Space4HGNN: A Novel, Modularized and Reproducible Platform to Evaluate Heterogeneous Graph Neural Network

Tianyu Zhao
tyzhao@bupt.edu.cn
Beijing University of Posts and Telecommunications
China

Cheng Yang
yangcheng@bupt.edu.cn
Beijing University of Posts and Telecommunications
China

Yibo Li
liushiliushi@bupt.edu.cn
Beijing University of Posts and Telecommunications
China

Quan Gan
quagan@amazon.com
AWS Shanghai AI Lab
China

Zhenyi Wang
zy_wang@bupt.edu.cn
Beijing University of Posts and Telecommunications
China

Fengqi Liang
lfq@bupt.edu.cn
Beijing University of Posts and Telecommunications
China

Huan Zhao
zhaohuan@4paradigm.com
4Paradigm Inc.
China

Yingxia Shao
shaoyx@bupt.edu.cn
Beijing University of Posts and Telecommunications
China

Xiao Wang
xiaowang@bupt.edu.cn
Beijing University of Posts and Telecommunications
Peng Cheng Laboratory
China

Chuan Shi†
shichuan@bupt.edu.cn
Beijing University of Posts and Telecommunications
Peng Cheng Laboratory
China

ABSTRACT
Heterogeneous Graph Neural Network (HGNN) has been successfully employed in various tasks, but we cannot accurately know the importance of different design dimensions of HGNNs due to diverse architectures and applied scenarios. Besides, in the research community of HGNNs, implementing and evaluating various tasks still need much human effort. To mitigate these issues, we first propose a unified framework covering most HGNNs, consisting of three components: heterogeneous linear transformation, heterogeneous graph transformation, and heterogeneous message passing layer. Then we build a platform Space4HGNN by defining a design space for HGNNs based on the unified framework, which offers modularized components, reproducible implementations, and standardized evaluation for HGNNs. Finally, we conduct experiments to analyze the effect of different designs. With the insights found, we distill a condensed design space and verify its effectiveness.

CCS CONCEPTS
• Computing methodologies → Machine learning; Neural networks; • Information systems → Evaluation of retrieval results.

KEYWORDS
heterogeneous graph; graph neural networks; design space

† The corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGIR '22, July 11–15, 2022, Madrid, Spain
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-8732-3/22/07...
https://doi.org/10.1145/3477495.3531720

1 INTRODUCTION
In information retrieval, graph neural network (GNN) as graph learning and representation method has been applied in recommendation [3, 4, 6, 50, 60] and knowledge representation [2, 17, 20, 37]. Most GNNs focus on homogeneous graphs, while more and more
research [12, 30, 41, 51, 57] shows that real world with complex interactions, e.g., social network [46, 48], can be better modeled by heterogeneous graphs (a.k.a., heterogeneous information networks). Taking recommender system as an example, it can be regarded as a bipartite graph consisting of users and items, and a lot of auxiliary information also has a complex network structure, which can be naturally modeled as a heterogeneous graph. Besides, some works [1, 5, 11, 15, 18, 29, 33, 35] have achieved state-of-the-art (SOTA) performance by designing heterogeneous graph neural network (HGNN). In fact, HGNNs can utilize the complex structure and rich semantic information [49], and have been widely applied in many fields, such as e-commerce [28, 61], and security [23, 44].

However, it is increasingly difficult for researchers in the field to compare existing methods and contribute with novel ones. The reason is that previous evaluations are conducted from the point of view of model-level, and we cannot accurately know the importance of each component due to diverse architecture designs and applied scenarios. To evaluate them from the sight of module-level, we first propose a unified framework of existing HGNNs that consists of three key components through systematically analyzing their underlying graph data transformation and aggregation procedures, as shown in Figure 1 (right). The first component Heterogeneous Linear Transformation is a general operation of HGNNs, which maps features to a shared feature space. Summarizing the transformed graphs used in different HGNNs, we abstract the second component Heterogeneous Graph Transformation containing relation subgraph extraction, meta-path subgraph extraction, and homogenization of the heterogeneous graph. With that, we can explicitly decouple the selection procedure of receptive field and message passing procedure. Hence the third component Heterogeneous Message Passing Layer can focus on the key procedure involving diverse graph convolution layers. As shown in Table 1, our framework both categorizes existing approaches and facilitates the exploration of novel ones.

With the help of the unified framework, we propose to define a design space for HGNNs, which consists of a Cartesian product of different design dimensions following GraphGym [55]. In GraphGym, there have been analysis results of design dimensions for GNNs, which is not enough for HGNNs. To figure out whether the guidelines distilled from GNNs are effective, our design space still contains common design dimensions with GraphGym. Besides, to capture heterogeneity, we distill three model families according to Heterogeneous Graph Transformation in our unified framework. Based on the design space, we build a platform Space4HGNN ¹, which offers reproducible model implementation, standardized evaluation for diverse architecture designs, and easy-to-extend API to plug in more architecture design options. We believe Space4HGNN can greatly facilitate the research field of HGNNs. Specifically, we could check the effect of the tricky designs or architecture design quickly, innovate the HGNN models easily, and apply HGNN in other interesting scenarios. In addition, the platform can be used as the basis of neural architecture search for HGNNs in future work.

With the platform Space4HGNN, we conduct extensive experiments and aim to analyze the design dimensions. We first evaluate common design dimensions used in GraphGym with uniform random search, and find that they are partly effective in HGNNs. More importantly, to accurately judge the diverse architecture designs in HGNNs, we comprehensively analyze unique design dimensions in HGNNs. And we sum up the following insights:

- Different model families have different suitable scenarios. The meta-path model family has an advantage in node classification task, and the relation model family performs outstandingly in link prediction task.
- The preference for different design dimensions may be opposite in different tasks. For example, node classification task prefers to apply L2 Normalization and remove Batch Normalization. However, the better choices of the same datasets for link prediction task are the opposite.
- We should select graph convolution carefully which varies greatly across datasets. Besides, the design dimensions like the number of message passing layers, hidden dimension and dropout are all important.

Finally, we distill a condensed design space according to the analysis results, whose scale is reduced by 500 times. We evaluate it in a new benchmark HGB [36] and demonstrate the effectiveness of the condensed design space.

And we sum up the following contributions:

- As far as we know, we are the first to propose a unified framework and define a design space for HGNNs. They offer us a module-level sight and help us evaluate the influences of different design dimensions, such as high-level architectural designs, and design principles.
- We release a platform Space4HGNN for design space in HGNNs, which offers modularized components, standardized evaluation, and reproducible implementation of HGNN. We conduct extensive experimental evaluations to analyze

1https://github.com/BUPT-GAMMA/Space4HGNN
Table 1: Different perspectives to categorize HGNN models: (1) The first row from the perspective of neighbors to be aggregated is mentioned in Section 2.1. (2) The second row shows we roughly divide the existing models into three categories mentioned in Section 4.1.2. (3) The third and the fourth rows show the components of our unified framework in Section 3. (4) The fifth row: from the perspective of implementation, they contain different convolution layers.

| Neighbors | One-hop | Meta-path |
|-----------|---------|-----------|
| Model Family | Homogenization | Relation |
| Heterogeneous Graph Transformation | Homogenization of the heterogeneous graph | Relation subgraph extraction | Meta-path subgraph extraction |
| Heterogeneous Message Passing | Direct-aggregation | Dual-aggregation |
| Graph Convolution | Single graph homogenous convolution | Single graph heterogeneous convolution | Multiple homogeneous graph convolutions applied to different subgraphs |
| Model | GCN [31], GAT [47], GraphSage [19], GIN [53] | HGAT [34], HetSANN [22], HGT [24], Simple-HGN [36] | RGCN [41], HGConv [56] | HAN [51], HPN [27] |

HGNNs comprehensively, and provide findings behind the results based on Space4HGNN. It allows researchers to find more interesting findings and explore more robust and generalized models.

- Following the findings, we distill a condensed design space. Experimental results on a new benchmark HGB [36] show that we can easily achieve state-of-the-art performance with a simple random search in the condensed space.

2 RELATED WORK

Notations and the corresponding descriptions used in the rest of the paper are given in Table 2. More preliminaries can be found in Appendix A.

Table 2: Notation and corresponding description.

| Notation | Description |
|----------|-------------|
| \( v_i \) | The node \( v_i \) |
| \( e_{ij} \) | The edge from node \( v_i \) to node \( v_j \) |
| \( N_i \) | The neighbors of node \( v_i \) |
| \( h \) | The hidden representation of a node |
| \( W \) | The trainable weight matrix |
| \( f_{node} \) | The node type mapping function |
| \( f_{edge} \) | The edge type mapping function |
| \( \phi \) | The message function |

2.1 Heterogeneous Graph Neural Network

Different from GNNs, HGNNs need to handle the heterogeneity of structure and capture rich semantics of heterogeneous graphs. According to the strategies of handling heterogeneity, HGNN can be roughly classified into two categories: HGNN based on one-hop neighbor aggregation (similar to traditional GNN) and HGNN based on meta-path neighbor aggregation (to mine semantic information), shown in Table 1.

2.1.1 HGNN based on one-hop neighbor aggregation. To deal with heterogeneity, this kind of HGNN usually contains type-specific convolution. Similar to GNNs, the aggregation procedure occurs in one-hop neighbors. As earliest work and an extension of GCN [31], RGCN [41] assigns different weight matrices to different relation types and aggregates one-hop neighbors. With many GNN variants appearing, homogeneous GNNs inspire more HGNNs and then HGConv [56] dual-aggregate one-hop neighbors based on GATConv [47]. A recent work SimpleHGN [36] designs relation-type weight matrices and embeddings to characterize the heterogeneous attention over each edge. Besides, some earlier models, like HGAT [34], HetSANN [22], HGT [24], modify GAT [47] with heterogeneity by assigning heterogeneous attention for either nodes or edges.

2.1.2 HGNN based on meta-path neighbor aggregation. Another class of HGNNs is to capture higher-order semantic information with hand-crafted meta-paths. Different from the previous, aggregation procedure occurs in neighbors connected by meta-path. As a pioneering work, HAN [51] first uses node-level attention to aggregate nodes connected by the same meta-path and utilizes semantic-level attention to fuse information from different meta-paths. Because the meta-path subgraph ignores all the intermediate nodes, MAGNN [13] aggregates all nodes in meta-path instances to ensure that information will not be missed. Though meta-paths contain rich semantic information, the selection of meta-paths needs human prior and determines the performance of HGNNs. Some works like GTN [58] learn meta-paths automatically to construct a new graph. HAN [51] and HPN [27], which are easy to extend, will be included in our framework for generality.

2.2 Model Evaluation and Design Space

There are many works to measure progress in the field by evaluating models. The work [7] reports the results of a systematic analysis of neural recommendation mentioning HGNNs and sheds light on potential problems. Several works [9, 10, 42] discuss how to make fair a comparison between GNN models. In HGNNs, a recent work [36] revisits HGNNs and proposes issues with existing HGNNs. The above work is to evaluate models from the model-level sight.
Figure 2: The detailed description of three components using the graph illustrated in Figure 1. (a) Heterogeneous Linear Transformation maps all node features to a shared feature space. (b) Heterogeneous Graph Transformation (The left one) The original graph consists of four adjacency matrices representing four relations. (The right four) Four transformation methods. (c) Two aggregation methods in Heterogeneous Message Passing Layer. (The left one) The direct-aggregation: lines assigned normalized attention coefficients indicate aggregation procedure. (The right one) The dual-aggregation: the black solid line indicates the micro-level aggregation procedure applied in different subgraphs and the thick red solid lines indicates the macro-level aggregation procedure.

Though rigorous theoretical understanding of neural network is not enough, it is imperative to perform empirical studies of neural network to discover better architectures. In visual recognition, some works [39, 40] design network design spaces to help advance the understanding of network design and discover design principles that generalize across settings. Inspired by that, GraphGym [55] proposes a GNN design space and a GNN task space to evaluate model-task combinations comprehensively. In recommendation system, the work [52] profiles the design space for GNNs based on collaborative filtering.

Here we aim to extensively explore the design space of HGNNs involving many design dimensions and evaluate different design architectures. Different from the model-level sight of [36], we evaluate the design dimensions from module-level sight and distill helpful design principles. Different from GraphGym [55], our design space focuses on the unique design dimension of HGNNs and explores the differences with GNNs.

3 A UNIFIED FRAMEWORK OF HETEROGENEOUS GRAPH NEURAL NETWORK

As shown in Table 1, we categorize many mainstream HGNN models, which could be applied in many scenarios, e.g., link prediction [20, 37] and recommendation [1, 15, 29]. Through analyzing the underlying graph data and the aggregation procedure of existing HGNNs, we propose a unified framework of HGNN that consists of three main components:

- **Heterogeneous Linear Transformation** maps features or representations with heterogeneity to a shared feature space.
- **Heterogeneous Graph Transformation** offers four transformation methods for heterogeneous graph data to select the receptive field.
- **Heterogeneous Message Passing Layer** defines two aggregation methods suitable for most HGNNs.

3.1 Heterogeneous Linear Transformation

Due to the heterogeneity of nodes, different types of nodes have different semantic features even different dimensions. Therefore, for each type of nodes (e.g., node $v_i$ with node type $f_{v_i}(u_j)$), we design a type-specific linear transformation to project the features (or representations) of different types of nodes to a shared feature space. The linear transformation is shown as follows:

$$h'_i = W_{f_{v_i}(u_j)} \cdot h_i,$$

where $h_i$ and $h'_i$ are the original and projected feature of node, respectively. As shown in Figure 2 (a), we transform node features with a type-specific linear transformation for projected features of node with features. Nodes without features or full of noise could be assigned embeddings as trainable vectors, which is equivalent to assigning them with a one-hot vector combined with a linear transformation.

3.2 Heterogeneous Graph Transformation

In previous work, aggregation based on one-hop neighbor usually applies the graph convolution layer in the original graph, which implicitly selects the one-hop (relation) receptive field. And aggregation based on meta-path neighbor is usually done on constructed
meta-path subgraphs, which explicitly selects the multi-hop (meta-
path) receptive field. Relation subgraphs are special meta-path sub-
graphs (note that the original graph is a special case of relation subgraphs). To unify both, we propose a component to abstract the
selection procedure of the receptive field, which determines which
nodes are aggregated. Besides, the component decouples the selec-
tion procedure of receptive field and message passing procedure
introduced in the following subsection.

As shown in Figure 2 (b), we therefore designate a separate stage
called \textit{Heterogeneous Graph Transformation} for graph construction,
and categorize it into (i) relation subgraph extraction that extracts
the adjacency matrices of the specified relations, (ii) meta-path
subgraph extraction that constructs the adjacency matrices based
on the pre-defined meta-paths, (iii) mixed subgraph extraction that
builds both kinds of subgraphs, (iv) homogenization of the hetero-
geous graph (but still preserving \( e_{ij} \)).

3.3 Heterogeneous Message Passing Layer

In Section 2.1, we introduce a conventional way to classify HGNNs.
However, this classification did not find enough commonality from
the implementation perspective, resulting in difficulties in defining
design space and searching for new models. Therefore, we instead
propose to categorize models by their aggregation methods.

Table 3: Direct-aggregation with attention mechanism.

| Mechanisms       | Attention Coefficients \( e_{ij} \) |
|------------------|-------------------------------------|
| GAT [47]         | LeakyReLU \( a^T \begin{bmatrix} Wh_i \mid Wh_j \end{bmatrix} \) |
| HGAT [34]        | LeakyReLU \( a^T \alpha_{T(j)} \begin{bmatrix} h_i \mid h_j \end{bmatrix} \) |
| HetSANN [22]     | LeakyReLU \( a^T \begin{bmatrix} W_{T(i)T(j)} h_i \mid W_{T(j)} h_j \end{bmatrix} \) |
| HGT [24]         | \( W_{Q(i)h_i} W_{A^{TT}} (W_{K(j)h_j})^T \) |
| Simple-HGN [36]  | LeakyReLU \( a^T \begin{bmatrix} Wh_i \mid Wh_j \mid W_{rT \phi(i,j)} \end{bmatrix} \) |

3.3.1 Direct-aggregation. The aggregation procedure is to re-
duce neighbors directly without distinguishing node types. The
basic baselines of HGNN models are GCN, GAT, and other GNNs
used in the homogeneous graph. A recent work\[36\] shows that
the simple homogeneous GNNs, e.g., GCN and GAT, are largely
underestimated due to improper settings.

As shown in Figure 2 (c: the left one), we will explain it under
the message passing GNNs formulation and take GAT [47] as an
example. The message function is \( \phi = a_{ij} h_i^{(L)} \), \( j \in N_i \). The feature
of node \( i \) in \((L+1)\)-th layer is defined as

\[
h_i^{L+1} = \sigma \left( \sum_{j \in N_i} a_{ij} Wh_j^L \right),
\]

where \( W \) is a trainable weight matrix, \( N_i \) is neighbors of node \( i \) and
\( a_{ij} \) is the normalized attention coefficients between node \( i \) and
\( j \), defined by that:

\[
a_{ij} = \frac{\exp \left( e_{ij} \right)}{\sum_{k \in N_i} \exp \left( e_{ik} \right)}. \tag{3}
\]

The correlation of node \( i \) with its neighbor \( j \in N_i \) is represented
by attention coefficients \( e_{ij} \). Changing the form of \( e_{ij} \) yields other
heterogeneous variants of GAT, which we summarize in Table 3.

3.3.2 Dual-aggregation. Following \[56\], we define two parts of
dual-aggregation: micro-level (intra-type) and macro-level (inter-
type) aggregation. As shown in Figure 2 (c: the right one), micro-
level aggregation is to reduce node features within the same rela-
tion, which generate type-specific features in relation/meta-path
subgraphs, and macro-level aggregation is to reduce type-specific
features across different relations. When multiple relations have
the same destination node types, their type-specific features are
aggregated by the macro-level aggregation.

Generally, each relation/meta-path subgraph utilizes the same
micro-level aggregation (e.g., graph convolution layer from GCN
or GAT). In fact, we can apply different homogeneous graph con-
volutions for different subgraphs in our framework. The multiple
homogeneous graph convolutions combined with macro-level ag-
gregation is another form of heterogeneous graph convolution com-
pared with heterogeneous graph convolution in direct-aggregation.
There is a minor difference between the heterogeneous graph con-
volution of direct-aggregation and that of dual-aggregation. We
modify Eq. 3 and define it in Eq. 4, where \( N_{Q(i)h_j} \) means the
neighbors type of node \( i \) is the same as type of node \( j \).

\[
a_{ij} = \frac{\exp \left( e_{ij} \right)}{\sum_{k \in N_{Q(i)h_j}} \exp \left( e_{ik} \right)}. \tag{4}
\]

Example: HAN \[51\] and HGConv \[56\]. In HAN, the node-
level attention is equivalent to a micro-level aggregation with GAT-
Conv, and the semantic-level attention is macro-level aggregation
with attention, which is the same with HGConv. The HGConv
uses the relation subgraphs, which means aggregating the one-hop
neighbors, but HAN extracts multiple meta-path subgraphs, which
means aggregating multi-hop neighbors. According to \textit{Hetero-
genous Graph Transformation} in Section 3.2, the graph constructed
can be a mixture of meta-path subgraphs and relation subgraphs.
So the dual-aggregation can also be operated in a mixture custom
of subgraphs to aggregate different hop neighbors.

4 DESIGN SPACE FOR HETEROGENEOUS
GRAPH NEURAL NETWORK

Inspired by GraphGym \[55\], we propose a design space for HGNN,
which is built as a platform \textit{Space4HGN} offering modularized
HGNN implementation for researchers introduced at last.

4.1 Designs in HGNN

As illustrated in Figure 3, we will describe it from two aspects:
common designs with GraphGym and unique designs distilled from
HGNNs.

4.1.1 Common Designs with GraphGym. The common designs
with GraphGym involves 12 design dimensions, categorized three
Figure 3: Design space: (1) The red and yellow font represents the design dimension and the blue font means choice of model families. (2) The red dotted frame includes the dimensions of the unique design in HGNNs. And blue dotted frame indicates some common dimensions with GraphGym.

Table 4: Common design dimensions with GraphGym.

| Design Dimension   | Choices                  |
|--------------------|--------------------------|
| Batch Normalization| True, False              |
| Dropout            | 0, 0.3, 0.6              |
| Activation         | Relu, LeakyRelu, Elu, Tanh, PReLU |
| L2 Normalization   | True, False              |
| Layer Connectivity | STACK, SKIP-SUM, SKIP-CAT|
| Pre-process Layers | 1, 2, 3                  |
| Message Passing Layers | 1, 2, 3, 4, 5, 6       |
| Post-process Layers| 1, 2, 3                  |
| Optimizer          | Adam, SGD                |
| Learning Rate      | 0.1, 0.01, 0.001, 0.0001 |
| Training Epochs    | 100, 200, 400            |
| Hidden dimension   | 8, 16, 32, 64, 128       |

Table 5: Unique design dimensions in HGNNs.

| Design Dimension   | Choices                                      |
|--------------------|----------------------------------------------|
| Model Family       | Homogenization, Relation, Meta-path          |
| Micro-level Aggregation (Graph Convolution Layer) | GCNConv, GATConv, SageConv, GINConv |
| Macro-level Aggregation | Mean, Max, Sum, Attention                   |

Aspects, intra-layer, inter-layer and training settings. The dimensions with corresponding choices are shown in Table 4. More detailed description is provided in Appendix B

4.1.2 Unique Design in HGNNs. With the unified framework, we try to transform the modular components into unique design dimensions in HGNNs. According to [39], a collection of related neural network architectures, typically sharing some high-level architectural structures or design principles (e.g., residual connections), could be abstracted into a model family. With that, we distill three model families in HGNNs.

The Homogenization Model Family. The homogenization model family uses the direct-aggregation combined with any graph convolutions. Here we use the term *homogenization* because all HGNNs included here apply direct-aggregation after the homogenization of the heterogeneous graph mentioned in Section 3.2. Homogeneous GNNs and heterogeneous variants of GAT mentioned in Section 3.3 all fall into this model family. The homogeneous GNNs are usually evaluated as basic baselines in HGNN papers. Though it loses type information, it is confirmed that the simple homogeneous GNNs can outperform some existing HGNNs [36], which means they are nonnegligible and supposed to be seen as a model family. We select four typical graph convolution layers, which are GraphConv [31], GATConv [47], SageConv-mean [19] and GINConv [53] as analyzed candidates.

The Relation Model Family. The model family applies relation subgraph extraction and dual-aggregation. The first HGNN model RGCN [41] is a typical example in relation model family, whose dual-aggregation consists of a micro-level aggregation with SageConv-mean and macro-level aggregation of Sum. HGConv [56] is a combination of GATConv and attention. We could get other designs by enumerating the combinations of micro-level and macro-level aggregation. In our experiments, we set the micro-level aggregations the same as graph convolutions in the homogenization model family, and macro-level aggregations are chosen among Mean, Max, Sum, and Attention.

The Meta-path Model Family. The model family applies meta-path subgraph extraction and dual-aggregation. The instance HAN [51] has the same dual-aggregation with HGConv [56] in the relation model family but different subgraph extraction. The candidate of micro-level and macro-level aggregations is the same as those in the relation model family.
### 4.2 Space4HGNN: Platform for Design Space in HGNN

We developed Space4HGNN, a novel platform for exploring HGNN designs. We believe Space4HGNN can significantly facilitate the research field of HGNNs. It is implemented with PyTorch\(^3\) and DGL\(^4\), using the OpenHGNN\(^4\) package. It also offers a standardized evaluation pipeline for HGNNs, much like [55] for homogeneous GNNs. For faster experiments, we offer parallel launching. Its highlights are summarized below.

#### 4.2.1 Modularized HGNN Implementation.

The implementation closely follows the GNN design space GraphGym. It is easily extendable, allowing future developers to plug in more choices of design dimensions (e.g., a new graph convolution layer or a new macro-aggregation). Additionally, it is easy to import new design dimensions to Space4HGNN, such as score function in link prediction.

#### 4.2.2 Standardized HGNN Evaluation.

Space4HGNN offers a standardized evaluation pipeline for diverse architecture designs and HGNN models. Benefiting from OpenHGNN, we can evaluate diverse datasets in different tasks easily and offer visual comparison results presented in Section 5.

### 5 EXPERIMENTS

#### 5.1 Datasets

We select the Heterogeneous Graph Benchmark (HGB) [36], a benchmark with multiple datasets of various heterogeneity (i.e., the number of nodes and edge types). To save time and submission resources, we report the test performance of the configuration with best validation performance in Table 8. Other experiments are evaluated on a validation set with three random 80-20 training-validation splits. The statistics of HGB are shown in Table 6.

#### 5.2 Evaluation Technique

Our design space covers over 40M combinations, and a full grid search will cost too much. We adapt controlled random search from GraphGym [55] setting the number of random experiments to 264, except that we ensure that every combination of dataset, model family, and micro-aggregation receives 2 hits. We draw bar plots and violin plots of rankings of each design choice following the same practice as GraphGym. As shown in Figure 4, in each subplot, rankings of each design choice are aggregated over all 264 setups via bar plot and violin plot. The bar plot shows the average ranking across all the 264 setups (lower is better). The violin plot indicates the smoothed distribution of the ranking of each design choice over all the 264 setups.

#### 5.3 Evaluation of Design Dimensions Common with GraphGym

##### 5.3.1 Overall Evaluation.

The evaluation results of design dimensions common with GraphGym [55] are shown in Figure 4, from which we draw the following conclusions.

**Findings aligned with GraphGym:**

- We also confirmed that BN [26] yields better results, while L2-Norm did not have a significant impact. However, task-wise evaluation of both dimensions in Section 5.3.2 reveals a different and more insightful story.
- There is no definitive conclusion for the best number of message passing layers; each dataset has its own best number, the same as what GraphGym observed.
- The characteristic of training settings (e.g., optimizer and training epochs) is similar to GraphGym.

**Findings different from GraphGym:**

- A single linear transformation (pre-process layer) is usually enough. We think that this is because our heterogeneous linear transformation is node type-specific which has enough parameters to transform representations.
- The widely used activation Relu may no longer be as suitable in HGNNs. Tanh, LeakyReLU, and ELU are better alternatives. PReLU, stood out in GraphGym, is not the best choice in our design space.
- Different from GraphGym, we found that Dropout is necessary to get better performance. We think the reason is that parameters specific to node types and relation types lead to over-parametrization.

##### 5.3.2 Task-wise Evaluation.

We previously observed that BN yields better performance in general. However, task-wise evaluation in Figure 5 showed that **BN is better on link prediction but worse on node classification**. Meanwhile, although L2-Norm does not seem to help in overall performance, it actually performs better on node classification but worse on link prediction. We think that BN scales and shifts nodes according to the global information, which may lead to more similar representations and damage the performance of the node classification task, and L2-Norm scales the representation and thus the link score to \([-1,1]\), which may invalidate the Sigmoid of the score function.

#### 5.4 Evaluation of Unique Design Dimensions in HGNNs

How to design and apply HGNN is our core issue. This section analyzes unique design dimensions in HGNNs to describe the characteristics of high-level architecture designs. From the average ranking shown in Figure 6, we can see that the meta-path model family has a small advantage in the node classification task. The relation model family outperforms in aggregated results in all datasets,

| Dataset | #Nodes | #Node Types | #Edges | #Edge Types | Name for Node Classification Task | Name for Link Prediction Task |
|---------|--------|-------------|--------|-------------|-----------------------------------|-------------------------------|
| DBLP    | 26,128 | 4           | 259,566| 6           | HGBl-DBLP                         | HGBl-DBLP                     |
| IMDB    | 21,420 | 4           | 86,642 | 6           | HGBl-IMDB                         | HGBl-IMDB                     |
| ACM     | 10,942 | 4           | 547,872| 8           | HGBl-ACM                          | HGBl-ACM                      |
| Freebase| 180,098| 8           | 1,057,688| 36          | HGBl-Freebase                     | HGBl-Freebase                 |
| PubMed  | 63,109 | 4           | 244,986| 10          | HGBl-PubMed                       | HGBl-PubMed                   |
| Amazon  | 10,099 | 1           | 148,659| 2           | -                                 | HGBl-amazon                   |
| LastFM  | 20,612 | 3           | 141,521| 3           | -                                 | HGBl-LastFM                   |
5.4.1 The Model Family. To more comprehensively describe the corresponding characteristics of different model families, we analyze the results across datasets as shown in Figure 7 and highlight some findings below.

The meta-path model family helps node classification. In node classification task, the meta-path model family outperforms visibly than the other model families on datasets HGBn-ACM and HGBn-DBLP, where we think some informative and effective meta-paths have been empirically discovered. Some experimental analysis for meta-paths can be found in Appendix D.1.

The meta-path model family does not help link prediction. In previous works, few variants of the meta-path model family were applied to the link prediction task. Although our unified framework can apply the meta-path model family to the link prediction task, the meta-path model family does not perform well on all datasets of link prediction as shown in Figure 7. We think this is because the information from the edges in the original graph is important in link prediction, and the meta-path model family ignores it.
The relation model family is a safer choice in all datasets. From Figure 7, the relation model family stands out in link prediction task, which confirms the necessity to preserve the edges as well as their type information in the original graph. Compared with the homogenization model family, the relation model family has more trainable parameters which have a linear relationship with the number of edge types. Surprisingly, the relation model family is not very effective in HGBn-Freebase with much heterogeneity, which has 8 node types and 36 edge types. We think that too many parameters lead to over-fitting, which may challenge the relation model family. According to the distribution of ranking, the relation model family has a significantly lower probability of being ranked last. Therefore, the relation model family is a safer choice.

The homogenization model family with the least trainable parameters is still competitive. As shown in Figure 7, the homogenization model family is still competitive against the relation model family on HGBn-IMDB, and even outperforms the latter on HGBn-Freebase and HGBn-LastFM. Therefore, the homogenization model family is not negligible as a baseline even on heterogeneous graphs, which aligned with [36].

### 5.4.2 The Micro-aggregation and the Macro-aggregation Design Dimensions

The existing HGNNs are usually inspired by GNNs and apply different micro-aggregation (e.g., GCNConv, GATConv). The micro-aggregation design dimension in our design space brings many variants to the existing HGNNs. As shown in Figure 6, the results of comparison between micro-aggregation vary greatly across tasks. We provide ranking analysis on different datasets in Appendix D.2.

For the macro-aggregation design dimension, Figure 6 shows that Sum has a great advantage in both tasks, which is aligned with the theory that **Sum aggregation is theoretically most expressive** [53]. Surprisingly, Attention is not so effective as Sum, and we think the micro-aggregation is powerful enough, resulting that complicated Attention in macro-aggregation is not necessary.

### 5.5 Evaluation of Condensed Design Space

Above experiments reveals that it is hard to design a single HGNN model that can guarantee outstanding performance across diverse scenarios in the real world. According to the findings in Section 5.3.1, we condensed the design space to facilitate model searching. Specifically, we remove some bad choices in design dimensions and retain some essential design dimensions (e.g., high-level architectural structures and helpful design principles). The evaluation of HGB shows that a simple random search in the condensed design space can find the best designs. More experimental results compared GraphGym [55] and GraphNAS [14] are analyzed in Appendix E.

#### 5.5.1 The Condensed Design Space

For common design dimensions with GraphGym, Table 7 compares the condensed design spaces we and GraphGym proposed. We retain some of the design dimensions same as GraphGym if the findings in Section 5.3.1 are aligned (i.e., layer connectivity, optimizer, and training epochs). We propose our own choices for the different dimensions in conclusion (e.g., Dropout, activation, BN, L2-Norm, etc.). For unique design dimensions in HGNN, we conclude that the micro-aggregation and

---

**Table 7: Condensed common design dimensions with GraphGym. Unique dimensions in HGNNs are not condensed.**

| Design Dimension               | Our Condensed Design Space | Condensed Design Space in GraphGym |
|-------------------------------|----------------------------|-----------------------------------|
| Batch Normalization           | True, False                | True                              |
| Dropout                       | 0, 0.3                     | 0                                 |
| L2 Normalization              | ELU, LeakyReLU, Tanh, True, False | PReLU                             |
| Layer Connectivity            | SKIP-SUM, SKIP-CAT         | SKIP-SUM, SKIP-CAT                |
| Pre-process Layers            | 1                          | 1, 2                              |
| Message Passing Layers        | 1, 2, 3, 4, 5, 6           | 2, 4, 6, 8                       |
| Post-process Layers           | 1, 2                       | 2, 3                              |
| Optimizer                     | Adam                       | Adam                              |
| Learning Rate                 | 0.1, 0.01                  | 0.01                              |
| Training Epochs               | 400                        | 400                               |
| Hidden dimension              | 64, 128                    | -                                 |
Table 8: Comparison with the standard HGNNs in HGB. The prefix HGBn means dataset of node classification and the prefix HGBI means dataset of link prediction. Vacant positions (-) are due to lack of baselines in HGB. The lower average rank is better.

| Model Family | HGBn-DBLP | HGBn-ACM | HGBI-amazon | HGBI-LastFM |
|--------------|-----------|----------|-------------|-------------|
|              | Macro-F1  | Micro-F1 | ROC-AUC     | MRR         |
| GCN          | 90.84±0.32| 91.47±0.34| 92.17±0.24  | 92.84±0.34  |
| GAT          | 93.83±0.27| 93.39±0.30| 92.26±0.94  | 91.65±0.80  |
| RGCN         | 91.52±0.50| 92.07±0.50| 91.55±0.74  | 86.34±0.28  |
| HAN          | 91.67±0.49| 92.05±0.62| 90.89±0.43  | -           |
| HGT          | 93.01±0.23| 93.49±0.25| 91.12±0.76  | 88.26±2.06  |
| Simple-HGN   | 94.01±0.24| 94.46±0.22| 93.42±0.44  | 93.49±0.62  |
| Ours         | 94.24±0.42| 94.63±0.40| 92.50±0.14  | 95.15±0.43  |

Table 9: Best designs in HGB, found by a simple random search in our condensed design space.

| Dataset       | Model Family | Aggregation | Macro-aggregation | BN | L2-Norm | Dropout | Activation | Layer Connectivity | Message Passing Layers | Post-process Layers |
|---------------|--------------|-------------|------------------|----|---------|---------|------------|-------------------|-----------------------|---------------------|
| HGBn-ACM      | Relation     | GCNConv     | Sum              | False | True        | 0.0     | Tanh       | SKIP-CAT          | 2                     | 2                   |
| HGBn-DBLP     | Homogenization | GATConv     | -                | True  | True        | 0.0     | LeakyRelu  | SKIP-CAT          | 3                     | 2                   |
| HGBI-amazon   | Meta-path    | GATConv     | Attention        | True  | False       | 0.0     | LeakyRelu  | SKIP-CAT          | 5                     | 1                   |
| HGBl-LastFM   | Homogenization | SageConv    | -                | True  | False       | 0.3     | Elu        | SKIP-CAT          | 5                     | 2                   |

model family design dimensions vary greatly across datasets or tasks. So we retain all choices in unique design dimensions and aim to find out whether the variants of existing HGNNs could gain improvements in HGB.

The original design space contains over 40M combinations, and the condensed design space contains 70K combinations. So the possible combination of the design dimensions in condensed design space is reduced by nearly 500 times.

5.5.2 Evaluation in Heterogeneous Graph Benchmark (HGB). To compare with the performance of the standard HGNNs, we evaluate our condensed design space in a new benchmark HGB. We randomly searched 100 designs from condensed design space and evaluated the best design of validation set in HGB. As shown in Table 8, our designs with condensed design space can achieve comparable performance. So we can easily achieve SOTA performance with a simple random search in the condensed design space. Table 9 shows the best designs we found in our condensed design space, which cover the variants of RGCN and HAN. It also confirms that the meta-path model family and the relation model family have great performance in HGB and answers the question "are meta-path or variants still useful in HGNN?" from [36]. Note that this result does not contradict the conclusion from [36], as our design space includes much more components than the vanilla RGCN or HAN model, and proper components can make up shortcomings of an existing model.

6 CONCLUSION AND DISCUSSION

In this work, we propose a unified framework of HGNN and define a design space for HGNN, which offers us a module-level sight to evaluate HGNN models. Specifically, we comprehensively analyze the common design dimensions with GraphGym and the unique design dimensions in HGNN. After that, we distill some findings and condense the original design space. Finally, experimental results show that our condensed design space outperforms others, and gains the best average ranking in a benchmark HGB. With that, we demonstrate that focusing on the design space could help drive advances in HGNN research.

How to condense the design space? In our work, the condensed design space is distilled according to the findings within extensive experiments, which still needs much effort and intuitive experience. A recently proposed work KGTuner [59], which analyzes the design space for knowledge graph embedding, proposed a more systematic way to shrink and decouple the search space, which can be a potential improvement of this work.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China (No. U20B2045, 62192784, 62002029, 62172052, 61772082).

REFERENCES

[1] Ye Bi, Liqiang Song, Mengqiu Yao, Zhenyu Wu, Jianming Wang, and Jing Xiao. 2020. A heterogeneous information network based cross domain insurance recommendation system for cold start users. In SIGIR. 2211–2220.
A PRELIMINARY

A.1 Graph Neural Network

Graph Neural Networks (GNNs) aim to apply deep neural networks to graph-structured data. Here we focus on message passing GNNs which could be implemented efficiently and proven great performance.

Definition A.1 (Message Passing GNNs [16]). Message passing GNNs aim to learn a representation vector \( h_v^{(L)} \in \mathbb{R}^{d_l} \) for each node \( v \) after \( L \)-th message passing layers of transformation, and \( d_l \) means the output dimension in \( L \)-th message passing layer. The message passing paradigm defines the following node-wise and edge-wise computation for each layer as:

Edge-wise: \( m_{e_{ij}}^{(L+1)} = \phi \left( h_i^{(L)}, h_j^{(L)} \right) \), \( i \in \mathcal{N}_i \), \( j \in \mathcal{N}_j \),

where \( \mathcal{N}_i \) means neighbors of node \( v_i \), \( \phi \) is a message function defined on each edge to generate a message by combining the features of its incident nodes, and \( e_{ij} \) denotes an edge from node \( v_j \) to \( v_i \);

Node-wise: \( h_i^{(L+1)} = \psi \left( h_i^{(L)}, \rho \left( \{ m_{e_{ij}}^{(L+1)} : j \in \mathcal{N}_i \} \right) \right) \),

where \( \mathcal{N}_i \) means neighbors of node \( v_i \), \( \psi \) is an update function defined on each node to update the node representation by aggregating its incoming messages using the aggregation function \( \rho \).

Example: GraphSAGE [19] can be formalized as a message passing GNN, where the message function is \( \phi = h_j^{(L)} \psi(W^{(L)}) \) and the update function is \( \psi = \text{SUM} \left( \{ m_{e_{ij}}^{(L+1)} : j \in \mathcal{N}_i \} \right) \).

A.2 Heterogeneous Graph

Definition A.2 (Heterogeneous Graph). A heterogeneous graph, denoted as \( G = (\mathcal{V}, E) \), consists of a node set \( \mathcal{V} \) and an edge set \( E \). A heterogeneous graph is also associated with a node type mapping \( f_v : \mathcal{V} \rightarrow T^v \) and an edge type (or relation type) mapping function \( f_e : E \rightarrow T^e \). \( T^v \) and \( T^e \) denote the sets of node types and edge types. Each node \( v_i \in \mathcal{V} \) has one node type \( f_v(v_i) \in T^v \). Similarly, for an edge \( e_{ij} \in E \) from node \( i \) to node \( j \), \( f_e(e_{ij}) \in T^e \). When \( |T^v| > 1 \) or \( |T^e| > 1 \), it is a heterogeneous graph, otherwise it is a homogeneous graph.

Example. As shown in Figure 1 (left), we construct a simple heterogeneous graph to show an academic network. It consists of multiple types of objects (Paper(P), Author(A), Conference(C)) and relations (written-relation between papers and authors, published-relation between papers and conferences).

Definition A.3 (Relation Subgraph). A heterogeneous graph can also be represented by a set of adjacency matrices \( \{ A_k \}_{k=1}^{K} \), where \( K \) is the number of edge types \( |T^e| \). \( A_k \in \mathbb{R}^{N \times N} \) is an adjacency matrix where \( A_k[i,j] \) is non-zero when there is an edge \( e_{ij} \) with \( k \)-th type from node \( v_i \) to node \( v_j \). \( N_v \) and \( N_e \) are numbers of source and target nodes corresponding to the edge type \( k \) respectively. A relation subgraph of \( k \)-th edge type is therefore a subgraph whose adjacency matrix is \( A_k \). As shown in Figure 2 (b), the underlying data structure of the academic network in Figure 1 (left) consists of four adjacency matrices.

Definition A.4 (Meta-path [45]). A meta-path \( P \) is defined as a path in the form of \( v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \ldots \rightarrow v_{l+1} \) which describes a composite relation \( r_1 \circ r_2 \circ \cdots \circ r_l \) between two nodes \( v_1 \) and \( v_{l+1} \), where \( r_i \in T^e \) denotes the \( i \)-th relation type of meta-path and \( \circ \) denotes the composition operator on relations.

Definition A.5 (Meta-path Subgraph). Given a meta-path \( P, r_1 \circ r_2 \circ \cdots \circ r_l \), the adjacency matrix \( A_P \) can be obtained by a multiplication of adjacency matrices according to relations as:

\[ A_P = A_{r_1} \cdots A_{r_l} \cdots A_{r_{l+1}}. \]
Table 10: Statistics of HGB datasets. The prefix HGBn presents datasets in the node classification task and target node with number of classes is for these datasets. The prefix HGBl presents datasets in the link prediction task and target link is for these datasets.

| Dataset   | #Nodes | #Node Types | #Edges | #Edge Types | Name for Node Classification Task | Target Node | #Classes | Name for Link Prediction Task | Target Link for Link Prediction |
|-----------|--------|-------------|--------|-------------|-----------------------------------|-------------|----------|-----------------------------|---------------------------------|
| DBLP      | 26,128 | 4           | 239,566| 6           | HGBl-DBLP                        | author      | 4        | HGBl-DBLP                   | author-paper                    |
| IMDB      | 21,420 | 4           | 86,642 | 6           | HGBl-IMDB                         | movie       | 5        | HGBl-IMDB                   | actor-movie                     |
| ACM       | 10,942 | 4           | 547,872| 8           | HGBl-ACM                          | paper       | 3        | HGBl-ACM                    | paper-paper                     |
| Freebase  | 180,098| 8           | 1,057,688| 36         | HGBl-Freebase                     | book        | 7        | HGBl-Freebase               | -                               |
| PubMed    | 63,109 | 4           | 244,986| 10          | HGBl-PubMed                       | disease     | 8        | HGBl-PubMed                 | disease-disease                 |
| Amazon    | 10,099 | 1           | 148,659| 2           | HGBl-amazon                       |             | -        | HGBl-amazon                 | product-product                 |
| LastFM    | 20,612 | 3           | 141,521| 3           | HGBl-LastFM                       |             | -        | HGBl-LastFM                 | user-artist                     |

Table 11: Homogeneous subgraph extracted by meta-paths or relations and the corresponding homophily $\beta$ (bold is highest in the dataset).

| Dataset   | Meaning | Meta-path | $\beta$ |
|-----------|---------|-----------|---------|
| HGBn-ACM  | A: author P: paper S: subject c: citation relation r: reference relation | PrP, PAP, PSB | *0.6311* |
|           |         | BOFB      | 0.4204  |
| HGBn-DBLP | A: author P: paper T: term V: venue | APA, APTPA, APVPA | *0.7564* |
|           |         | DGD       | 0.3896  |
| HGBn-PubMed| D: disease G: gene C: chemical S: species | DD, DCD, DGD | 0.2567 |
|           |         | DSB       | 0.2477  |
| HGBn-Freebase | B: book F: film L: location M: music P: person S: sport O: organization U: business | BB, BUB, BFB, BLMB, BOFB, BFB | *0.3341* |

D.1 Analysis for Meta-path

In node classification task, the meta-path model family outperforms visibly than the other model families on datasets HGBn-ACM and HGBn-DBLP, where we think some informative and effective meta-paths have been empirically discovered. The micro-aggregation modules are MPNN networks that tend to learn similar representations for proximal nodes in a graph [38]. Moreover, the meta-path model family aims to bring nodes with the same type topologically closer with meta-path subgraph extraction, hoping that the extracted subgraph is assortative (e.g., citation networks) where node homophily holds (i.e., nodes with the same label tend to be proximal, and vice versa). Based on that, we measure the homophily [38] in subgraphs extracted by meta-path $\mathcal{P}$, which is defined as

$$
\beta = \frac{1}{|V|} \sum_{v \in V} \frac{|\{u : A_P[u, v] = 1, y_u = y_v\}|}{|\{u : A_P[u, v] = 1\}|},
$$

where $y_u$ and $y_v$ represent the label of node $u$ and $v$, respectively.

As shown in Table 11, the homophily of homogeneous subgraphs extracted by predefined meta-path in HGBn-ACM and HGBn-DBLP is significantly higher than that in HGBn-PubMed and HGBn-Freebase. For node classification task, the homophily of subgraphs extracted by meta-paths may be a helpful reference for meta-path selection. So for the question “are meta-path or variants still useful in GNNs?” from [36], we think that the meta-path model family is still useful with well-defined meta-paths that reveal task-specific semantics.

D.2 Analysis for Micro-aggregation

As shown in Figure 8, the results of comparison between micro-aggregation vary greatly across datasets. The GCNConv has gained significant advantages on datasets HGBl-amazon and HGBl-LastFM. The GATConv performs best on the two datasets. The SOTA model GINConv for graph-level tasks can also stand out in one dataset HGBl-DBLP here. It confirms that there is no single GNN model can perform well in all situations.
Figure 8: Ranking analysis for the micro-aggregation design dimension on different datasets. Different micro-aggregation vary greatly across datasets.

Figure 9: Distribution estimates for different design space. Curves closer to the lower-right corner indicates a better design space. The vertical dashed line indicates the best performance GraphNAS can get.

E EVALUATION OF CONDENSED DESIGN SPACE

E.1 Evaluation of Different Design Spaces
Plotting ranking with controlled random search can only work in the same design space, and is not suitable for evaluation across different design spaces. Therefore, we plot for each design space the empirical distribution function (EDF) [39]: given $n$ configurations and their respective scores $s_i$, EDF is defined as

$$F(s) = \frac{1}{n} \sum_{i=1}^{n} 1[s_i < s].$$

EDF essentially tells the probability of a random hyperparameter configuration that cannot achieve a given performance metric. Therefore, with $x$-axis being the performance metric and $y$-axis being the probability, an EDF curve closer to the lower-right corner indicates that a random configuration is more likely to get a better result.

Though the condensed design space in GraphGym is small enough to perform a full grid search for GNNs, it is not so much suitable for HGNNs due to more complicated HGNN models. To verify the effectiveness of our condensed design space, we compare it with the original design space and the condensed design from GraphGym [55]. We randomly search 100 designs in three spaces, respectively.

As shown in Figure 9, our condensed design space outperforms the others. Specifically, the original design space has many bad choices (i.e., optimizer with SGD) and performs worst in the distribution estimates. On the other hand, the best design in the original design space is competitive, but at a much higher search cost. Besides, the better performance in our condensed design space compared with GraphGym shows that we cannot simply transfer the design space condensed from homogeneous graphs to HGNNs, and specific condensation is required.

E.2 Comparison with GraphNAS
We also apply a GNN neural architecture search method GraphNAS [14] in the original design space as a comparison. The NAS is to find the best architecture, so we only report the best performance of GraphNAS in Figure 9. Though GraphNAS outperforms in HGBn-DBLP and gains excellent performance in HGBn-ACM and HGBn-IMDB, it performs worst in other datasets. So compared with GraphNAS, our design space has more significant advantages in robustness and stability. We think that we need a more advanced NAS method (i.e., DiffMG [8]) for our design space in future work.