3 Methodology

This section provides a detailed description of the MAPN model architecture. As shown in Figure 2, MAPN first generates node sequences through random walks, and then constructs a state space model for each node based on these sequences. Using the state space model, node information is asynchronously aggregated through "hop"-level and "layer"-level jump connections, ultimately resulting in the node representations.

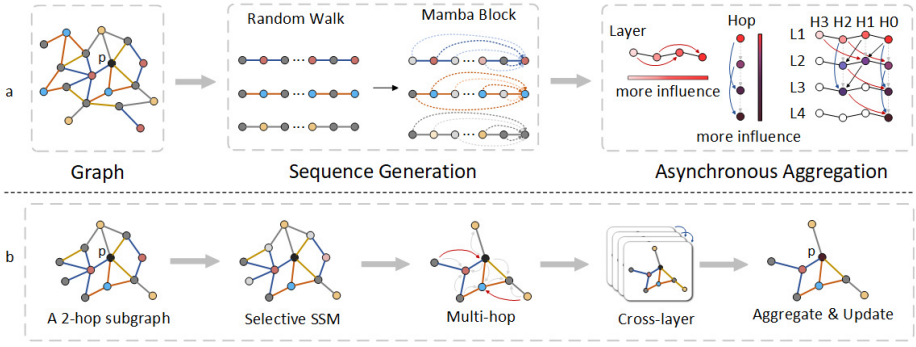


Fig. 2. Architecture of MAPN.

3.1 Node Sequences Generation Module

Structure Information Extaction Before discussing the heterogeneous aggregation process, we first address the issue of unequal data distribution, which can negatively impact the embedding of "hub" nodes and inadequately represent "cold-start" nodes. To mitigate this, MAPN begins with MPU-based sampling using random walk with restart (RWR) in two steps: (a) Sampling Fixed-Length RWR. Beginning with a node $a \in V^\phi$, the random walker either moves to an

adjacent node or returns to the starting node with probability p until a fixed number of nodes for each type are collected. (b) Grouping Neighbors by Type. For each node type $A \in \mathcal{A}$, the top k_A neighbors most frequently encountered during the RWR are selected as A-type neighbors, based on the intuition that frequently co-occurring nodes are more similar. This process enables MAPN to effectively capture and utilize the rich local information around each node, enhancing the quality of node embeddings.

Inspired by recent advancements in Mamba, this paper introduces a sequence modeling architecture based on Mamba under meta-path ϕ. This approach uses a selection mechanism to preserve information from nodes at various distances, enhancing the HGNNs to capture node information in sparse large graphs. Topological data from random walks are used as input sequences, and the selection mechanism updates the hidden states based on relevant previous nodes, controlled by the current node. The variable g_t, ranging from 0 to 1, allows the model to completely filter out irrelevant contexts when needed. This mechanism refines long-distance dependencies while minimizing the influence of less relevant nodes, enhancing graph attention and preserving key dependencies within long sequences. Specifically, MAPN employs a state space model based on the meta-path ϕ to differentiate between relevant and irrelevant information.

3.2 Semantic Aggregation Module

Node Information Transformation In heterogeneous graphs with node attributes, feature vectors for different content types of a node a often have unequal dimensions and belong to distinct feature spaces. As a result, managing feature vectors of diverse dimensions within a unified framework presents significant challenges. To address this, a two-step transformation is used to capture node information.

Type-Specific Transformation For each content type of a node, features are projected into a unified feature space via a type-specific neural network f. For the n-th content of node a, we have:

$$H_{an} = f_n\big(C_{an}\big) \tag{3}$$

where C_{an} is the original feature vector of n-th content for node a and H_{an} is the projected latent vector, f_n is the type-specific transformation function, which can be pre-trained using different techniques for various content types. The node information of node a can then be represented as:

$$H_a = \{H_{an}, n \in |C_a|\} \tag{4}$$

where H_a is the collection of latent feature vectors of node a, and $|C_a|$ denotes the amount of content feature of node a.

Node Information Aggregation After transforming the content for each node, all node features are standardized to the same dimension. We utilize Bidirectional LSTM, as described by [17], to aggregate the diverse set of unordered features of the node. The feature embedding process is as follows:

$$\hat{H}_a = \frac{\sum_{n \in H_a} \left[\overrightarrow{LSTM}\left(H_{an}\right) \bigoplus \overleftarrow{LSTM}\left(H_{an}\right) \right]}{|H_a|} \tag{5}$$

where $\bigoplus$ denotes concatenation. This method enables the integration of features from neighbors of the same type, ensuring comprehensive representation in the embeddings.

This operation aligns the projected features of all nodes to a uniform dimension, facilitating the subsequent processing of nodes of arbitrary types.

Semantics Information Aggregation

Synchronous Semantic Aggregation Local priority by layers: Extracting distinguishable category information from the feature matrix is crucial for the success of GNNs. However, the synchronous nature of deeper GNNs leads to a loss of important original information, causing them to fail in retaining sufficient distinctions.

Theorem 1. *Consider an L-layer Semantic Propagation Network, suppose the activation function $\sigma(\cdot)$ has a Lipschitz constant α, and the norm of each element in W is bounded, such that $\|W\|_p \leq c$. Within a K-regular graph, for any two nodes a and b, and any layer range $0 \leq l' < l \leq L$, the following relationship is established:*

$$\sum_{a \in V} \left|\left| \frac{\partial H_a^{(l)}}{\partial H_b^{(l')}} \right|\right|_p^2 \leq C \sum_{a \in V} \left(\frac{\partial H_a^{(l-1)}}{\partial H_b^{(l')}} \right)_p^2, where C = \frac{\alpha^2 c^2 K^2}{(K+1)^2} \tag{6}$$

This theorem demonstrates how influence from any layer l' propagates across l layers. For most GNN activation functions such as tanh, sigmoid, and ReLU, α generally does not exceed 1. Input features X are typically normalized, keeping c small [38]. Consequently, $C \leq 1$ in most scenarios, indicating that the impact of the original input weakens with increasing layer depth, favoring local processing. Skip connections, particularly initial ones, help maintain more original features by reducing the instances of compression, thereby avoiding excessive smoothing in deeper layers.

Local priority by hops: Information flow from higher-order neighbors must pass through lower-order ones to reach the central node. The Ollivier-Ricci curvature $\kappa(a, b)$, which ranges from (-1, 2) [3, 21], measures the connectivity between one-hop neighbors and helps examine the hop distance effect on information flows (IFs). Higher $\kappa(a, b)$ indicates stronger local connectivity.

Theorem 2. *Consider an L-layer semantic propagation network, where the activation function $\sigma(\cdot)$ is the identity function. Define η as the lower bound for the Ollivier-Ricci curvature on a K-regular graph, meaning $\kappa(a,b) \geq \eta$ for every edge (a,b) in E. If $\eta \geq \frac{1}{2} - \frac{3}{2K}$, the following holds:*

$$\sum_{a \in N_1(b)} \left\| \frac{\partial h_u^{(l+2)}}{\partial h_v^{(l)}} \right\|_p \geq \sum_{a \in N_2(v)} \left\| \frac{\partial H_a^{(l+2)}}{\partial H_v^{(l)}} \right\|_p \tag{7}$$

Focusing on one-hop and two-hop neighbors reveals the local structure's impact on central nodes in MAPN. The theorem shows that denser local connections (higher κ) and fewer one-hop neighbors (lower K) lead to greater gradient decay with hop distance, demonstrating a preference for hopping. In contrast, multi-hop propagation reduces local density (lower κ) and increases node degree (higher K), aiding GNNs in reducing the influence of irrelevant first-order neighbors while preserving long-distance dependencies.

Asychronous Semantic Aggregation As analyzed earlier, overcoming the local priority caused by synchrony is key to enhancing MPNN performance. This section presents an asynchronous semantic propagation network that uses a multi-hop messaging method based on shortest paths to speed up communication between different node pairs, as illustrated in Fig. 3. This approach not only preserves original information from distant endpoints but also reduces information compression. Specifically, each node gathers information from neighbors within k hops, which is then filtered through a selective state space for aggregation, ensuring retention of distance-related information.

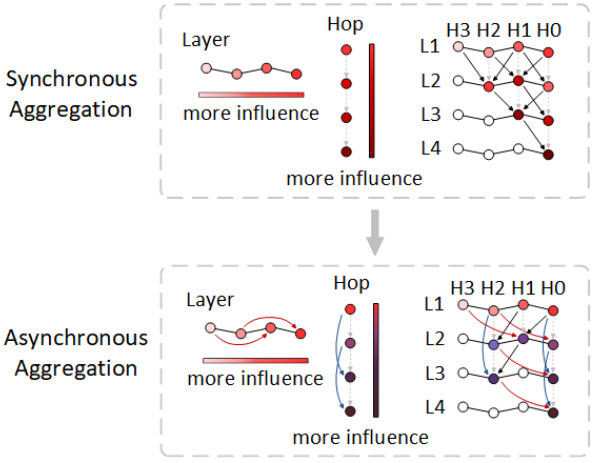


Fig. 3. Asynchronous aggregation.

Intra-meta-path-aggregation: The process selectively filters and learns about the structural and semantic information of the target node, its neighbors, and their interrelationships using a selective state space.

MAPN employs a bidirectional LSTM for aggregating node features. As described in [17], this approach captures long-distance information transmission and maintains long-term dependencies. The computation of feature embeddings is as follows:

$$\hat{H}^{\phi}(a) = \frac{\sum_{n \in H^{\phi}(a)} \left[\overrightarrow{LSTM} \left(H_n^{\phi}(a) \right) \bigoplus \overleftarrow{LSTM} \left(H_n^{\phi}(a) \right) \right]}{|H^{\phi}(a)|} \tag{8}$$

where $\bigoplus$ denotes concatenation.

To unify the feature vectors of different types of nodes, MAPN aggregates the fixed-size neighbor set of each node a using LSTM.

$$Q^{\phi}(a) = \frac{\sum_{b \in N^{\phi}(a)} \left[\overrightarrow{LSTM} \left(\hat{H}^{\phi}(b) \right) \bigoplus \overleftarrow{LSTM} \left(\hat{H}^{\psi}(b) \right) \right]}{|N^{\phi}(a)|} \tag{9}$$

where $N^{\phi}(a)$ denotes the sampled neighbor set of node a, where b is a neighbor of a within the meta-path ϕ.

Inter-meta-path-aggregation: Nodes across various meta-paths ϕ contribute differently to the target node's representation, prompting MAPN to filter their vector representations using a selective state space [15]:

$$\begin{aligned}
&\alpha^{\phi}(a,b) = \frac{exp\big(LeakyReLU\left(u^T[Q^{\phi}(a) \bigoplus Q^{\psi}(b)]\right)\big)}{\sum_{s \in N_a} exp(LeakyReLU\left(u^T[Q^{\phi}(a) \bigoplus Q^{\phi}(s)]\right))} \\
&H(a) = \overline{\mathbf{A}}\alpha^{\phi}(a,b) + \overline{\mathbf{B}}u(a) \\
&y(a) = \mathbf{C}H(a) \\
&Zi_a^{\phi} = \sum\nolimits_{b \in N_a} y(a) \odot Q^{\phi}(b)
\end{aligned} \tag{10}$$

where, $e^{\phi}(a,b)$ indicates the importance of node b from V^{ϕ} to a, and $(a^{\phi})^T$ is the node-level attention vector within ϕ. The weight $\alpha^{\phi}(a,b)$ is normalized using the softmax function, with $y(a)$ being the output from the selective state space, allowing the integration of diverse node features into a unified vector space within ϕ.

Considering the varying significance of each meta-path in heterogeneous graphs, MAPN introduces an attention mechanism to assign differential weights to them. Specifically, $w^{\psi}(a)$ assesses the importance of the meta-path ϕ for node a, which is then normalized using softmax to compute $\beta^{\phi}(a)$, the weight of ψ for a. These weights help MAPN synthesize a specific representation Z^{ψ} for node a

in ϕ.

$$\begin{aligned}
\beta^{\psi}(a) &= \frac{exp\left(LeakyReLU\left(q^T \odot Zi^{\psi}(a)\right)\right)}{\sum_{i\in|\psi|} exp\left(LeakyReLU\left(q^T \odot Zi^{i}(a)\right)\right)} \\
H(a) &= \overline{\mathbf{A}}\beta^{\psi}(a) + \overline{\mathbf{B}}u(a) \\
y(a) &= \mathbf{C}H(a) \\
Zw^{\psi}(a) &= y(a) \cdot Zi^{\psi}(a)
\end{aligned} \tag{11}$$

where LeakyReLU is the activation function and q^T is the parameterized attention vector. MAPN aggregates information within and between meta-paths to derive node representations Z^{ψ} specific to particular semantics, employing node representations' inner product to establish node similarities.

3.3 Training

We optimize the model weights by minimizing the cross-entropy loss function using negative sampling [29]:

$$\mathcal{L}^{\psi} = \sum_{\langle a,b,b'\rangle \in \tau^{\psi}} \log\sigma\Big((Z^{\psi}(a)) \cdot Z^{\psi}(b)\Big) - \log\sigma\Big(-(Z^{\psi}(a)) \cdot Z^{\psi}(b')\Big) \tag{12}$$

where $\sigma(\cdot)$ represents the sigmoid function, τ denotes the set of triples $\langle a, b, b'\rangle$ collected by walk sampling based on MPU ψ in the heterogeneous graph.

4 Experiments

In this section, we begin by outlining the experimental settings. We then evaluate the performance of MAPN across several benchmarks: node classification (Section 5.2), graph classification (Section 5.3), and long-range learning (Section 5.4). In Section 5.5, we delve into an analysis of critical parameter in MAPN.

4.1 Experimental Setup

Model Setting. For node and graph classification tasks, we employ a simplified backbone model to ensure fair comparisons across different versions of MAPN. Specifically, we use a basic, parameter-free average aggregation defined as:

$$h(v)^l = \text{Mean}(\{h(v)^{l-1} | v \in B(v)\}).$$

The optimal number of backbone layers l and the receptive field size B are determined based on the best values listed in Tables 6 and 7. For more demanding long-range learning tasks, we utilize the GCN framework from the Long Range Graph Benchmark (LRGB) [8] as our backbone. In line with recommendations from SPN [1], we set l to 10 and B to 8, with all learnable hidden layer dimensions fixed at 300. We also incorporate Laplacian positional encoding (LapPE) [15] to

improve the model's ability to learn structural information, which is crucial for long-range tasks [24, 36, 44].

Training Setting. During training, we utilize the AdamW optimizer with a default learning rate $lr = 0.1$ and $weight\_decay = 0$. We employ a ReduceLROnPlateau scheduler for experiments on LRGB, and a CosineAnnealingWarmRestarts scheduler for other tasks. The maximum number of epochs is set at 500.

Baselines. MAPN was evaluated against the following baseline methods, (1) MPNNs: GCN [19], GIN [43], Gated-GCN [4], GPS (BigBird) [33], Exphormer [37], GraphSAGE [17], GAT [41]; (2)HGNNs: RGCN [34], HAN [42], MAGNN [11], HetGNN [49], SeHGNN [46], HGT [18]; (3)our method: MAPN.

4.2 Node Classification

Table 1 compares the performance of four graph neural network models across various datasets. The MAPN model consistently outperforms the others, showcasing its robustness and efficiency across diverse data environments, especially on datasets like ACM and DBLP where it has a clear advantage. SeHGNN also demonstrates excellent results, with matching or exceeding MAPN. It excels in processing data with complex structural relationships.

The performance disparity across different datasets is notable. For instance, all models struggle on the IMDB dataset, likely due to unique challenges such as class imbalance or feature heterogeneity. In contrast, data sets such as ACM and DBLP exhibit high performance across all models, suggesting better feature representation and less noise. HGNNs perform better than methods on homogeneous graphs because they are optimized for heterogeneous graphs, enabling them to aggregate different types of edges and nodes according to their various relations. Among HGNNs, the HGT model shows varied results, excelling in IMDB but underperforming on ogbn-mag, highlighting its sensitivity to complex data relations.

4.3 Graph Classification

Table 2 showcases the performance comparison of GCN, GIN, GT, and MAPN across different datasets. The MAPN model consistently demonstrates superior performance, suggesting its more effective learning mechanisms or feature capture capabilities, particularly in bioinformatics data. Notably, MAPN achieves 0.8481 on the MUTAG dataset, significantly outperforming other models. While GCN shows consistent results across all datasets with performances above 0.75, it generally underperforms compared to MAPN, indicating its structure may not be as adaptable or efficient for these tasks. Both GIN and GT exhibit similar performances across various datasets, with GT slightly leading on the NCI109 dataset at 0.7862 versus 0.7758 for GIN, suggesting GT may be slightly better at handling certain types of structured data. The impact of dataset complexity is also evident, as all models score lower on the PROTEINS dataset, possibly

Table 1. Node classification task.

	ogbn-mag Accuracy	**ACM** F1 score	**DBLP** F1 score	**IMDB** F1 score
GCN	0.3489	0.8743	0.9084	0.5273
GIN	0.3613	0.8832	0.9013	0.5127
GatedGCN	0.3578	0.8902	0.9095	0.5378
GPS(BigBird)	0.3390	0.8769	0.8896	0.5258
Exphormer	0.3934	0.8845	0.9107	0.5332
GraphSAGE	0.4678	0.8983	0.9182	0.5591
GAT	0.3767	0.8845	0.9142	0.5380
RGCN	0.4737	0.9141	0.9207	0.6205
HAN	0.5037	0.9079	0.9205	0.6463
MAGNN	0.4926	0.9077	0.9376	0.6467
HetGNN	0.4732	0.8591	0.9176	0.4825
SeHGNN	0.5671	0.9367	0.9524	0.6921
HGT	0.4929	0.9100	0.9349	0.6720
MAPN	0.5744	0.9283	0.9615	0.6876

due to its more challenging or complex biomolecular structures, which hinder higher accuracy.

Table 2. The performance of MAPN on graph classification tasks.

Dataset	GCN	GIN	GT	MAPN
MUTAG	0.7371	0.7783	0.7602	0.8481
NCI1	0.7781	0.7528	0.7652	0.8455
NCI109	0.7806	0.7758	0.7862	0.8370
DD	0.7777	0.7738	0.7791	0.8199
PROTEINS	0.7579	0.7586	0.7469	0.7889

4.4 Experiment on Long-Range Graph Benchmark (LRGB)

Table 3 shows performance metrics of various models on the Peptides dataset for function (Peptides-func, measured by AP value improvement) and structure (Peptides-struct, measured by MAE reduction). Performance is categorized based on the use of Laplacian positional encoding (LapPE). Generally, using LapPE (marked as "✔") substantially boosts model performance. For example, in function prediction, the MAPN model scores an AP of 0.6832 with LapPE compared to 0.7144 without. This indicates that LapPE likely enhances model sensitivity and accuracy during preprocessing. The DRew model excels in function prediction, outperforming others regardless of LapPE usage, possibly due to superior data handling mechanisms. In structure prediction, DRew also records

lower MAE values, particularly with LapPE (0.2536), showcasing its effectiveness in predicting peptide structures accurately. Overall, model performance improves with LapPE, underscoring its critical role in feature extraction and preprocessing, significantly affecting outcomes.

Table 3. The performance of MAPN on two LRGB datasets.

Metric	Use LapPE	GCN	MAPN	DRew
Peptides-func (AP↑)	✓	—	0.7144	0.7150
	×	0.5930	0.6832	0.6996
Peptides-struct (MAE↓)	✓	—	0.2545	0.2536
	×	0.3496	0.2711	0.2781

4.5 Experiments on Hyperparameters

Table 4 shows how various datasets perform across different hop counts ($K = 1, 2, 3, 4$), exploring the impact of increasing hops on heterogeneous graph neural network performance. On datasets such as Texas, Wisconsin, and Cornell, performance generally improves with more hops, peaks, and then begins to decline. For example, on the Texas dataset, performance increases from 0.7204 at $K = 1$ to 0.8849 at $K = 2$, then slightly decreases, suggesting that while expanding the scope of information aggregation captures broader context, too many hops can cause over-smoothing and reduce node distinctions.

Conversely, on more heterogeneous datasets like Squirrel and Chameleon, performance steadily improves with additional hops, indicating that more hops help the model learn richer structural and semantic information. For instance, on Chameleon, performance significantly rises from 0.3961 at $K = 1$ to 0.6596 at $K = 4$.

However, on more homogeneous datasets like Wiki CS, Citeseer, Computers, Photos, and Physics, increasing hops generally leads to a decline in performance. This trend suggests that too many hops on these datasets cause an excessive blending of features, losing unique node-specific information and negatively impacting performance.

Overall, the number of hops crucially influences the effectiveness of heterogeneous graph neural networks. Properly setting the hop count can enhance learning on complex graph structures, but excessive hops can degrade performance, particularly on more homogeneous datasets. Thus, choosing the optimal number of hops is key to optimizing network performance.

Table 5 presents graph classification results for five datasets (UTAG, NCI1, NCI109, DD, PROTEINS) across different K values (1, 2, 3, 4). For most datasets, MAPN's performance improves as K increases from 1 to 3, suggesting

Table 4. Impact of K on node classification tasks.

Dataset	$K = 1$	$K = 2$	$K = 3$	$K = 4$
Texas	0.7204	0.8849	0.8761	0.8408
Wisconsin	0.8363	0.8946	0.8729	0.8659
Squirrel	0.3022	0.3314	0.3490	0.3441
Chameleon	0.3961	0.5146	0.5620	0.6596
Cornell	0.7215	0.7776	0.7964	0.8178
Wiki CS	0.7825	0.7710	0.7596	0.7489
Citeseer	0.7484	0.7588	0.7483	0.7362
Computers	0.8791	0.8752	0.8629	0.8603
Photos	0.9457	0.9358	0.9058	0.8650
Physics	0.9593	0.9511	0.9215	0.8763

that higher K values better capture the data's deeper structural or relational features, enhancing predictive accuracy. At $K = 3$, performance peaks for almost all datasets, particularly for UTAG and NCI1, which exhibit notable improvements. This indicates that an optimal K value allows the model to effectively leverage the data's inherent structure.

Performance disparities across the datasets at the same K values indicate that UTAG and NCI1 consistently outperform DD and PROTEINS. These variations may be due to differences in each dataset's characteristics, such as sample size, complexity, or noise levels. Although increasing K generally enhances performance, a slight decline at $K = 4$ for datasets like UTAG and NCI109 suggests potential overfitting or excessive model complexity, which may impede effective learning. Based on these insights, using $K = 3$ is recommended for handling these datasets as it balances performance gains with overfitting risks.

Overall, the choice of K significantly influences model performance on different datasets, making the selection of an optimal K value crucial for maximizing model effectiveness.

Table 5. Impact of K on graph classification tasks.

Dataset	**K = 1**	**K = 2**	**K = 3**	**K = 4**
UTAG	0.8512	0.8906	0.8964	0.8897
NCI1	0.8337	0.8376	0.8586	0.8583
NCI109	0.8185	0.8273	0.8334	0.8353
DD	0.7823	0.7904	0.8012	0.8031
PROTEINS	0.7784	0.7957	0.7934	0.8105

5 Related Work

Over the past decade, numerous research on mining information from graphs have shifted from traditional representation learning approaches [32, 14, 7] to methods utilizing deep neural networks, including GNNs [9, 44, 50, 53, 35, 26] and GCNs [19, 23]. Inspired by the Transformer [40], GAT [41] integrates the attention to aggregate node-level information in homogeneous networks, while HAN [42] introduces a two-level attention mechanism for node and semantic information in heterogeneous networks. MAGNN [11], MHGNN [20] and R-HGNN [47] proposed meta-path-based models to learn meta-path-based node embeddings. HetGNN [49] and MEGNN [5] take a meta-path-free approach to consider both structural and content information for each node jointly. HGT [18] incorporates information from high-order neighbors of different types through messages passing across "soft" meta-paths. MHGCN [10] captures local and global information by modeling the multiplex structures with depth and breadth behavior pattern aggregation. SeHGNN [46] simplifies structural information capture by precomputing neighbor aggregation and incorporating a transformer-based semantic fusion module. HAGNN [52] integrates meta-path-based intra-type aggregation and meta-path-free inter-type aggregation to generate the final embeddings.

6 Discussion and Conclusion

In this study, we present MAPN, a Mamba-based asynchronous which combines a selective state space model and dual asynchronous aggregation strategy to mitigate network degradation. The method consists of two core modules: node sequence generation and asynchronous semantic aggregation. First, random walk techniques are employed to generate initial node sequences. Then, neighbor information are aggregated asynchronously based on meta-paths across multiple hops and layers. The selective state space model is key to filtering the information from nodes at varying distances, retaining only the most crucial data. This approach not only deepens our understanding of the graph structure but also reduces the homogenization of information during multi-layer network propagation. Extensive experiments and analyses confirm the effectiveness of the MAPN model for capturing information from sparse large HetGs.

Acknowledgments. This work was mainly supported by the National Natural Science Foundation of China (NSFC No. 61732004).

References

1. Agrawal, N., Sirohi, A.K., Kumar, S., et al.: No prejudice! fair federated graph neural networks for personalized recommendation. In: AAAI. vol. 38, pp. 10775–10783 (2024)
2. Avery, K., Houmansadr, A., Jensen, D.: The effect of alter ego accounts on a/b tests in social networks. In: WWW. pp. 565–568 (2024)

3. Bauer, F., Jost, J., Liu, S.: Ollivier-ricci curvature and the spectrum of the normalized graph laplace operator. CoRR (2011), https://arxiv.org/abs/1105.3803
4. Bresson, X., Laurent, T.: Residual gated graph convnets. arXiv preprint arXiv:1711.07553 (2017)
5. Chang, Y., Chen, C., Hu, W., Zheng, Z., Zhou, X., Chen, S.: Megnn: Meta-path extracted graph neural network for heterogeneous graph representation learning. Knowledge-Based Systems **235**, 107611 (2022)
6. Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., Sun, X.: Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: AAAI. vol. 34, pp. 3438–3445 (2020)
7. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: Scalable representation learning for heterogeneous networks. In: SIGKDD. pp. 135–144 (2017)
8. Dwivedi, V.P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A.T., Beaini, D.: Long range graph benchmark. NIPS **35**, 22326–22340 (2022)
9. Fan, S., Zhu, J., Han, X., Shi, C., Hu, L., Ma, B., Li, Y.: Metapath-guided heterogeneous graph neural network for intent recommendation. In: SIGKDD. pp. 2478–2486 (2019)
10. Fu, C., Zheng, G., Huang, C., Yu, Y., Dong, J.: Multiplex heterogeneous graph neural network with behavior pattern modeling. In: SIGKDD. pp. 482–494. ACM (2023)
11. Fu, X., Zhang, J., Meng, Z., King, I.: Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In: WWW. pp. 2331–2341 (2020)
12. Gamba, D., Yu, Y., Yuan, Y., Schoenebeck, G., Romero, D.M.: Exit ripple effects: Understanding the disruption of socialization networks following employee departures. In: WWW. pp. 211–222 (2024)
13. Gao, Y., Zhang, P., Zhou, C., Yang, H., Li, Z., Hu, Y., Yu, P.S.: Hgnas++: Efficient architecture search for heterogeneous graph neural networks. TKDE **35**(9), 9448–9461 (2023)
14. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: SIGKDD. pp. 855–864 (2016)
15. Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces. CoRR (2023), https://doi.org/10.48550/arXiv.2312.00752
16. Gu, A., Goel, K., Ré, C.: Efficiently modeling long sequences with structured state spaces. In: ICLR (2022)
17. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. NIPS **30** (2017)
18. Hu, Z., Dong, Y., Wang, K., Sun, Y.: Heterogeneous graph transformer. In: WWW. pp. 2704–2710 (2020)
19. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
20. Liang, X., Ma, Y., Cheng, G., Fan, C., Yang, Y., Liu, Z.: Meta-path-based heterogeneous graph neural networks in academic network. International Journal of Machine Learning and Cybernetics pp. 1–17 (2022)
21. Lin, Y., Lu, L., Yau, S.T.: Ricci curvature of graphs. Tohoku Mathematical Journal, Second Series **63**(4), 605–627 (2011)
22. Liu, S., Cai, Q., He, Z., Sun, B., McAuley, J., Zheng, D., Jiang, P., Gai, K.: Generative flow network for listwise recommendation. In: SIGKDD. pp. 1524–1534 (2023)
23. Liu, X., Zhang, K., Liu, Y., Chen, E., Huang, Z., Yue, L., Yan, J.: Rhgn: Relation-gated heterogeneous graph network for entity alignment in knowledge graphs. In: ACL. pp. 8683–8696 (2023)

24. Lu, K., Yu, Y., Fei, H., Li, X., Yang, Z., Guo, Z., Liang, M., Yin, M., Chua, T.S.: Improving expressive power of spectral graph neural networks with eigenvalue correction. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 14158–14166 (2024)
25. Lu, K., Yu, Y., Huang, Z., Li, J., Wang, Y., Liang, M., Qin, X., Ren, Y., Chua, T.S., Wang, X.: Addressing heterogeneity and heterophily in graphs: A heterogeneous heterophilic spectral graph neural network. arXiv preprint arXiv:2410.13373 (2024)
26. Ma, Y., Yan, N., Li, J., Mortazavi, M.S., Chawla, N.V.: Hetgpt: Harnessing the power of prompt tuning in pre-trained heterogeneous graph neural networks. In: WWW. pp. 1015–1023. ACM (2024)
27. Mao, X., He, Z., Jing, Y., Zhang, K., Wang, X.S.: Hetfs: a method for fast similarity search with ad-hoc meta-paths on heterogeneous information networks. WWW **27**(6), 66 (2024)
28. Mao, X., He, Z., Wang, X.S.: Fhge: A fast heterogeneous graph embedding with ad-hoc meta-paths. arXiv preprint arXiv:2502.13373 (2025)
29. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. NIPS **26** (2013)
30. Muppasani, B., Narayanan, V., Srivastava, B., Huhns, M.N.: Expressive and flexible simulation of information spread strategies in social networks using planning. In: AAAI. vol. 38, pp. 23820–23822 (2024)
31. Pei, H., Wei, B., Chang, K.C.C., Lei, Y., Yang, B.: Geom-gcn: Geometric graph convolutional networks. arXiv preprint arXiv:2002.05287 (2020)
32. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: SIGKDD. pp. 701–710 (2014)
33. Rampášek, L., Galkin, M., Dwivedi, V.P., Luu, A.T., Wolf, G., Beaini, D.: Recipe for a general, powerful, scalable graph transformer. Advances in Neural Information Processing Systems **35**, 14501–14515 (2022)
34. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: ESWC. pp. 593–607. Springer (2018)
35. Shan, D., Du, X., Wang, W., Wang, N., Liu, A.: Kpi-hgnn: Key provenance identification based on a heterogeneous graph neural network for big data access control. Information Sciences **659**, 120059 (2024)
36. Shi, C.: Heterogeneous Graph Neural Networks, pp. 351–369. Springer Nature Singapore, Singapore (2022)
37. Shirzad, H., Velingker, A., Venkatachalam, B., Sutherland, D.J., Sinop, A.K.: Exphormer: Sparse transformers for graphs. In: International Conference on Machine Learning. pp. 31613–31632. PMLR (2023)
38. Tang, H., Liu, Y.: Towards understanding generalization of graph neural networks. In: Proceedings of the 40th International Conference on Machine Learning. vol. 202, pp. 33674–33719. PMLR (2023)
39. Topping, J., Di Giovanni, F., Chamberlain, B.P., Dong, X., Bronstein, M.M.: Understanding over-squashing and bottlenecks on graphs via curvature. arXiv preprint arXiv:2111.14522 (2021)
40. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. NIPS **30** (2017)
41. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., et al.: Graph attention networks. stat **1050**(20), 10–48550 (2017)
42. Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., Yu, P.S.: Heterogeneous graph attention network. In: The world wide web conference. pp. 2022–2032 (2019)

43. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: ICLR (2019)
44. Yan, Q., Zhang, Y., Liu, Q., Wu, S., Wang, L.: Relation-aware heterogeneous graph for user profiling. In: ICKM. pp. 3573–3577 (2021)
45. Yang, M., Zhu, M., Wang, Y., Chen, L., Zhao, Y., Wang, X., Han, B., Zheng, X., Yin, J.: Fine-tuning large language model based explainable recommendation with explainable quality reward. In: AAAI. vol. 38, pp. 9250–9259 (2024)
46. Yang, X., Yan, M., Pan, S., Ye, X., Fan, D.: Simple and efficient heterogeneous graph neural network. In: AAAI. pp. 10816–10824. AAAI Press (2023)
47. Yu, L., Sun, L., Du, B., Liu, C., Lv, W., Xiong, H.: Heterogeneous graph representation learning with relation awareness. TKDE **35** (2023)
48. Yu, Z., Jin, D., Huo, C., Wang, Z., Liu, X., Qi, H., Wu, J., Wu, L.: Kgtrust: Evaluating trustworthiness of siot via knowledge enhanced graph neural networks. In: WWW. pp. 727–736. ACM (2023)
49. Zhang, C., Song, D., Huang, C., Swami, A., Chawla, N.V.: Heterogeneous graph neural network. In: SIGKDD. pp. 793–803 (2019)
50. Zhang, S., Zhang, J., Song, X., Adeshina, S., Zheng, D., Faloutsos, C., Sun, Y.: Page-link: Path-based graph neural network explanation for heterogeneous link prediction. In: WWW. p. 3784–3793. WWW '23, Association for Computing Machinery (2023)
51. Zheng, X., Wang, Y., Liu, Y., Li, M., Zhang, M., Jin, D., Yu, P.S., Pan, S.: Graph neural networks for graphs with heterophily: A survey. arXiv preprint arXiv:2202.07082 (2022)
52. Zhu, G., Zhu, Z., Chen, H., Yuan, C., Huang, Y.: Hagnn: Hybrid aggregation for heterogeneous graph neural networks. arXiv preprint arXiv:2307.01636 (2023)
53. Zhu, G., Zhu, Z., Wang, W., Xu, Z., Yuan, C., Huang, Y.: Autoac: Towards automated attribute completion for heterogeneous graph neural network. In: ICDE. pp. 2808–2821 (2023)