Long-range Meta-path Search on Large-scale Heterogeneous Graphs

Chao Li[1,2,∗], Zijie Guo[1,3,∗], Qiuting He[1], Kun He[1,†]
[1]School of Computer Science and Technology, Huazhong University of Science and Technology
[2] China Mobile Information Technology Co.,Ltd.
[3] School of Computer Science, Fudan University
{chaoli,zijieguo,brooklet60}@hust.edu.cn

Abstract

Utilizing long-range dependency, a concept extensively studied in homogeneous graphs, remains underexplored in heterogeneous graphs, especially on large ones, posing two significant challenges: Reducing computational costs while maximizing effective information utilization in the presence of heterogeneity, and overcoming the over-smoothing issue in graph neural networks. To address this gap, we investigate the importance of different meta-paths and introduce an automatic framework for utilizing long-range dependency on heterogeneous graphs, denoted as *Long-range Meta-path Search through Progressive Sampling* (LMSPS). Specifically, we develop a search space with all meta-paths related to the target node type. By employing a progressive sampling algorithm, LMSPS dynamically shrinks the search space with hop-independent time complexity. Through a sampling evaluation strategy, LMSPS conducts a specialized and effective meta-path selection, leading to retraining with only effective meta-paths, thus mitigating costs and over-smoothing. Extensive experiments across diverse heterogeneous datasets validate LMSPS's capability in discovering effective long-range meta-paths, surpassing state-of-the-art methods. Our code is available at `https://github.com/JHL-HUST/LMSPS`.

1 Introduction

Heterogeneous graphs are widely used for modeling multiple types of entities and relationships in complex systems by various types of nodes and edges. For instance, the large-scale academic network, `OGBN-MAG`, contains multiple node types, *i.e.*, Paper (P), Author (A), Institution (I), and Field (F), as well as multiple edge types, such as Author $\xrightarrow{\text{writes}}$ Paper, Paper$\xrightarrow{\text{cites}}$Paper, Author $\xrightarrow{\text{is affiliated with}}$ Institution, and Paper$\xrightarrow{\text{has a topic of}}$Field. These elements can be combined to build higher-level semantic relations called meta-paths [44]. For example, APA is a 2-hop meta-path representing the co-author relationship, and PFPAPFP is a 6-hop meta-path related to long-range dependency.

Utilizing long-range dependency is essential for graph representation learning. For homogeneous graphs, many graph neural networks (GNNs) [28, 4, 1, 54] have been developed to gain benefit from long-range dependency. Utilizing long-range dependency is also crucial for heterogeneous graphs. For instance, the Internet movie database (`IMDB`) contains $21K$ nodes with only $87K$ edges. Such sparsity means each node has only a few directly connected neighbors and requires models to enhance the node embedding from long-range neighbors. The main challenge in using long-range dependency on heterogeneous graphs is how to alleviate costs while striving to effectively utilize information in exponentially increased receptive fields, which is much more challenging compared to homogeneous

---

[∗]The first two authors contribute equally.
[†]Corresponding author.

graphs due to the heterogeneity. In addition, the well-known over-smoothing issue [30, 24] occurring in many GNNs also needs to be addressed.

Heterogeneous Graph Neural Networks (HGNNs) are popular deep learning techniques for heterogeneous graph representation learning. The key idea is to aggregate valuable neighbor information based on a range of relations to enhance the semantics of vertices. Traditional HGNNs are typically classified into two categories: metapath-free methods and metapath-based methods. Most metapath-free HGNNs [63, 18, 21, 34] utilize information from l-hop neighborhoods by stacking l layers. However, on large-scale heterogeneous graphs, the number of nodes in receptive fields grows exponentially with the number of layers, making them hard to expand to large hops. A recent work [36] employs the graph transformer [54] to learn heterogeneous graphs, which can only exploit long-range dependency in small datasets due to the quadratic complexity of transformer [17].

Metapath-based HGNNs [48, 11, 23, 52] obtain information from l-hop neighborhoods by utilizing single-layer structures and meta-paths with maximum hop l, *i.e.*, all meta-paths no more than l hops. Benefiting from the feature of selective aggregation of neighbors based on meta-path, metapath-based HGNNs show greater potential in handling large-scale datasets [55, 52]. For metapath-based HGNNs, to exploit long-range dependency, the maximum hop needs to be large enough as there are no stacking layers. However, the number of meta-paths also grows exponentially with maximum hop l, corresponding to exponential receptive fields associated with layers in metapath-free methods. For instance, on `OGBN-MAG`, to gain a 3-hop meta-path based on the 2-hop meta-path PAP, the next node type has three choices (A, P, and F). For each additional hop, the number of possible meta-paths increases exponentially. Hence, utilizing long-range dependencies on large-scale heterogeneous graphs has not been resolved yet.

This paper investigates the importance of various meta-paths and makes two key observations: (1) *A small number of meta-paths dominate the performance*, and (2) *certain meta-paths can have a negative impact on performance*. The second observation explains why few HGNNs gain benefit from long-range neighbors. Different from homogeneous graphs, messages on heterogeneous graphs can be noisy or redundant for specific tasks, and the presence of long-range dependencies makes it more challenging to exclude negative heterogeneous information, even with the use of attention mechanisms. For example, focusing too much on Institution instead of related Papers is harmful to predicting a paper's field. These observations highlight the opportunity to leverage long-range dependencies by selectively utilizing effective meta-paths.

Motivated by the observations mentioned above, we propose a novel method called *Long-range Meta-path Search through Progressive Sampling* (LMSPS). LMSPS focuses on meta-path search and aims to reduce the exponentially growing meta-paths to a subset that is specifically effective for the given dataset and task. LMSPS builds a comprehensive search space initially, including all meta-paths related to the target nodes. Then, it adopts a progressive sampling algorithm to reduce the search space. Finally, LMSPS selects the top-M meta-paths from the reduced search space based on a sampling evaluation strategy. This search stage reduces the exponential number of meta-paths to a constant for retraining. Experimental results on both real-world and manual sparse large-scale datasets demonstrate that LMSPS outperforms existing state-of-the-art methods for heterogeneous graph representation learning.

Our main contributions are summarized as follows:

- We propose a novel meta-path search framework termed LMSPS, which to our knowledge is the first HGNNs to utilize long-range dependency in large-scale heterogeneous graphs.
- To search for effective meta-paths efficiently, we introduce a novel progressive sampling algorithm to reduce the search space dynamically and a sampling evaluation strategy for meta-path selection.
- Moreover, the searched meta-paths of LMSPS can be generalized to other HGNNs to boost their performance.

2   Preliminaries

Heterogeneous graph [43]. *A heterogeneous graph is defined as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, f_{\mathcal{T}}, f_{\mathcal{R}}\}$ with $|\mathcal{T}+\mathcal{R}|>2$, where $\mathcal{V}$ denotes the set of nodes, $\mathcal{E}$ denotes the set of edges, $\mathcal{T}$ is the node-type set and $\mathcal{R}$ the edge-type set. Each node $v_i \in \mathcal{V}$ is maped to a node type $f_{\mathcal{T}}(v_i) \in \mathcal{T}$ by mapping function $f_{\mathcal{T}} : \mathcal{V} \to \mathcal{T}$. Similarly, each edge $e_{t \leftarrow s} \in \mathcal{E}$ (e_{ts} for short) is mapped to an edge type $f_{\mathcal{R}}(e_{ts}) \in \mathcal{R}$ by mapping function $f_{\mathcal{R}} : \mathcal{E} \to \mathcal{R}$.*

Meta-path [44]. *A meta-path P is a composite relation that consists of multiple edge types,* i.e.*, $P \triangleq c_1 \xleftarrow{r_{12}} c_2 \cdots c_{l-1} \xleftarrow{r_{(l-1)l}} c_l$ ($P = c_1 \cdots c_l$ for short), where $c_1, \ldots, c_l \in \mathcal{T}$ and $r_{12}, \ldots, r_{(l-1)l} \in \mathcal{R}$.*

A meta-path P corresponds to multiple meta-path instances in the underlying heterogeneous graph. For example, meta-path APA corresponds to all paths of co-author relationships on the heterogeneous graph. Using meta-paths means selectively aggregating neighbors on the meta-path instances.

3   Related Works

Heterogeneous Graph Neural Networks. HGNNs are proposed to learn rich and diverse semantic information on heterogeneous graphs. Several HGNNs [29, 23, 34, 36] have involved high-order semantic aggregation. However, their methods are not applied to large-scale datasets due to high costs. Additionally, many HGNNs [56, 21, 48, 23] have implicitly learned meta-paths by attention. However, few work employs the discovered meta-paths to produce final results, let alone generalize them to other HGNNs to demonstrate their effectiveness. For example, GTN [56] and HGT [21] only list the discovered meta-paths. HAN [48], HPN [23] and MEGNN [5] validate the importance of discovered meta-paths by experiments not directly associated with the learning task. GraphMSE [31] is the only work that shows the performance of the discovered meta-paths. However, they are not as effective as the full meta-path set. So, their learned meta-paths are not effective enough.

Meta-structure Search on Heterogeneous Graphs. Recently, some works have attempted to utilize neural architecture search (NAS) to discover meta-structures. GEMS [15] is the first NAS method on heterogeneous graphs, which utilizes an evolutionary algorithm to search meta-graphs for recommendation tasks. DiffMG [8] searches for meta-graphs by a differentiable algorithm to conduct an efficient search. PMMM [27] performs a stable search to find meaningful meta-multigraphs. However, meta-path-based HGNNs are mainstream methods [41, 57, 48, 11], while meta-graph-based HGNNs are specialized. So, their searched meta-graphs are extremely difficult to generalize to other HGNNs. RL-HGNN [60] proposes a reinforcement learning (RL)-based method to find meta-paths. On recommendation tasks, RMS-HRec [37] also proposes an RL-based meta-path selection strategy to discover meta-paths. Both of them are very time-consuming.

4   Motivation of Long-range Meta-path Search

Long-range meta-paths can complete the missing information that can not be obtained from close nodes. Take the meta-path MDMDMK (M←D←M←D←M←K) from IMDB as an example. IMDB includes four different entity types: Movies (M), Directors (D), Keywords (K), and Actors (A). The task is to predict the category of the target movies. MDMDMK is a 5-hop meta-path that is hard for experts to understand and then apply. However, for many movies without keywords, the meta-path M←D←M←D←M←K is important because the target movies can aggregate the keyword information from the movies of co-directors.

In addition, SeHGNN [52] employs attention mechanisms to fuse all the target-node-related meta-paths and outperforms the existing HGNNs. SeHGNN has an important observation that models with single-layer structures and long meta-paths outperform those with multi-layers and short meta-paths, indicating the advantages of long-range meta-paths. However, because the number of meta-paths increases exponentially with maximum hops, SeHGNN has to use a small maximum hop to save memory and reduce costs. For example, The maximum hop is 2 for large-scale datasets `OGBN-MAG` [20], which is insufficient, as shown in Table 2 of the experiments. This inspires us to consider whether we can only use effective meta-paths instead of all meta-paths to reduce the consumption of large maximum hops.

We analyze the importance of different meta-paths for SeHGNN on two widely-used real-world datasets `DBLP` and `ACM` from HGB [34]. All results are the average of 10 times running with different random initialization. As the number of meta-paths exponentially increases with the maximum hop, in exploratory experiments, we set the maximum hop $l = 2$ for ease of illustration. Then the meta-path sets are {A, AP, APA, APT, APV} on `DBLP`, and {P, PA, PC, PP, PAP, PCP, PPA, PPC, PPP} on `ACM`.

In each experiment on `DBLP`, we remove one meta-path and compare the performance with the result of leveraging the full meta-path set to analyze the importance of the removed meta-path. As shown
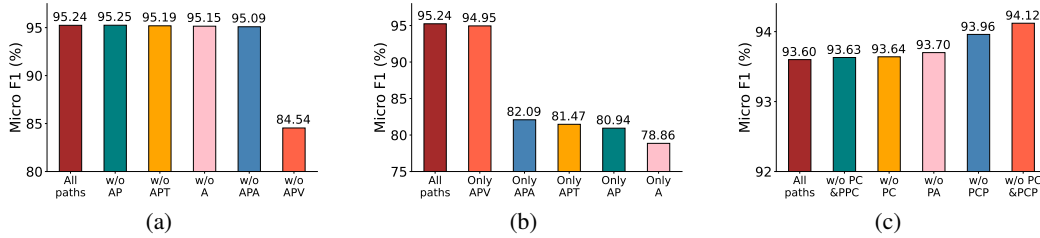
Figure 1: Analysis of the importance of different meta-paths. (a) illustrates the results after removing a single meta-path on DBLP; (b) shows the performance of utilizing a single meta-path on DBLP; (c) illustrates the performance after removing a part of meta-paths on ACM.

in Figure 1 (a), removing either A or AP or APA or APT has little impact on the final performance. However, removing APV results in severe degradation in performance, demonstrating APV is the critical meta-path on DBLP when $l = 2$. We further retain one meta-path and remove others as shown in Figure 1 (b). The performance of utilizing APV is only slightly degraded compared to the full meta-paths set. Consequently, we obtain the first observation: *A small number of meta-paths provide major contributions.* In each experiment on ACM, we remove some meta-paths to analyze their impact on the final performance. Results in Figure 1 (c) show that the performance of SeHGNN improves after removing a part of meta-paths. For example, after removing PC and PCP, the Micro-F1 scores are improved by 0.52%. So, we can conclude the second observation: *Certain meta-paths can have a negative impact for heterogeneous graphs.* The second observation is reasonable because heterogeneous information is not consistently beneficial for various tasks compared to homogeneous information. It is supported by the fact that various recent HGNNs [8, 27, 52] have removed some edge types to exclude corresponding heterogeneous information during pre-processing based on substantial domain expertise or empirical observations. This observation explains why most HGNNs use a maximum hop of 2, *i.e.*, it is hard to exclude negative information under larger maximum hops. Additionally, because SeHGNN employs an attention mechanism, the performance degradation indicates the attention mechanism has limitations in dealing with noise.

Although long-range meta-paths outperform short meta-paths [52], they need a large maximum hop, resulting in exponentially increasing meta-paths. Motivated by the above two observations, unlike existing methods [5, 31, 52, 56], we can employ effective meta-paths instead of the full meta-path set without sacrificing performance. Although the number of meta-paths grows exponentially with the maximum hop, the proportion of effective meta-paths is small, which is similar to the $80/20$ rule of Pareto principle [40, 10]. To keep efficiency while trading off the performance, we choose a fixed number of meta-paths (like 30) over all datasets. Therefore, the exponential meta-paths can be reduced to a constant. Now the key point is how to find effective meta-paths.

5 The Proposed Method

The key of our LMSPS is to utilize a search stage to reduce the exponentially increased meta-paths to a subset effective to the current dataset and task. It can overcome the main challenges of utilizing long-range dependency on heterogeneous graphs. First, utilizing effective meta-paths instead of all meta-paths alleviates computational and memory costs while keeping effective heterogeneous information. Second, each target node only aggregates neighbors on the path instances of effective meta-paths. Because the proportion of effective meta-paths is small and each node has a different neighborhood condition, each target node aggregates different neighbors under the constraints of effective meta-path instances. In this way, the over-smoothing issue can also be overcome.

Then, the main challenge becomes how to discover effective meta-paths, especially long-range ones. Searching for long-range meta-paths has two main challenges: the exponentially increasing issue and the noise issue. We propose a progressive sampling algorithm and a sampling evaluation strategy to respectively overcome the two challenges. Figure 2 illustrates the overall framework of LMSPS, which consists of a super-net in the search stage and a target-net in the training stage.

The super-net aims to automatically discover effective meta-paths for specific datasets or tasks, so the search results should not be affected by specific modules. Based on this consideration, we develop a simple MLP-based instead of transformer-based architecture for meta-path search because the
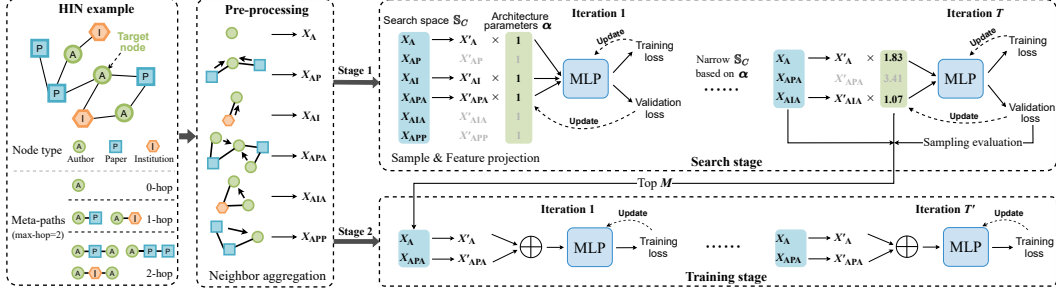
Figure 2: The overall framework of LMSPS. Based on the progressive sampling and sampling evaluation in the search stage, the training stage employs M effective meta-paths instead of the full K target-node-related meta-paths. It exhibits aggregation of meta-paths with maximum hop 2, *i.e.*, 0, 1, and 2-hop meta-paths. The weight updates of feature projection are not shown for ease of illustration.

former involves fewer human interventions [45, 16, 3]. In addition, the sampling strategy keeps the parametric modules changing in the search stage, which is important for preventing the search meta-paths from being affected by specific modules. The super-net contains five blocks: neighbor aggregation, feature projection, progressive sampling search, sampling evaluation, and MLP.

5.1 Progressive Sampling Search

Let $\mathbb{P} = \{P_1, \cdots, P_k, \cdots, P_K\}$ be the initial search space with all the K target-node-related meta-paths, $\boldsymbol{X}^{c_i}$ be the raw feature matrix of all nodes belonging to type c_i, and $\hat{\boldsymbol{A}}_{c_i,c_{i+1}}$ be the row-normalized adjacency matrix between node type c_i and c_{i+1}. The neighbor aggregation block follows SeHGNN [52], which employs an efficient one-time message passing to pre-process an entire heterogeneous graph into regular-shaped tensors for target nodes. Specifically, it uses the multiplication of adjacency matrices to calculate the final contribution weight of each metapath-based neighbor to targets. As shown in Figure 2, the neighbor aggregation process of l-hop meta-path $P_k = c_0 c_1 c_2 \ldots c_l$ is:

$$\boldsymbol{X}_k = \begin{cases} \boldsymbol{X}^{c_0} & l = 0 \\ \hat{\boldsymbol{A}}_{c_0,c_1} \hat{\boldsymbol{A}}_{c_1,c_2} \cdots \hat{\boldsymbol{A}}_{c_{l-1},c_l} \boldsymbol{X}^{c_l} & l > 0 \end{cases}, \tag{1}$$

where $\boldsymbol{X}_k$ is the feature matrices of meta-path P_k, and c_0 is the target node type. Then, an MLP-based feature projection block is used to project different feature matrices into the same dimension, namely, $\boldsymbol{X}'_k = \text{MLP}_k(\boldsymbol{X}_k)$.

To automatically discover meaningful meta-paths for various datasets or tasks without prior, our search space contains all target-node-related meta-paths, severely challenging the efficiency and effectiveness of the search. To address the efficiency challenge, LMSPS utilizes a progressive sampling algorithm to sample meta-paths in each iteration and progressively shrink the search space.

Specifically, LMSPS assigns one architecture parameter to each meta-path. Let $\boldsymbol{\alpha} = \{\alpha_1, \cdots, \alpha_k, \cdots, \alpha_K\} \in \mathbb{R}^K$ be the architecture parameters of K meta-paths. We use a Gumbel-softmax [35, 9] over architecture parameters α_k to calculate the strength of different meta-paths:

$$q_k = \frac{\exp\left[(\alpha_k + u_k)/\tau\right]}{\sum_{j=1}^{K} \exp\left[(\alpha_j + u_j)/\tau\right]}, \tag{2}$$

where q_k is the path strength, representing the relative importance of meta path P_k. $u_k = -\log(-\log(U))$ where $U \sim \text{Uniform}(0, 1)$, and τ is temperature controlling relaxation's extent.

The progressive sampling algorithm uses the path strength to progressively narrow the search space from K to V to exclude useless meta-paths. Generally, $K \gg V$ under a large maximum hop. Let $\widetilde{q}_C$ be the C-th largest path strength of $\mathbb{Q} = \{q_1, \cdots, q_k, \cdots, q_K\}$. During the search stage, the search space retains all meta-paths no less than $\widetilde{q}_C$. The dynamic search space can be formulated as follows:

$$\mathbb{S}_C = \{k | q_k \geq \widetilde{q}_C, \forall 1 \leq k \leq K\} \quad \text{where} \quad C = \lceil \lambda(K-V) \rceil + V. \tag{3}$$

Here C is the search space size, $\mathbb{S}_C$ consists of the indexes of retained meta-paths, and $\lceil \cdot \rceil$ indicates the rounding symbol. $\lambda \in [0, 1]$ is a parameter controlling the number of retrained meta-paths and decreases from 1 to 0 as the epoch increases.

5

As the search stage aims to determine top-M meta-paths, we sample M meta-paths from dynamic search space in each iteration. In each iteration, only parameters on the M activated meta-paths are updated, while others remain unchanged. Therefore, the search cost is relevant to M instead of K. The forward propagation can be expressed as:

$$\boldsymbol{Z} = \text{MLP}\Big(\sum_{k \in \mathbb{S}} q'_k \cdot \text{MLP}_k(\boldsymbol{X}_k) \Big) \quad \text{where } \mathbb{S} = \text{UniformSample}(\mathbb{S}_C, M). \tag{4}$$

Here $q'_k = \exp\left[(\alpha_k + u_k)/\tau\right] / \sum_{k \in \mathbb{S}} \exp\left[(\alpha_j + u_j)/\tau\right]$ indicates the path strength of activated meta-paths, and $\text{UniformSample}(\mathbb{S}_C, M)$ indicates a set of M elements chosen randomly from set $\mathbb{S}_C$ without replacement via a uniform distribution.

The parameter update in the super-net involves a bilevel optimization problem [2, 7, 51].

$$\min_{\boldsymbol{\alpha}} \ \mathcal{L}_{val}(\mathbb{S}, \boldsymbol{\omega}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \quad \text{s.t. } \boldsymbol{\omega}^*(\boldsymbol{\alpha}) = \text{argmin}_{\boldsymbol{\omega}} \, \mathcal{L}_{train}(\mathbb{S}, \boldsymbol{\omega}, \boldsymbol{\alpha}). \tag{5}$$

Here $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ denote the training and validation loss, respectively. $\boldsymbol{\alpha}$ is the architecture parameters calculating the path strength. $\boldsymbol{\omega}$ is the network weights in MLP. Following the NAS-related works in the computer vision field [33, 50, 53], we address this issue by first-order approximation. Specifically, we alternatively freeze architecture parameters $\boldsymbol{\alpha}$ when training $\boldsymbol{\omega}$ on the training set and freeze $\boldsymbol{\omega}$ when training $\boldsymbol{\alpha}$ on the validation set.

The progressive sampling strategy can contribute to a more compact search space specifically driven by the current HIN and task, leading to a more effective meta-path discovery. Additionally, it can overcome the deep coupling issue [13] because of the randomness in each iteration.

5.2 Sampling Evaluation

After the completion of the progressive sampling search, the search space is narrowed from K to V. Traditional methods in the computer vision field directly derive the final architecture based on the architecture parameters [33, 50, 53]. However, as different meta-paths could be noisy or redundant to each other, top-M meta-paths are not necessarily the optimal solution when their importance is calculated independently. Based on this consideration, we specially designed a novel sampling evaluation strategy by evaluating the overall performance of each meta-path set. Specifically, using path strength at the end of progressive sampling as the probability, we sample M meta-paths from the compact search space $\mathbb{S}_V$ to evaluate their overall performance. The sampling evaluation is repeated 200 times to filter out the meta-path set with the lowest validation loss. So, we can select the best meta-path set instead of independent top-M meta-paths, which is more reasonable. This stage is not time-consuming because the evaluation does not involve weight training. This sampling process can be represented as:

$$\bar{\mathbb{S}} = \text{DiscreteSample}(\mathbb{S}_V, M, \bar{\mathbb{Q}}). \tag{6}$$

Here, $\text{DiscreteSample}(\mathbb{S}_V, M, \bar{\mathbb{Q}})$ indicates a set of M elements chosen from set $\mathbb{S}_V$ without replacement via discrete probability distribution $\bar{\mathbb{Q}}$. $\bar{\mathbb{Q}}$ is the set of relative path strength calculated by architecture parameters of the V meta-paths based on Equation 2. The overall search algorithm and more discussion are shown in Appendix C.

Thereafter, the retained meta-path set is recorded as $\mathbb{S}_M = \text{argmin}_{\bar{\mathbb{S}}} \mathcal{L}_{val}(\bar{\mathbb{S}}, \boldsymbol{\omega}^*, \boldsymbol{\alpha}^*)$. The forward propagation of the target-net for representation learning can be formulated as:

$$\hat{\boldsymbol{Z}} = \text{MLP}\Big(\underset{k \in \mathbb{S}_M}{\big\|} \text{MLP}_k(\boldsymbol{X}_k) \Big). \tag{7}$$

Here, $\|$ denotes the concatenation operation. Unlike existing HGNNs, the architecture of the target-net does not contain neighbor attention and semantic attention. Instead, the parametric modules consist of pure MLPs. The training objective is shown in Appendix A.3.

5.3 Discussion on Differences with Prior Works

As discussed in Section 3, there have been some attempts to find meta-paths automatedly [15, 56, 60]. Three aspects highlight the differences between LMSPS and existing methods. 1) Large-scale dataset: LMSPS is the first HGNN that makes it possible to achieve automated meta-path selection for large-scale heterogeneous graph node property prediction. 2) Long-range dependency: LMSPS is

Table 1: Performance on small and large datasets. Best is in bold, and the runner-up is underlined.

Method	DBLP		IMDB		ACM		Freebase		OGBN-MAG*
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Test Acc.
MLP [19]	-	-	-	-	-	-	-	-	26.92 ± 0.26
GraphSAGE [14]	-	-	-	-	-	-	-	-	46.78 ± 0.67
RGCN [41]	91.52 ± 0.50	92.07 ± 0.50	58.85 ± 0.26	62.05 ± 0.15	91.55 ± 0.74	91.41 ± 0.75	46.78 ± 0.77	58.33 ± 1.57	47.37 ± 0.48
HAN [48]	91.67 ± 0.49	92.05 ± 0.62	57.74 ± 0.96	64.63 ± 0.58	90.89 ± 0.43	90.79 ± 0.43	21.31 ± 1.68	54.77 ± 1.40	*OOM*
GTN [56]	93.52 ± 0.55	93.97 ± 0.54	60.47 ± 0.98	65.14 ± 0.45	91.31 ± 0.70	91.20 ± 0.71	*OOM*	*OOM*	*OOM*
RSHN [63]	93.34 ± 0.58	93.81 ± 0.55	59.85 ± 3.21	64.22 ± 1.03	90.50 ± 1.51	90.32 ± 1.54	*OOM*	*OOM*	*OOM*
HetGNN [57]	91.76 ± 0.43	92.33 ± 0.41	48.25 ± 0.67	51.16 ± 0.65	85.91 ± 0.25	86.05 ± 0.25	*OOM*	*OOM*	*OOM*
MAGNN [11]	93.28 ± 0.51	93.76 ± 0.45	56.49 ± 3.20	64.67 ± 1.67	90.88 ± 0.64	90.77 ± 0.65	*OOM*	*OOM*	*OOM*
HetSANN [18]	78.55 ± 2.42	80.56 ± 1.50	49.47 ± 1.21	57.68 ± 0.44	90.02 ± 0.35	89.91 ± 0.37	*OOM*	*OOM*	*OOM*
GCN [26]	90.84 ± 0.32	91.47 ± 0.34	57.88 ± 1.18	64.82 ± 0.64	92.17 ± 0.24	92.12 ± 0.23	27.84 ± 3.13	60.23 ± 0.92	*OOM*
GAT [46]	93.83 ± 0.27	93.39 ± 0.30	58.94 ± 1.35	64.86 ± 0.43	92.26 ± 0.94	92.19 ± 0.93	40.74 ± 2.58	65.26 ± 0.80	*OOM*
Simple-HGN [34]	94.01 ± 0.24	94.46 ± 0.22	63.53 ± 1.36	67.36 ± 0.57	93.42 ± 0.44	93.35 ± 0.45	47.72 ± 1.48	66.29 ± 0.45	*OOM*
HGT [21]	93.01 ± 0.23	93.49 ± 0.25	63.00 ± 1.19	67.20 ± 0.57	91.12 ± 0.76	91.00 ± 0.76	29.28 ± 2.52	60.51 ± 1.16	46.78 ± 0.42
GraphMSE [31]	94.08 ± 0.14	94.44 ± 0.13	57.60 ± 2.13	62.37 ± 1.03	92.58 ± 0.50	92.54 ± 0.14	*OOM*	*OOM*	*OOM*
DiffMG [8]	94.01 ± 0.37	94.20 ± 0.36	58.09 ± 1.35	59.75 ± 1.23	88.16 ± 2.83	88.07 ± 3.04	*OOM*	*OOM*	*OOM*
Random	93.57 ± 0.64	93.84 ± 0.53	52.13 ± 0.74	53.83 ± 0.66	90.91 ± 1.02	90.82 ± 0.93	21.22 ± 2.58	37.54 ± 2.66	35.14 ± 3.78
NARS [55]	94.18 ± 0.47	94.61 ± 0.42	63.51 ± 0.68	66.18 ± 0.70	93.36 ± 0.32	93.31 ± 0.33	49.98 ± 1.77	63.26 ± 1.26	50.66 ± 0.22
space4HGNN [59]	94.24 ± 0.42	94.63 ± 0.40	61.57 ± 1.19	63.96 ± 0.43	92.50 ± 0.14	92.38 ± 0.10	41.37 ± 4.49	65.66 ± 4.94	*OOM*
PMMM [27]	94.82 ± 0.26	95.14 ± 0.22	65.81 ± 0.29	67.58 ± 0.22	93.78 ± 0.25	93.71 ± 0.17	*OOM*	*OOM*	*OOM*
HINormer [36]	94.57 ± 0.23	94.94 ± 0.21	64.65 ± 0.53	67.83 ± 0.34	93.91 ± 0.42	93.83 ± 0.45	<u>52.18 ± 0.39</u>	64.92 ± 0.43	*OOM*
SeHGNN [52]	94.86 ± 0.14	95.24 ± 0.13	<u>66.63 ± 0.34</u>	68.21 ± 0.32	93.95 ± 0.48	93.87 ± 0.50	50.71 ± 0.44	63.41 ± 0.47	<u>51.45 ± 0.29</u>
SlotGAT [62]	<u>94.95 ± 0.20</u>	<u>95.31 ± 0.19</u>	64.05 ± 0.60	<u>68.64 ± 0.33</u>	<u>93.99 ± 0.23</u>	<u>94.06 ± 0.22</u>	49.68 ± 1.97	**66.83 ± 0.30**	*OOM*
LMSPS (ours)	**95.35 ± 0.22**	**95.66 ± 0.20**	**66.99 ± 0.32**	**68.70 ± 0.26**	**94.73 ± 0.41**	**94.69 ± 0.36**	**53.26 ± 0.47**	<u>66.09 ± 0.51</u>	**54.83 ± 0.20**

   * OGBN-MAG is a large dataset with nodes' numbers 10 to 175 times that of the other four datasets.

the first HGNN to utilize long-range dependency in large-scale heterogeneous graphs. To achieve the above two goals, LMSPS has addressed two key challenges: (a) alleviating costs while striving to effectively utilize information in exponentially increased receptive fields and (b) overcoming the well-known over-smoothing issue. 3) High generalization: Based on Table 4, the searched meta-paths of LMSPS can be generalized to other HGNNs to boost their performance, which has not been achieved by existing works. To accomplish this objective, LMSPS uses an MLP-based architecture instead of a transformer-based one for meta-path search because the former involves fewer inductive biases [45, 16, 3], i.e., human interventions.

6 Experiments and Analysis

This section evaluates the benefits of our method against state-of-the-art models on nine heterogeneous datasets. We aim to answer the following questions: **Q1.** How does LMSPS perform compared with state-of-the-art baselines? **Q2.** Can LMSPS overcome the over-smoothing and noise issues? **Q3.** Are the search algorithm and searched meta-paths effective? **Q4.** Does LMSPS perform better on sparser heterogeneous graphs?

6.1 Datasets and Baselines

We evaluate LMSPS on several representative heterogeneous graph datasets, including `DBLP`, `IMDB`, `ACM`, and `Freebase` from HGB benchmark [34], and the large-scale dataset `OGBN-MAG` from OGB challenge [20]. The statistics of the datasets are listed in Table 7. The details about all datasets, baselines, and experiment settings are recorded in Appendix A and B.

6.2 Performance Analysis

To answer **Q1**, we report the performance of our approaches and baselines in Table 1. *Random* means the result of replacing our searched meta-paths with 30 random meta-paths. We show the average result of 20 random samples. Based on the results, we have the following observations. First, LMSPS outperforms all baselines for different metrics on almost all datasets except Micro-F1 scores on Freebase, sometimes by a significant margin, which validates the power of LMSPS. For example, on the largest dataset `OGBN-MAG`, LMSPS achieves 54.83% test accuracy, while the best competitor has 51.45% test accuracy. Second, all metapath-free methods encounter out-of-memory (OOM) issues when dealing with large datasets, including highly competitive methods, HINormer and SlotGAT, indicating the advantage of employing meta-paths electively aggregating neighbors on the meta-path instances. Third, although LMSPS is an MLP-based method, the pure MLP method [19] has the worst performance with only 26.92% test accuracy on `OGBN-MAG`, validating the advantages of pre-processing and meta-path search. Finally, LMSPS outperforms *Random* significantly, further indicating the importance of meta-path search. More comparisons with top method combinations

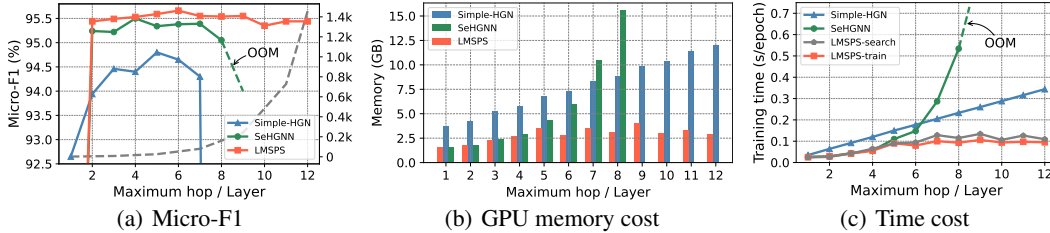|	(a) Micro-F1	(b) GPU memory cost	(c) Time cost

Figure 3: Illustration of (a) performance, (b) memory cost, (c) average training time of Simple-HGN, SeHGNN, and LMSPS relative to the maximum hop or layer on DBLP. The *gray dotted line* in (a) indicates the number of target-node-related meta-paths under different maximum hops, which is exponential.

Table 2: Experiments on OGBN-MAG to analyze the performance of SeHGNN and LMSPS under different maximum hops. #MP is the number of meta-paths under different maximum hops.

Max hop	#MP	SeHGNN		LMSPS	
		Time	Test accuracy	Time	Test accuracy
1	4	4.35	47.18 ± 0.28	3.98	46.88 ± 0.10
2	10	6.44	51.79 ± 0.24	5.63	51.91 ± 0.13
3	23	11.28	52.44 ± 0.16	10.02	52.72 ± 0.24
4	50	OOM	OOM	14.34	53.43 ± 0.18
5	107	OOM	OOM	14.77	53.90 ± 0.19
6	226	OOM	OOM	14.71	$\mathbf{54.83 \pm 0.20}$

Table 3: Experiments to explore the effectiveness of our search algorithm. In our LMSPS, the meta-paths are replaced by those discovered by other methods.

Method	DBLP	IMDB	ACM	Freebase
HAN	95.44 ± 0.14	65.95 ± 0.31	90.66 ± 0.30	-
GTN	95.33 ± 0.05	65.99 ± 0.16	90.66 ± 0.30	-
DARTS	95.35 ± 0.17	66.23 ± 0.14	93.45 ± 0.13	63.25 ± 0.42
SPOS	95.41 ± 0.43	67.10 ± 0.29	93.64 ± 0.37	64.02 ± 0.62
DiffMG	95.45 ± 0.49	66.98 ± 0.37	93.61 ± 0.45	64.56 ± 0.78
PMMM	95.48 ± 0.27	67.49 ± 0.24	93.74 ± 0.22	64.83 ± 0.46
LMSPS	$\mathbf{95.66 \pm 0.20}$	$\mathbf{68.70 \pm 0.26}$	$\mathbf{94.69 \pm 0.36}$	$\mathbf{66.09 \pm 0.51}$

on OGBN-MAG are shown in Table 10 of Appendix E.2, in which LMSPS also shows the best performance.

6.3 Analysis on Large Maximum Hops

To answer **Q2**, we conduct experiments to compare the performance, memory, and efficiency of LMSPS with the method of HGB benchmark, Simple-HGN [34], and the best metapath-based method, SeHGNN [52], on DBLP. Figure 3 (a) shows that LMSPS has consistent performance with the increment of maximum hop. The failure of Simple-HGN demonstrates its attention mechanism can not overcome the over-smoothing issue or eliminate the effects of noise under large hop. Figure 3 (b), (c) illustrate each training epoch's average memory and time costs relative to the maximum hop or layer. We can observe that the consumption of SeHGNN exponentially grows, and the consumption of Simple-HGN linearly increases, which is consistent with their time complexity as listed in Table 8. Meanwhile, LMSPS has almost constant consumption as the maximum hop grows. Figure 3 (c) shows the time cost of LMSPS in the search stage, which also approximates a constant when the number of meta-paths is larger than $M = 30$. More results about efficiency are shown in Figure 4 of Appendix E.1.

We also compare the performance and training time of LMSPS with SeHGNN under different maximum hops on large-scale dataset OGBN-MAG. When the maximum hop $l = 1, 2, 3$, we utilize the full meta-path set because the number of meta-paths is smaller than $M = 30$. Following the convention [34, 52], we measure the average time consumption of one epoch for each model. As shown in Table 2, the performance of LMSPS keeps increasing as the maximum hop value grows. It indicates that LMSPS can overcome the issues caused by utilizing long-range dependency, *e.g.*, over-smoothing and noise. In addition, when the number of meta-paths is larger than 30, the training time of LMSPS is stable under different maximum hops.

6.4 Effectiveness of the Search Algorithm and Searched Meta-paths

To answer **Q3**, we first explore the effectiveness of our search algorithm. In our architecture, our meta-paths are replaced by those meta-paths discovered by other methods. DARTS [32] is the first differentiable search algorithm in neural networks. SPOS [13] is a classic singe-path differentiable algorithm. DARTS and SPOS aim to search operations, like 3×3 convolution, in CNNs. DiffMG [8] and PMMM [27] search for meta-graphs instead of meta-paths. We ignore these differences and focus on the algorithms. The derivation strategies of the four methods are unsuitable for discovering

Table 4: Experiments on the generalization of the searched meta paths. * means using the meta-paths searched in LMSPS.

Method	DBLP	IMDB	ACM	Freebase
HAN	92.05 ± 0.62	64.63 ± 0.58	90.79 ± 0.43	54.77 ± 1.40
HAN*	93.54 ± 0.15	65.89 ± 0.52	92.28 ± 0.47	57.13 ± 0.72
SeHGNN	95.24 ± 0.13	68.21 ± 0.32	93.87 ± 0.50	63.41 ± 0.47
SeHGNN*	95.57 ± 0.23	68.59 ± 0.24	94.46 ± 0.18	65.37 ± 0.42

Table 5: Results of LMSPS and SeHGNN on the sparse large-scale heterogeneous graphs. ↑ means the improvements in test accuracy.

Dataset	SeHGNN	LMSPS	↑
OGBN-MAG-5	36.04 ± 0.64	40.82 ± 0.42	**4.78**
OGBN-MAG-10	38.27 ± 0.19	42.30 ± 0.23	**4.03**
OGBN-MAG-20	39.18 ± 0.09	42.65 ± 0.17	**3.47**
OGBN-MAG-50	39.50 ± 0.13	42.82 ± 0.16	**3.32**

Table 6: Experiments on small and large datasets to analyze the effects of different blocks in LMSPS. *PS* means progressive sampling strategy, and *SE* means sampling evaluation strategy. † means employing all meta-paths and replacing the concatenation operation with the transformer module.

Method	DBLP		IMDB		ACM		Freebase		OGBN-MAG
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Test Acc.
LMSPS w/o *PS*	94.71 ± 0.23	95.00 ± 0.19	64.85 ± 0.46	66.52 ± 0.37	93.19 ± 0.34	93.14 ± 0.41	48.89 ± 0.47	61.61 ± 0.51	47.66 ± 0.45
LMSPS w/o *SE*	95.15 ± 0.28	95.48 ± 0.24	65.46 ± 0.48	67.13 ± 0.47	94.20 ± 0.35	94.15 ± 0.31	52.08 ± 0.33	64.84 ± 0.38	52.94 ± 0.34
LMSPS †	95.06 ± 0.24	95.38 ± 0.21	66.85 ± 0.37	68.58 ± 0.34	94.60 ± 0.42	94.57 ± 0.39	*OOM*	*OOM*	*OOM*
LMSPS	$\mathbf{95.35 \pm 0.22}$	$\mathbf{95.66 \pm 0.20}$	$\mathbf{66.99 \pm 0.32}$	$\mathbf{68.70 \pm 0.26}$	$\mathbf{94.73 \pm 0.41}$	$\mathbf{94.69 \pm 0.36}$	$\mathbf{53.26 \pm 0.47}$	$\mathbf{66.09 \pm 0.51}$	$\mathbf{54.83 \pm 0.20}$

multiple meta-paths. So we changed their derivation strategies to ours to improve their performance. we report the Micro-F1 scores in Table 3. The performance of LMSPS verifies the effectiveness of our search algorithm.

To demonstrate the effectiveness of searched meta-paths, on the one hand, the meta-paths should be effective in the proposed model. On the other hand, the effective meta-paths mainly depend on the dataset instead of the architecture, so the meta-paths should be effective after being generalized to other HGNNs. Because finding meta-paths that work effectively across various HGNNs is a tough task, it has not been achieved by previous works. We verify the generalization of our searched meta-path on the most famous HGNN, HAN [48], and the SOTA metapath-based method SeHGNN [52]. The Micro-F1 scores on three representative datasets are shown in Table 4. After simply replacing the original meta-path set with our searched meta-paths and keeping other settings unchanged, the performance both improve, demonstrating the effectiveness of our searched meta-paths. In addition, we have shown the interpretability of searched meta-paths in Appendix D.

6.5 Necessity of Long-range Dependency

To answer **Q4** and explore the necessity of long-range dependency on heterogeneous graphs, we construct four large-scale datasets with high sparsity based on `OGBN-MAG`. To avoid inappropriate preference seed settings of randomly removing, we construct fixed heterogeneous graphs by limiting the maximum in-degree related to edge type. Specifically, we gradually reduce the maximum in-degree related to edge type in `OGBN-MAG` from 50 to 5 but leave all nodes unchanged. Details of the four datasets are listed in Appendix A. The test accuracy of LMSPS and SOTA method SeHGNN are shown in Table 5. LMSPS outperforms SeHGNN more significantly with the increasing sparsity. In addition, the leading gap of LMSPS over SeHGNN is more than 4.7% on the highly sparse dataset `OGBN-MAG-5`. The main difference between SeHGNN and LMSPS is that the former cannot utilize large hops and only use hop 2 while the latter has a maximum hop of 6, demonstrating that long-range dependencies are more effective with decreased direct neighbors on heterogeneous graphs.

6.6 Ablation Study

Two components differentiate our LMSPS from other HGNNs: the search algorithm and semantic fusion without attention. The search algorithm consists of a progressive sampling algorithm and a sampling evaluation strategy. We explore how each of them improves performance through ablation studies under the same settings as the main experiments in Table 1. As shown in Table 6, the performance of LMSPS significantly decreases when removing progressive sampling or sampling evaluation strategy. In addition, employing a transformer block for semantic attention on all meta-paths shows slightly worse performance even if using many more meta-paths and is out-of-memory on larger datasets, indicating that the transformer cannot eliminate the negative effects of noise. It is reasonable because attention values are calculated based on softmax and are positive even for noise.

7 Conclusion

This work presented a novel approach, the Long-range Meta-path Search through Progressive Sampling (LMSPS), to tackle the challenges of leveraging long-range dependencies in large-scale heterogeneous graphs, *i.e.*, reducing computational costs while effectively utilizing information and addressing the over-smoothing issue. Based on our two observations, *i.e.*, a few meta-paths dominate the performance, and certain meta-paths can have negative impact on performance, LMSPS introduced a progressive sampling search algorithm and a sampling evaluation strategy to automatically identify effective meta-paths, hence reducing the exponentially growing number of meta-paths to a manageable constant. Extensive experiments demonstrated the superiority of LMSPS over existing methods, particularly on sparse heterogeneous graphs that require long-range dependencies. By employing simple MLPs and complex meta-paths, LMSPS offers a novel direction that emphasizes data-dependent semantic relationships rather than relying solely on sophisticated neural architectures. The reproducibility and limitations are discussed in Appendix F and Appendix G, respectively.

Acknowledgments

References

[1] U. Alon and E. Yahav. On the bottleneck of graph neural networks and its practical implications. In *9th International Conference on Learning Representations, ICLR*, 2021.

[2] G. Anandalingam and T. L. Friesz. Hierarchical Optimization: An Introduction. *Annals of Operations Research*, pages 1–11, 1992.

[3] G. Bachmann, S. Anagnostidis, and T. Hofmann. Scaling mlps: A tale of inductive bias. In *Advances in Neural Information Processing Systems, NeurIPS*, 2023.

[4] F. M. Bianchi, D. Grattarola, and C. Alippi. Spectral clustering with graph neural networks for graph pooling. In *International conference on machine learning, ICML*, pages 874–883. PMLR, 2020.

[5] Y. Chang, C. Chen, W. Hu, Z. Zheng, X. Zhou, and S. Chen. MEGNN: Meta-path Extracted Graph Neural Network for Heterogeneous Graph Representation Learning. *Knowledge-Based Systems*, page 107611, 2022.

[6] W. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C. Hsieh. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *KDD '19: In Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 257–266, 2019.

[7] B. Colson, P. Marcotte, and G. Savard. An Overview of Bilevel Optimization. *Annals of Operations Research*, pages 235–256, 2007.

[8] Y. Ding, Q. Yao, H. Zhao, and T. Zhang. DiffMG: Differentiable Meta Graph Search for Heterogeneous Graph Neural Networks. In *KDD '21: In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 279–288, 2021.

[9] X. Dong and Y. Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, ICCV*, pages 1761–1770, 2019.

[10] P. Erridge. The pareto principle. *British Dental Journal*, 201(7):419–419, 2006.

[11] X. Fu, J. Zhang, Z. Meng, and I. King. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *Proceedings of the ACM Web Conference, WWW*, pages 2331–2341, 2020.

[12] X. Glorot and Y. Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 249–256, 2010.

[13] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun. Single Path One-Shot Neural Architecture Search with Uniform Sampling. In *European Conference on Computer Vision, ECCV*, pages 544–560, 2020.

[14] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems, NeurIPS*, 30, 2017.

[15] Z. Han, F. Xu, J. Shi, Y. Shang, H. Ma, P. Hui, and Y. Li. Genetic Meta-Structure Search for Recommendation on Heterogeneous Information Network. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management*, pages 455–464, 2020.

[16] X. He, B. Hooi, T. Laurent, A. Perold, Y. LeCun, and X. Bresson. A generalization of vit/mlp-mixer to graphs. In *International Conference on Machine Learning, ICML*, volume 202, pages 12724–12745, 2023.

[17] X. He, B. Hooi, T. Laurent, A. Perold, Y. LeCun, and X. Bresson. A generalization of vit/mlp-mixer to graphs. In *International conference on machine learning, ICML*, pages 12724–12745. PMLR, 2023.

[18] H. Hong, H. Guo, Y. Lin, X. Yang, Z. Li, and J. Ye. An Attention-based Graph Neural Network for Heterogeneous Structural Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, pages 4132–4139, 2020.

[19] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems, NeurIPS*, 33:22118–22133, 2020.

[20] W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong, and J. Leskovec. OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[21] Z. Hu, Y. Dong, K. Wang, and Y. Sun. Heterogeneous Graph Transformer. In *Proceedings of the ACM Web Conference, WWW*, pages 2704–2710, 2020.

[22] Q. Huang, H. He, A. Singh, S. Lim, and A. R. Benson. Combining label propagation and simple models out-performs graph neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[23] H. Ji, X. Wang, C. Shi, B. Wang, and S. Y. Philip. Heterogeneous Graph Propagation Network. *IEEE Transactions on Knowledge and Data Engineering*, pages 521–532, 2021.

[24] N. Keriven. Not too little, not too much: a theoretical analysis of graph (over) smoothing. *Advances in Neural Information Processing Systems, NeurIPS*, 35:2268–2281, 2022.

[25] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In Y. Bengio and Y. LeCun, editors, *International Conference on Learning Representations, ICLR*, 2015.

[26] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representation, ICLR*, 2017.

[27] C. Li, H. Xu, and K. He. Differentiable Meta Multigraph Search with Partial Message Propagation on Heterogeneous Information Networks. *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, 2023.

[28] G. Li, M. Muller, A. Thabet, and B. Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision, ICCV*, pages 9267–9276, 2019.

[29] J. Li, H. Peng, Y. Cao, Y. Dou, H. Zhang, S. Y. Philip, and L. He. Higher-order Attribute-enhancing Heterogeneous Graph Neural Networks. *IEEE Transactions on Knowledge and Data Engineering*, pages 560–574, 2021.

[30] Q. Li, Z. Han, and X.-M. Wu. Deeper Insights Into Graph Convolutional Networks for Semi-supervised Learning. In *Proceedings of the AAAI conference on artificial intelligence, AAAI*, 2018.

[31] Y. Li, Y. Jin, G. Song, Z. Zhu, C. Shi, and Y. Wang. GraphMSE: Efficient Meta-path Selection in Semantically Aligned Feature Space for Graph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, pages 4206–4214, 2021.

[32] H. Liu, K. Simonyan, and Y. Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR*, 2019.

[33] H. Liu, K. Simonyan, and Y. Yang. DARTS: Architecture Search. In *7th International Conference on Learning Representations, ICLR*, 2019.

[34] Q. Lv, M. Ding, Q. Liu, Y. Chen, W. Feng, S. He, C. Zhou, J. Jiang, Y. Dong, and J. Tang. Are We Really Making Much Progress? Revisiting, Benchmarking and Refining Heterogeneous Graph Neural Networks. In *KDD '21: In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1150–1160, 2021.

[35] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[36] Q. Mao, Z. Liu, C. Liu, and J. Sun. Hinormer: Representation learning on heterogeneous information networks with graph transformer. In *Proceedings of the ACM Web Conference, WWW*, pages 599–610, 2023.

[37] W. Ning, R. Cheng, J. Shen, N. A. H. Haldar, B. Kao, X. Yan, N. Huo, W. K. Lam, T. Li, and B. Tang. Automatic meta-path discovery for effective graph-based recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1563–1572, 2022.

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems, NeurIPS*, 32, 2019.

[39] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin. Large-scale evolution of image classifiers. In *International conference on machine learning, ICML*, pages 2902–2911, 2017.

[40] R. Sanders. The pareto principle: its use and abuse. *Journal of Services Marketing*, 1(2):37–40, 1987.

[41] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling. Modeling Relational Data with Graph Convolutional Networks. In *European semantic web conference*, pages 593–607, 2018.

[42] C. Sun, H. Gu, and J. Hu. Scalable and Adaptive Graph Neural Networks with Self-label-enhanced Training. *arXiv preprint arXiv:2104.09376*, 2021.

[43] Y. Sun and J. Han. Mining Heterogeneous Information Networks: A Structural Analysis Approach. *SIGKDD Explor.*, pages 20–28, 2012.

[44] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *Proc. VLDB Endow.*, pages 992–1003, 2011.

[45] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems, NeurIPS*, 34:24261–24272, 2021.

[46] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[47] R. Wang, M. Cheng, X. Chen, X. Tang, and C.-J. Hsieh. Rethinking architecture selection in differentiable nas. In *International Conference on Learning Representation, ICLR*, 2021.

[48] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous Graph Attention Network. In *Proceedings of the ACM Web Conference, WWW*, pages 2022–2032, 2019.

[49] L. Xie and A. L. Yuille. Genetic CNN. In *IEEE International Conference on Computer Vision, ICCV*, pages 1388–1397, 2017.

[50] S. Xie, H. Zheng, C. Liu, and L. Lin. SNAS: Stochastic Neural Architecture Search. In *7th International Conference on Learning Representations, ICLR*, 2019.

[51] C. Xue, X. Wang, J. Yan, Y. Hu, X. Yang, and K. Sun. Rethinking Bi-Level Optimization in Neural Architecture Search: A Gibbs Sampling Perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, pages 10551–10559, 2021.

[52] X. Yang, M. Yan, S. Pan, X. Ye, and D. Fan. Simple and Efficient Heterogeneous Graph Neural Network. *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, 2023.

[53] Q. Yao, J. Xu, W.-W. Tu, and Z. Zhu. Efficient Neural Architecture Search via Proximal Iterations. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, 2020.

[54] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems, NeurIPS*, 34:28877–28888, 2021.

[55] L. Yu, J. Shen, J. Li, and A. Lerer. Scalable Graph Neural Networks for Heterogeneous Graphs. *arXiv preprint arXiv:2011.09679*, 2020.

[56] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim. Graph Transformer Networks. In *Advances in Neural Information Processing Systems, NeurIPS*, pages 11960–11970, 2019.

[57] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous Graph Neural Network. In *KDD '19: In Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 793–803, 2019.

[58] W. Zhang, Z. Yin, Z. Sheng, Y. Li, W. Ouyang, X. Li, Y. Tao, Z. Yang, and B. Cui. Graph Attention Multi-layer Perceptron. In *KDD '22: In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 4560–4570, 2022.

[59] T. Zhao, C. Yang, Y. Li, Q. Gan, Z. Wang, F. Liang, H. Zhao, Y. Shao, X. Wang, and C. Shi. Space4hgnn: a novel, modularized and reproducible platform to evaluate heterogeneous graph neural network. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2776–2789, 2022.

[60] Z. Zhong, C.-T. Li, and J. Pang. Reinforcement learning enhanced heterogeneous graph neural network. *arXiv preprint arXiv:2010.13735*, 2020.

[61] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems, NeurIPS*, pages 321–328, 2003.

[62] Z. Zhou, J. Shi, R. Yang, Y. Zou, and Q. Li. Slotgat: slot-based message passing for heterogeneous graphs. In *International Conference on Machine Learning, ICML*, pages 42644–42657. PMLR, 2023.

[63] S. Zhu, C. Zhou, S. Pan, X. Zhu, and B. Wang. Relation Structure-aware Heterogeneous Graph Neural Network. In *2019 IEEE International Conference on Data Mining, ICDM*, pages 1534–1539, 2019.

[64] X. J. Zhu. Semi-supervised learning literature survey. 2005.

Appendix / Supplemental Material

In the Appendix, we provide additional details and results not included in the main text due to space limitations. First, we provide the details of datasets, tasks, baselines, and parameter settings for our experiments. Secondly, we show the search algorithm of LMSPS and conduct theoretical analysis about the reasonableness of sampling search and analyze the time complexity. Thirdly, for interpretability, the searched meta-paths are listed and analyzed. Fourthly, we explore the training efficiency and search convergence of LMSPS and compare LMSPS with top method combinations on `OGBN-MAG` leaderboard. Hyperparameter studies are also conducted. Finally, we discuss the reproducibility and limitations of our research work.

A Dataset and Task

A.1 Dataset Details

We evaluate our method on four widely-used heterogeneous graphs including `DBLP`, `IMDB`, `ACM`, and `Freebase` from HGB benchmark [34] and the famous large-scale dataset `OGBN-MAG` from OGB challenge [20]. The datasets from HGB follow a transductive setting, where all edges are available during training, and target type nodes are divided into 24% for training, 6% for validation, and 70% for testing. For the `OGBN-MAG` dataset, we use the official data partition, where papers published before 2018, in 2018, and since 2019 are nodes for training, validation, and testing, respectively. In addition, we construct four sparse datasets to evaluate the performance of LMSPS by reducing the maximum in-degree related to edge type in `OGBN-MAG`. The four sparse datasets have the same number of nodes with `OGBN-MAG`. Follow the convention [34, 52, 36], we show the details of all datasets in Table 7.

A.2 Node Classification Tasks

Following the convention [34, 27, 52, 36], we concentrate on semi-supervised node classification under the transductive setting and leave other downstream tasks related to heterogeneous graph representation learning as future work. Each dataset contains a target node type, and our task is to learn to predict the labels of vertices of this target node type. All the datasets provide fixed data splitting (into training, validation, and test sets) for node classification tasks.

A.3 Searching and Training Objective

An MLP layer is appended after the main module of LMSPS, which reduces the dimension of node-level representations of a heterogeneous graph to the number of classes:

$$\bar{Y} = \mathrm{softmax}(Z\omega_o), \tag{8}$$

where $\omega_o \in \mathbb{R}^{d_o \times N}$ is the output weight matrix, d_o is the dimension of output node representations, and N is the number of classes. Then, the cross-entropy loss is used over all labeled nodes:

$$\mathcal{L} = -\sum_{v \in \mathcal{V}_L} \sum_{n=1}^{N} \boldsymbol{y}_v[n] \log \bar{\boldsymbol{y}}_v[n], \tag{9}$$

Table 7: Statistics of the datasets.

Dataset	#Nodes	#Node types	#Edges	#Edge types	Target	#Classes
DBLP	$26,128$	4	$239,566$	6	author	4
IMDB	$21,420$	4	$86,642$	6	movie	5
ACM	$10,942$	4	$547,872$	8	paper	3
Freebase	$180,098$	8	$1,057,688$	36	book	7
OGBN-MAG	$1,939,743$	4	$21,111,007$	4	paper	349
OGBN-MAG-50	$1,939,743$	4	$9,531,403$	4	paper	349
OGBN-MAG-20	$1,939,743$	4	$7,975,003$	4	paper	349
OGBN-MAG-10	$1,939,743$	4	$6,610,464$	4	paper	349
OGBN-MAG-5	$1,939,743$	4	$4,958,941$	4	paper	349

where $\mathcal{V}_L$ denotes the set of labeled nodes, $\boldsymbol{y}_v$ is a one-hot vector indicating the label of node v, and $\bar{\boldsymbol{y}}_v$ is the predicted label for the corresponding node in $\bar{Y}$. In the search stage, $\mathcal{V}_L$ is the training set when updating $\boldsymbol{\omega}$ and the validation set when updating $\boldsymbol{\alpha}$. In the training stage, $\mathcal{V}_L$ is the training set, and ω is reinitialized and trained.

B Baselines and Parameter Settings

B.1 Baselines

We compare LMSPS with a large number of HGNN baselines, including MLP [19], GraphSAGE [14], RGCN [41], HAN [48], GTN [56], RSHN [63], HetGNN [57], MAGNN [11], HetSANN [18], GCN [26], GAT [46], , Simple-HGN [34], HGT [21], GraphMSE [31], SlotGAT [62], DiffMG [8], PMMM [27], HINormer [36], NARS [55], space4HGNN [59], SeHGNN [52]. GEMS [15] and RMS-HRec [37] are ignored because they are designed for recommendation tasks. RL-HGNN [60] is omitted due to a lack of source code.

Note that most of these baselines encounter out-of-memory (OOM) issues when dealing with large datasets. However, there are exceptions among them, including MLP, GraphSAGE, HGT, RGCN, NARS, and SeHGNN. The results of MLP and GraphSAGE come from the OGBN-MAG leaderboard. MLP [19] is famous for its lightweight and GraphSAGE [14] uses a neighbor sampling strategy, so they can handle large-scale datasets. HGT is designed with a graph sampling strategy, making it well-suited for managing large datasets. As for RGCN, while the original version does present OOM issues with large datasets, it is often used as a baseline for large-scale HGNNs [55]. NARS [55] and SeHGNN [52] are metapath-based and pre-computation-based HGNNs, allowing them to handle large-scale datasets.

It's worth noting that in the original paper, SeHGNN refers to the full version incorporating label propagation techniques. For clarity and consistency in our study, we have adjusted the terminology: we refer to the version without label propagation as SeHGNN and the version with label propagation as SeHGNN+LP. Note that label propagation is a general technique [64, 61, 22] that is compatible with most GNNs and is not used in the results of LMSPS and all other baselines in Table 1. Therefore, it is considered only in Appendix E.2, where we demonstrate the successful integration of our model with other techniques.

B.2 Training Settings

For training, we follow the settings of the HGB benchmark [34] and utilize the performance improvement on the validation set as a guide to determine whether the model has been improved. Specifically, if we observe a performance boost on the validation set during training, we update the final model parameters accordingly. Additionally, we adopt the early stopping strategy employed by the HGB benchmark: if no performance improvement on the validation set is observed within a specific count (referred to as patience), the training will be stopped early before reaching the maximum epochs. This approach helps prevent overfitting while also enhancing the computational efficiency of our model training. Following the setting of HGB [34] and OGB [20], we train our model 5 times for HGB and 10 times for OGB with different random seeds and report the mean performance and standard deviation, respectively. We use Pytorch [38] to run all experiments on one Tesla V100 GPU with 16GB GPU memory.

B.3 Parameter Settings for Baselines

For models such as RGCN, HAN, GTN, RSHN, HetGNN, MAGNN, HetSANN, HGT, GCN, GAT, Simple-HGN, and SeHGNN, the literature [34, 52] provides tuned parameter settings along with corresponding performance on small datasets, namely DBLP, IMDB, ACM, and Freebase. Therefore, we utilize these tuned parameter settings for these models on small datasets and utilize the reported performance.

Conversely, in cases where official results are unavailable (such as NARS on small datasets), or where the experimental settings differ, we employ the official or benchmark implementations [3][4][5][6][7][8][9][10][11] of the baseline models. We meticulously fine-tune their hyperparameters to the best of our capabilities. In instances involving hyperparameter analysis, such as in Figure 3, we only modify the relevant hyperparameters to ensure a fair comparison.

B.4 Parameter Settings for LMSPS

We set the number of selected meta-paths $M = 30$ for all datasets. The final search space $V = 60$. The maximum hop is 6 for `ogbn-mag`, DBLP, 5 for IMDB, ACM, and 3 for `Freebase`. All K architecture parameters $\alpha_1, \alpha_2, \cdots, \alpha_K$ are initialized as 1s. For searching in the super-net, we train for 200 epochs. To train the target-net, we use an early stop mechanism based on the validation performance to promise full training. A two-layer MLP is adopted for each meta-path in the feature projection step, and the hidden size is 512. All network weights are initialized by the Xavier uniform distribution [12] and are optimized with Adam [25] during training. In the search stage, λ is 1 during the first 20 epochs for warmup and decreases to 0 linearly. τ linearly decays with the number of epochs from 8 to 4. The learning rate is 0.001 for all search stages and HGB training stage, and 0.003 for `OGBN-MAG` training stage. The weight decay is always 0. For the initial search space, we simply preset the maximum hop and use all target-node-related meta-paths no more than this maximum hop.

B.5 Evaluation Metrics

To assess the performance of the models, we employ evaluation metrics consistent with those employed in baseline models. The metrics are chosen as follows. For small datasets, DBLP, IMDB, ACM, and Freebase, we adhere to the evaluation standards established by the HGB benchmark [34]. The metrics reported for these datasets are Macro-F1 and Micro-F1 scores, which evaluate the classification performance. For the OGBN-MAG dataset, evaluation follows the methodology outlined in NARS [55] and SeHGNN [52], where the classification accuracy score is reported for this dataset.

C Algorithm

C.1 The Search Algorithm

Our search stage aims to discover the most effective meta-path set from all target-node-related meta-paths, severely challenging the efficiency of searching. Take `OGBN-MAG` as an example. The number of target-node-related meta-paths K is 226 under the maximum hop 6, and we need to find the most effective meta-path set with size 30. Because different meta-paths could be noisy or redundant to each other, top-30 meta-paths are not necessarily the optimal solution when their importance is calculated independently. Therefore, the total number of possible meta-path sets is $C_{226}^{30} \approx 10^{37}$. Such a large search space is hard to solve efficiently by traditional RL-based algorithms [60, 37] or evolution-based algorithms [49, 39].

To overcome this challenge, our LMSPS first uses a progressive sample algorithm to shrink the search space size from 226 to 60, then utilizes a sampling evaluation strategy to discover the best meta-path set with the lowest validation loss, which is more effective than architecture parameters [47]. In each iteration, we only uniformly sample meta-paths M from the whole search space for parameter updates, so the search cost is relevant to M, which is a predefined small number, rather than K. Because the search stage has many iterations and the initial values of architecture parameters are

---

Algorithm 1 The search algorithm of LMSPS

Input: meta-path sets $\mathbb{P} = \{P_1, \cdots, P_K\}$; number of sampling meta-paths M; number of training iterations T; number of sampling evaluation E

Parameter: Network weights $\boldsymbol{\omega}$ in MLP_k for feature projection and MLP for downstream tasks; architecture parameters $\boldsymbol{\alpha} = \{\alpha_1, \cdots, \alpha_K\}$

Output: The index set of selected meta-paths $\mathbb{S}_M$

 1: **% Neighbor aggregation**
 2: Calculate neighbor aggregation of raw features for each $P_k \in \mathbb{P}$ based on Equation 1
 3: **while** $t < T$ **do**
 4:     **% Path strength**
 5:     Calculate the path strength of all meta-paths based on Equation 2
 6:     **% Dynamic search space**
 7:     Calculate the current search space $\mathbb{S}_C$ based on Equations 3
 8:     **% Sampling**
 9:     Determine the set of indexes of sampled meta-paths $\mathbb{S}$ based on Equation 4
10:     **% Semantic fusion**
11:     Fused the semantic information of the sampled meta-paths based on Equation 4
12:     **% Parameters updation**
13:     Update weights $\boldsymbol{\omega}$ by $\nabla_\omega \mathcal{L}_{train}(\boldsymbol{\omega}, \boldsymbol{\alpha})$
14:     Update parameters $\boldsymbol{\alpha}$ by $\nabla_\alpha \mathcal{L}_{val}(\boldsymbol{\omega}, \boldsymbol{\alpha})$
15: **end while**
16: **% Evaluation**
17: **while** $e < E$ **do**
18:     Randomly sample M meta-paths from $\mathbb{S}_C$ as $\bar{\mathbb{S}}$ based on Equation 6
19:     Calculate $\mathcal{L}_{val}(\bar{\mathbb{S}})$ of the sampled meta-paths
20: **end while**
21: Select the best meta-path set $\mathbb{S}_M \leftarrow \mathrm{argmin}_{\bar{\mathbb{S}}} \mathcal{L}_{val}(\bar{\mathbb{S}})$
22: **return** $\mathbb{S}_M$

---

the same, all architecture parameters will be updated multiple times and the relative importance can be learned during training, making the total search cost similar to training a single HGNN once. Specifically, for `OGBN-MAG`, LMSPS can finish searching in two hours.

Except for efficiency, LMSPS can also overcome the over-squashing issue [1] when utilizing long-range dependency. Over-squashing means the distortion of messages being propagated from distant nodes, which has been heuristically attributed to graph bottlenecks where the number of l-hop neighbors grows rapidly with l. In LMSPS, we set the number of searched meta-paths $M = 30$ for all datasets, which is independent of l. Because the exponential meta-paths in metapath-based methods correspond to exponential receptive fields in metapath-free methods, the constant M means approximately constant l-hop neighbors, *i.e.*, LMSPS selectively aggregate effective neighbors for each target-net. Therefore, the over-squashing issue is overcome.

We have introduced the components of LMSPS in detail in the main text. LMSPS first employs a progressive sample algorithm to narrow the search space from K to V, then utilizes sample evaluation to filter out the best meta-path set with M meta-paths. V is a parameter to trade off the importance of progressive sampling search and sampling evaluation. When V is too large, we need to repeat the sampling evaluation many more times, which will decrease the efficiency. When V is too small, some effective meta-paths may be dropped too early. So, we simply set $V = 2M$. Generally, $K \gg M$ under a large maximum hop. For a small maximum hop, when $K \leq M$, the search stage is unnecessary because we can directly use all target-node-related meta-paths; when $K \leq 2M$, the progressive sampling algorithm is unnecessary because the search space is small enough. When $K > 2M$, we show the overall search algorithm in Algorithm 1.

C.2   Theoretical analysis of the reasonableness of sampling search

Definition 1. ***Zero-order condition****: Given two high-dimensional random variables, $\mathbf{y} = \mathbf{f}(\mathbf{x}) \in \mathbf{R}^{\mathbf{M} \times \mathbf{d_1}}$ and $\mathbf{z} = \mathbf{g}(\mathbf{x}) \in \mathbf{R}^{\mathbf{M} \times \mathbf{d_1}}$, the zero-order condition is satisfied if $|\mathbf{y} - \mathbf{z}|_2 \leq \epsilon$ for any valid sample $\mathbf{x} \in \mathbf{R}^{\mathbf{N} \times \mathbf{d}}$, where ϵ is a small positive constant.*

Table 8: Time complexity comparison of every training epoch. † means time complexity under small-scale datasets and full-batch training.

Method	Feature projection	Neighbor aggregation	Semantic fusion	Total
HAN	$O(N(rd)^l F^2)$	$O(N(rd)^l F)$	$O(Nr^l F^2)$	$O(N(rd)^l F^2)$
Simple-HGN	$O(N(rd)^l F^2)$	$O(N(rd)^l F)$	-	$O(N(rd)^l F^2)$
Simple-HGN†	$O(NrdlF^2)$	$O(NrdlF)$	-	$O(NrdlF^2)$
SeHGNN	$O(Nr^l F^2)$	-	$O(Nr^l F^2 + Nr^{2l} F)$	$O(N(r^l F^2 + r^{2l} F))$
LMSPS-search	$O(NMF^2)$	-	$O(NMF^2)$	$O(NMF^2)$
LMSPS-train	$O(NMF^2)$	-	$O(NMF^2)$	$O(NMF^2)$

Lemma C.1. *Let* $\mathbf{m}$ *represent the maximum number of activatable paths, and assume each pair of operations satisfies the zero-order condition. We can approximate all* $2^{\mathbf{m}}$ *combinations using* $\mathbf{m}$ *types of expectations and variances.*

Lemma C.1 assures us that we can approximate all $2^{\mathbf{m}}$ combinations by multiple times sampling. The proof is given as follows.

Let $\mathbf{y}_{p_{\mathbf{y}}(\mathbf{y})} = f(\mathbf{x})$, $\mathbf{z}_{p_{\mathbf{z}}(\mathbf{z})} = g(\mathbf{x})$, $\mathbf{x} \sim p_{\mathbf{x}}(\mathbf{x})$. For the case $m = 1$, the expectation of $\mathbf{y}$ and $\mathbf{z}$ can be written respectively as:

$$\mathbb{E}[\mathbf{y}] = \mathbb{E}[f(\mathbf{x})] = \int p_{\mathbf{x}}(\mathbf{x}) f(\mathbf{x}) \mathrm{d}\mathbf{x}$$
$$\mathbb{E}[\mathbf{z}] = \mathbb{E}[g(\mathbf{x})] = \int p_{\mathbf{x}}(\mathbf{x}) g(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{10}$$

According to the zero-order condition, we have $f(\mathbf{x}) \approx g(\mathbf{x})$. And $p(\mathbf{x})$ is same for both $\mathbf{y}$ and $\mathbf{z}$, so $\mathbb{E}[\mathbf{y}] \approx \mathbb{E}[\mathbf{z}]$.

Now we prove $Var[\mathbf{y}] \approx Var[\mathbf{z}]$. Note that $Var[\mathbf{y}] = \mathbb{E}\left[\mathbf{y}^2\right] - (\mathbb{E}[\mathbf{y}])^2$ and $Var[\mathbf{z}] = \mathbb{E}\left[\mathbf{z}^2\right] - (\mathbb{E}[\mathbf{z}])^2$, thus we only need to prove $\mathbb{E}\left[\mathbf{y}^2\right] \approx \mathbb{E}\left[\mathbf{z}^2\right]$. It can be similarly proved as follows:

$$\mathbb{E}\left[\mathbf{y}^2\right] = \int p_{\mathbf{y}}(\mathbf{y}) \mathbf{y}^2 \mathrm{d}\mathbf{y} = \int p_{\mathbf{x}}(\mathbf{x}) f^2(\mathbf{x}) \mathrm{d}\mathbf{x}$$
$$\mathbb{E}\left[\mathbf{z}^2\right] = \int p_{\mathbf{z}}(\mathbf{z}) \mathbf{z}^2 \mathrm{d}\mathbf{z} = \int p_{\mathbf{x}}(\mathbf{x}) g^2(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{11}$$

According to the zero-order condition, we have $Var[\mathbf{y}] \approx Var[\mathbf{z}]$.

For the case of $m = 2$, when the two paths are both selected, the output becomes $\mathbf{y} + \mathbf{z}$, its expectation can be written as:

$$\mathbb{E}[\mathbf{y} + \mathbf{z}] = \mathbb{E}[\mathbf{y}] + \mathbb{E}[\mathbf{z}] \approx 2\mathbb{E}[\mathbf{y}] \tag{12}$$

And the variance of $\mathbf{y} + \mathbf{z}$ is,

$$Var[\mathbf{y} + \mathbf{z}] \approx Var[2\mathbf{y}] = 4Var[\mathbf{y}] \tag{13}$$

Therefore, there are two kinds of expectations and variances: $\mathbb{E}[\mathbf{y}]$ and $Var[\mathbf{y}]$ for $\{\mathbf{y}, \mathbf{z}\}$, and $2\mathbb{E}[\mathbf{y}]$ and $4Var[\mathbf{y}]$ for $\{\mathbf{y} + \mathbf{z}\}$. Similarly, in the case where $m \in [1, n]$, there will be m kinds of expectations and variances.

C.3 Time Complexity Analysis

Following the convention [6, 52], we compare the time complexity of LMSPS with HAN [48], Simple-HGN [34], and SeHGNN [52] under mini-batch training with the total N target nodes. All methods employ l-hop neighborhood. For simplicity, we assume that the number of features is fixed to F for all layers. The average degree of each node is rd, where r is the number of edge types and d is the number of edges connected to the node for each edge type. The complexity analysis is

Table 9: Meta-paths searched by LMSPS on different datasets.

Dataset	Meta-paths learnt by LMSPS
DBLP	AP, APT, APVP, APAPA, APTPA, APTPT, APTPV, APVPA, APVPV, APAPAP, APAPTP, APAPVP, APTPTP, APTPVP, APVPTP, APAPAPV, APAPTPA, APAPTPV, APAPVPA, APAPVPT, APAPVPV, APTPAPA, APTPAPT, APTPTPT, APTPVPV, APVPAPT, APVPTPA, APVPVPA, APVPVPT, APVPVPV
IMDB	M, MA, MK, MAM, MDM, MKM, MAMK, MDMK, MKMD, MDMKM, MKMAM, MKMDM, MKMKM, MAMAMK, MAMDMA, MAMDMK, MAMKMD, MAMKMK, MDMAMA, MDMAMD, MDMDMA, MDMDMK, MDMKMA, MKMAMA, MKMAMD, MKMAMK, MKMDMA, MKMDMK, MKMKMD, MKMKMK
ACM	PPP, PAPP, PCPA, PCPP, PPPC, PPPP, PAPAP, PAPPP, PCPAP, PCPPA, PPAPA, PPAPC, PPAPP, PAPAPA, PAPCPA, PAPPAP, PAPPCP, PAPPPP, PCPAPP, PCPCPP, PCPPAP, PCPPPP, PPAPAP, PPAPCP, PPAPPA, PPAPPP, PPCPAP, PPPAPA, PPPCPA, PPPPPP
OGBN-MAG	PF, PAPF, PFPP, PAPPP, PFPFP, PPAPP, PPPAP, PPPFP, PAIAPP, PAPAPF, PFPAPF, PFPPPF, PFPPPP, PPAPPF, PPPAPF, PPPPAP, PPPPPF, PAIAPAP, PAPAPAP, PAPAPPP, PAPPPAI, PAPPPPF, PAPPPPP, PFPAPAP, PFPPFPF, PFPPPPF, PPAPAPF, PPAPAPP, PPPAIAI, PPPAPPP

summarized in Table 8. Because HAN and Simple-HGN require neighbor aggregation during training, and the number of neighbors grows exponentially with hops. So, they have neighbor aggregation costs of $O((rd)^l)$. SeHGNN employs a pre-processing step to avoid the training cost of neighbor aggregation. However, the exponential meta-paths cause SeHGNN to suffer from $O(r^l)$ costs in semantic aggregation. Unlike the above methods, LMSPS samples M meta-paths in each iteration of the search stage and employs M effective meta-paths in the training stage to avoid exponential costs. Generally, we have $O((rd)^l) \gg O(r^l) = O(K) \gg O(M)$ when maximum hop l is large. The time complexity of LMSPS is a constant when N and F are determined, which is the key point for utilizing long-range meta-paths.

D   Interpretability of Searched Meta-paths

We have conducted an extensive experimental study to validate the effectiveness of our searched meta-paths in the main text. Here, we illustrate the searched meta-paths of each dataset in Table 9. Because we discover many more meta-paths than traditional methods and most meta-paths are longer than traditional meta-paths, it is tough to interpret them one by one. So, we focus on the interpretability of meta-paths on large-scale `obgn-mag` dataset from the Open Graph Benchmark. The `OGBN-MAG` dataset is a heterogeneous graph composed of a subset of the Microsoft Academic Graph. It includes four different entity types: Papers (P), Authors (A), Institutions (I), and Fields of study (F), as well as four different directed relation types: Author $\xrightarrow{\text{writes}}$ Paper, Paper$\xrightarrow{\text{cites}}$Paper, Author $\xrightarrow{\text{is affiliated with}}$ Institution, and Paper$\xrightarrow{\text{has a topic of}}$Field. The target node is the paper, and the task is to predict each paper's venue (conference or journal).

Based on Table 9, the hop of effective meta-paths on `obgn-mag` ranges from 1 to 6, which means utilizing information from neighbors at different distances is important. Because long-range meta-paths provide larger receptive fields, LMSPS shows stronger capability in utilizing heterogeneity compared to traditional metapath-based HGNNs. The source node type of 16 meta-paths is P, *e.g.*, PFPFP (P←F←P←F←P). It indicates that the neighborhood papers of the target paper are most significant for predicting its venue, which is consistent with reality: the citation relationship, co-author relationship, and co-topic relationship between papers are usually the most effective information. 12 meta-paths' source node type is F. It implies that the neighborhood fields of the target paper are also crucial in determining its venue, which is also consistent with reality. Because most conferences or journals focus on a few fixed fields, a paper's venue is highly related to its field. The source node type of 2 meta-paths is I. It means the neighborhood institution is not very important for predicting the paper's venue, which is reasonable because almost all institutions have a wide range of conference or journal options for publishing papers. No meta-path has source node type A. It means the neighborhood author is unimportant in determining the paper's venue, which is logical because each paper has multiple authors, and each author can consider different venues. So, it is difficult to determine the paper's venue based on its neighborhood authors. If using too much institution or author information to predict the paper's venue, it actually introduces much useless information, which can be viewed as a kind of noise in `obgn-mag` for predicting each paper's venue.
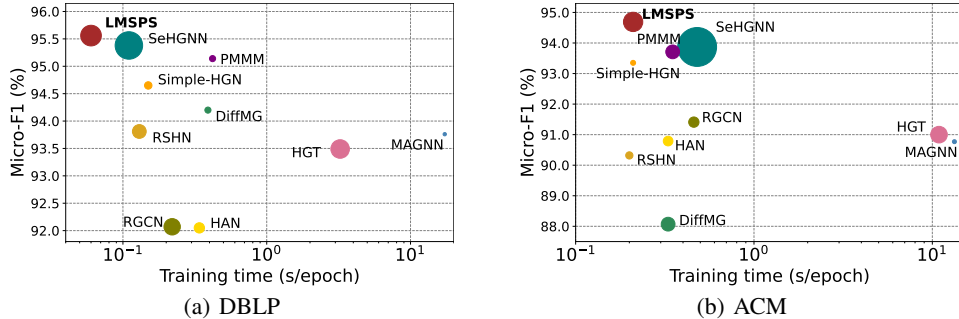
Figure 4: Micro-F1 scores, time consumption, and parameters of various HGNNs on `DBLP` and `ACM`. GTN has a large time consumption and parameters. We ignore it for ease of illustration.

Table 10: Performance of top mothod combinations on OGBN-MAG leaderboard.

Method	OGBN-MAG	
	Validation Accuracy	Test Accuracy
SAGN [42]+LP	52.25 ± 0.30	51.17 ± 0.32
GAMLP [58]+LP	53.23 ± 0.23	51.63 ± 0.22
SeHGNN+LP	55.95 ± 0.11	53.99 ± 0.18
LMSPS +LP	$\mathbf{56.98 \pm 0.10}$	$\mathbf{55.10 \pm 0.11}$
SeHGNN+LP+MS	58.70 ± 0.08	56.71 ± 0.14
SeHGNN+LP+MS+Emb	59.17 ± 0.09	57.19 ± 0.12
LMSPS +LP+MS	$\mathbf{59.51 \pm 0.07}$	$\mathbf{57.84 \pm 0.22}$

We also provide some simple insights from the searced meta-paths of DBLP, IMDB, and ACM. In DBLP with target node type Author, the information from P (Paper) and A (Author) is slightly more important than that from T (Term) and V (Venue). In IMDB with target node type Movie, the importance of information of K (Keyword), M (Movie), A (Actor) and D (Director) gradually decreases. In ACM with target node type Paper, the importance of information of P (Paper), A (Author) and C (Conference) gradually decreases. For all datasets, the importance of node type is highly related to the target node type.

In addition, the hand-crafted meta-paths rely on intense expert knowledge, which is both laborious and data-dependent. In contrast, automatic meta-path search frees researchers from the understand-then-apply paradigm. In Table 9, LMSPS can search effective meta-paths without prior knowledge for various datasets, which is more valuable than manually-defined meta-paths.

E   More Experiments

E.1   Comparison on Training Efficiency

Following the convention [34, 52], we show the time cost and parameters of LMSPS and the advanced baselines on `DBLP` and `ACM` in Figure 4. We measure the average time consumption of one epoch for each model. The area of the circles represents the parameters. The hidden size is set to 512 and the maximum hop or layer is 6 for `DBLP` and 5 for `ACM` for all methods to test the training time and parameters under the same setting. Some methods perform quite poorly under the large maximum hop or layer. So we show the performance from Table 1 of the main text, which is the results under their best settings. Figure 4 shows that LMSPS has advantages in both training efficiency and performance. Our searched meta-paths are universal after searching once and can be applied to other metapath-based HGNNs based on Table 4. Because other metapath-based HGNNs don't include the time for discovering manual meta-paths, we also exclude our search time for discovering meta-paths in Figure 4. The search time is shown in Figure 3 (c) of the main text.

E.2   Comparison with Top Method Combinations

The OGBN-MAG benchmark dataset is associated with a public leaderboard. In Table 10, we compare our method combination against top-performing approaches (method combinations) on the leaderboard. These methods integrate several general techniques, such as Label Propagation

Table 11: The performance of LMSPS under different searching epochs.

Dataset	1	5	10	20	40	60	80	100	120	140	160	180	200
DBLP	38.32	68.75	71.20	72.03	71.05	70.56	73.12	73.70	73.56	73.85	73.84	74.36	74.04
IMDB	43.48	49.78	53.88	58.48	57.03	58.40	58.05	58.65	57.54	58.70	58.59	58.72	58.81
ACM	79.03	85.43	87.42	86.75	88.30	89.85	88.96	89.40	88.52	89.85	89.70	89.85	89.91
Freebase	18.37	29.55	39.62	39.58	44.43	46.57	47.52	48.33	54.59	55.81	55.52	55.66	55.79
OGBN-MAG	14.21	26.20	32.41	37.00	40.82	45.89	41.74	44.00	48.40	48.55	48.70	48.43	48.59

(a) DBLP   (b) IMDB   (c) ACM
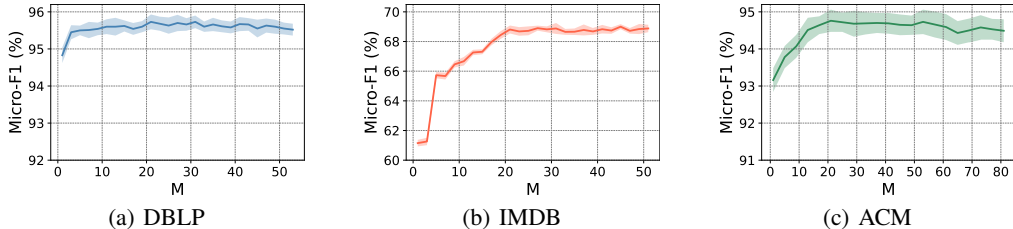
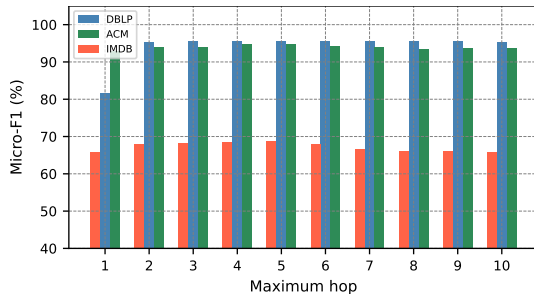Figure 5: Micro-F1 with respect to different hyper-parameter M on DBLP, IMDB, and ACM.

Figure 6: Micro-F1 scores with respect to different maximum hops on DBLP, IMDB, and ACM.

(LP), Multi-Stage Training (MS), and pre-trained Embeddings (Emb). By integrating LP and MS, LMSPS+LP+MS outperforms SeHGNN+LP+MS by a large margin. SeHGNN shows the results with extra embeddings as an enhancement. Though we cannot conduct a fair comparison with this trick due to the untouchability of its embedding file, LMSPS still outperforms their best result.

E.3 Convergence of the Search Stage

As the search stage of LMSPS relies on sampling, we conduct experiments to explore the convergence of the search stage. Because the test set is unavailable in the search stage to avoid label leakage, we show the Micro-F1 scores of the validation set of DBLP, IMDB, ACM, Freebase and test accuracy of the validation set of OGBN-MAG under different searching epochs in Table 11. As we can see, LMSPS coverages during 100 to 180 epochs for all datasets. The main reason is that the search space size is progressively shrunk from K to $2M$ under the progressive sampling algorithm. Most negative meta-paths have been removed from the search space in the second half of the search stage.

E.4 Hyperparameter Study

LMSPS randomly samples M meta-paths at each epoch in the search stage and selects the top-M meta-paths in the training stage. Here, we perform analysis on hyper-parameter M on DBLP, IMDB, and ACM. Freebase and OGBN-MAG are ignored because they are relatively large and the experiments are time-consuming. As illustrated in Figure 5, the performance of LMSPS increases with the growth of M when M is small. In addition, the performance on DBLP and ACM slightly decreases when M is larger than a certain threshold, indicating excessive meta-paths don't benefit performance for some datasets. For unity, we set $M = 30$ for all datasets.

To observe the impact of different maximum hop values, we show the Micro-F1 of LMSPS with respect to different maximum hop values on DBLP, IMDB, and ACM in Figure 6. We can see LM-SPS show the best performance when the maximum hop is 5 or 6. Additionally, the performance of

LMSPS does not always increase with the value of the maximum hop, and the best maximum hop depends on the dataset.

F Reproducibility Statement

We have provided the details of datasets, tasks, baselines, and parameter settings in Appendix A and B and conducted the hyperparameter study in Appendix E.4. All reported results are the average of multiple experiments with standard deviations. We have included a pseudocode description of our method in Appendix C. The source code has been provided through an anonymized URL with clear commands on reproducing our results.

G Limitations

It is noteworthy that the performance of LMSPS does not always increase with the value of the maximum hop. For instance, based on Figure 3 (a), LMSPS can effectively utilize 12-hop meta-paths on DBLP with high performance and low cost. However, the optimal performance for LMSPS is observed when the maximum hop is 6. It can be attributed to the constraint of maintaining constant time complexity by setting the number of searched meta-paths to 30 for different maximum hop values. When the maximum hop value is 12, the number of target-node-related meta-paths exceeds 1400 (Figure 3 (a)). Searching for 30 effective meta-paths from such an extensive search space is notably challenging, despite LMSPS being the most effective method for meta-path search (as indicated in Table 3). The benefits of using longer meta-paths are outweighed by the drawbacks of the significantly more complex search space. So, the best maximum hop depends on the dataset and task and cannot be determined automatically.

Nevertheless, based on Table 8, the time complexity of LMSPS does not increase with the maximum hop. Consequently, it provides an effective solution for utilizing long-range dependency on heterogeneous graphs for possible applications on sparser real-world datasets. In Table 5, we conduct experiments on four constructed datasets based on OGBN-MAG to demonstrate that the advantages of utilizing long-range dependencies are more obvious for sparser large-scale heterogeneous graphs. For the constructed datasets, we carefully avoid inappropriate preference seed settings of randomly removing by limiting the maximum in-degree related to edge type. We expect sparser large-scale real heterogeneous datasets or more suitable tasks requiring long-range dependency to emerge in the future so we can fully explore our approach.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have provided cases, motivation, and also extensive experiments for verification to support our claim.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have talked about the limitation of our work in Appendix G.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We have provided a theoretical analysis of the reasonableness of the sampling search in Appendix C.2.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We have provided extensive experiments, results, descriptions, and also provided the code.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is provided in a URL in the Abstract.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The details of datasets, tasks, baselines, and parameter settings are shown in Appendix A and B. The hyperparameter study is shown in Appendix E.4

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The mean and variance of the experimental results are reported.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We have provided that the computation resource is one Tesla V100 GPU with 16GB GPU memory.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: We have read.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: Our work is specific to machine learning for heterogeneous graph node classification; therefore, it has limited possibility for a wide societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Similarly, our work is specific to machine learning for heterogeneous graph node classification.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All the data can be found publicly, and the code we provided has no constraints.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The URL of our code provides a detailed README to reproduce our results.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.