# Reinforcement Learning Enhanced Heterogeneous Graph Neural Network

Zhiqiang Zhong
University of Luxembourg
Esch-sur-Alzette, Luxembourg
zhiqiang.zhong@uni.lu

Cheng-Te Li
National Cheng Kung University
Tainan, Taiwan
chengte@mail.ncku.edu.tw

Jun Pang
University of Luxembourg
Esch-sur-Alzette, Luxembourg
jun.pang@uni.lu

## ABSTRACT

Heterogeneous Information Networks (HINs), involving a diversity of node types and relation types, are pervasive in many real-world applications. Recently, increasing attention has been paid to heterogeneous graph representation learning (HGRL) which aims to embed rich structural and semantics information in HIN into low-dimensional node representations. To date, most HGRL models rely on manual customisation of meta paths to capture the semantics underlying the given HIN. However, the dependency on the handcrafted meta-paths requires rich domain knowledge which is extremely difficult to obtain for complex and semantic rich HINs. Moreover, strictly defined meta-paths will limit the HGRL's access to more comprehensive information in HINs. To fully unleash the power of HGRL, we present a **R**einforcement **L**earning enhanced **H**eterogeneous **G**raph **N**eural **N**etwork (RL-HGNN), to design different meta-paths for the nodes in a HIN. Specifically, RL-HGNN models the meta-path design process as a Markov Decision Process and uses a policy network to adaptively design a meta-path for each node to learn its effective representations. The policy network is trained with deep reinforcement learning by exploiting the performance of the model on a downstream task. We further propose an extension, RL-HGNN++, to ameliorate the meta-path design procedure and accelerate the training process. Experimental results demonstrate the effectiveness of RL-HGNN, and reveals that it can identify meaningful meta-paths that would have been ignored by human knowledge.

## 1 INTRODUCTION

The complex interactions in real-world data, such as social networks, biological network and knowledge graphs, can be modelled as Heterogeneous Information Networks (HINs) [29], which are commonly associated with multiple types of objects (nodes) and meta-relations. Taking the heterogeneous academic network depicted in Figure 1-(a) as an example, it involves four types of nodes including papers, authors, institutions and publication venues, and eight types of meta-relations between these nodes. Due to the capability of HINs to retain the rich and complex inter-dependency between nodes, they have attracted increasing attention from the research community and have recently been applied in the fields of recommendation system [17, 39], information retrieval [26, 36] and etc. However, the complex semantics and the non-Euclidean nature of HINs render them difficult to be modelled by traditional machine learning models.

Over the past decade, a significant line of research on HINs is *heterogeneous graph representation learning* (HGRL). The goal of HGRL is to learn node latent representations, which encode graph structure, multi-typed nodes and meta-relations, for downstream tasks including link prediction [37], node classification [16, 38] and node clustering [10]. As discussed in a recent survey [7], one of the conventional paradigms on HGRL is to manually define and use meta-paths to model HIN's structures and leverage random walks to transform graph structure into sequences [6, 9], which can be explored by shallow sequence-based embedding learning algorithms, such as DeepWalk [23] and node2vec [12]. A meta-path scheme of HIN is defined as a sequence of meta relations over HIN's network schema. For instance, an illustrative meta-path in the heterogeneous academic network in Figure 1-(a) is $Author \xrightarrow{Cite} Author \xrightarrow{Write} Paper$. The "shallow" methods are characterised as an embedding lookup table, meaning that they directly encode each node as a vector, and this embedding table is the parameter to optimise.

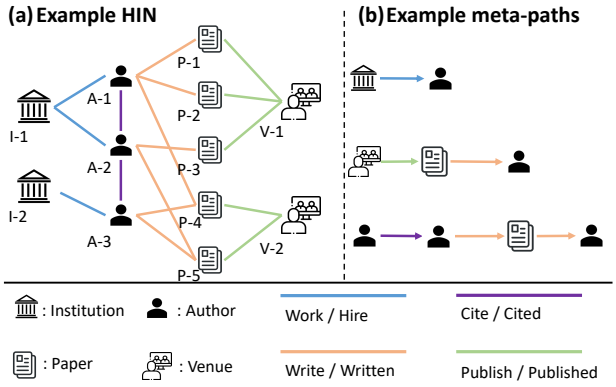


**Figure 1: A toy example of HIN and some possible meth-paths: (a) an academia HIN; (b) some example meta-paths exported from (a).**

Recently, in view of the impressive success of Graph Neural Networks (GNNs) [13, 19, 34, 41], the second paradigm of HGRL attempts to design Heterogeneous Graph Neural Networks (HGNNs) for HGRL [10, 16, 25, 38, 42], which extend graph convolutional operations on HIN with pre-defined meta-paths. Compared with the "shallow" methods for HGRLs, HGNNs support an end-to-end training mechanism which can learn node representations with the guidance of some labelled nodes. They are also empowered by more complex encoders, instead of using the shallow embedding learning algorithms. The encoders usually have the form of a deep neural network, enabling the natural modelling of both structure and node attributes in HINs.

**Key Challenges in HGRL**. Nevertheless, the task of node representation learning on HINs is still non-trivial, mainly due to its dependency on the handcrafted meta-paths. For instance, the design

of meta-paths requires rich domain knowledge which is extremely difficult to obtain in complex and semantic rich HINs [7]. More specifically, given a HIN $G$ with a node type set $\mathcal{N}$, a relation type set $\mathcal{R}$ and a fixed meta-path length $T$. The possible meta-paths are contained in a set of size $(|\mathcal{N}|\times|\mathcal{R}|)^T$. Such a huge set can result in a combinatorial explosion when increasing the scales of $|\mathcal{N}|$, $|\mathcal{R}|$ and $T$. Second, the representation capacities of manually designed meta-paths are limited to a specific task on specific data, since different $G$ with the same node type set $\mathcal{N}$ and relation type set $\mathcal{R}$ may have different node attributes and meta-relation distributions. It requires to design appropriate meta-paths for each task on each dataset which is extremely difficult for practical applications. Moreover, existing meta-path guided methods simply assume that different nodes/relations of the same type share the same meta-path, which ignores the differences among individual nodes. Take the heterogeneous academic networks as an example (Figure 1-(a)), assume we plan to learn node representations to determine the research area of *Author*s. A meta-path $\Omega_1$ : $Author \xrightarrow{Write} Paper \xrightarrow{Published} Venue$ may be useful to learn a representation of a senior researcher, since his/her published papers and attended venues may provide sufficient information to decide his/her research area. While learning the representation of a junior PhD candidate with just a few published papers, we may need to extract information from his collaborators following the meta-path: $\Omega_2$ : $Author \xrightarrow{Cite} Author \xrightarrow{Write} Paper$, because the previous meta-path $\Omega_1$ retains little information in the case of PhD candidates. Hence, we argue that we should design an appropriate meta-path for each node according to its own attributes and structure properties, instead of giving each node type several pre-defined meta-paths in general. It is critical to developing an adaptive strategy to design appropriate meta-paths for nodes in a HIN, which still remains a challenge in the area of HGRL.

In light of these identified limitations and challenges, we propose to investigate HGRL with the objectives of (i) learning to create an appropriate meta-path for each node in HINs automatically, (ii) generating node representations efficiently with the automatically designed meta-paths, and (iii) retaining the end-to-end training strategy. To achieve these objectives, we present our framework of **R**einforcement **L**earning enhanced **H**eterogeneous **G**raph **N**eural **N**etwork (RL-HGNN), to fully unleash the power of HGRL.

**Key Ideas of RL-HGNN**. Generally, we aim to replace the human efforts on meta-path design by a Reinforcement Learning (RL) agent to address the limitations of the dependency on handcrafted meta-paths of HGNNs. Compared with experts with domain knowledge, the RL agent can adaptively design appropriate meta-paths for each node in terms of a specific task, through sequential exploration and exploitation. Both network structure and node attributes are considered while designing meta-paths, and this is attractive and practicable for HINs with complex semantics.

As illustrated in Figure 2, the meta-path design process can be naturally considered as a Markov Decision Process (MDP) in which the next relation to extend the meta-path depends on the current state in a meta-path. Then we propose an HGNN model to perform information aggregation on the designed meta-paths to learn node representations which can be applied to a downstream task, such as node classification. Motivated by the recent success of RL [21, 22], we propose to employ a policy network (agent) to solve this MDP, and use an RL algorithm enhanced by a reward function to train the policy network. The reward function is defined as the performance improvement on the downstream task, which encourages the RL agent to achieve better and stable performance. In addition, we find that there exists a large computational redundancy during the classic information aggregation on meta-paths, thus we introduce an optimal aggregation strategy to achieve redundancy-free information aggregation on HINs.

We showcase an instance of our framework, i.e. RL-HGNN, by implementing it with a classic RL algorithm, i.e., Deep Q-learning (DQN) [22], and its extension RL-HGNN++. RL-HGNN finds a meta-path for each node according to the node's attributes, while RL-HGNN++ further enables the meta-path design to explore structural semantics of HIN and as well consider encoder's status. It also significantly accelerates the HGRL process based on a theoretical complexity analysis.

In order to validate the effectiveness of the proposed framework, we conduct extensive experiments on two HINs and the experimental results demonstrate that our framework outperforms the state-of-the-art models.

**Main Contributions**. We summarise our contributions in this paper as follows:

(1) We present an RL-enhanced framework, RL-HGNN[1], to learn node representations from an HIN without the need of manual meta-path design.
(2) We design a redundancy-free heterogeneous information aggregation method to reduce redundant computation during information aggregation on meta-paths.
(3) We develop two practical instances of our framework, RL-HGNN and RL-HGNN++ which improves the meta-path design procedure and accelerates the training process.
(4) We conduct a node classification task with different experimental settings to evaluate the performance of our models. Experimental results show the effectiveness of the two instances of our framework, both can identify meaningful meta-paths that have been ignored by human experts.

## 2 PRELIMINARIES

### 2.1 Problem Statement

**Definition 1.** (**Heterogeneous Information Network**): A Heterogeneous Information Network (HIN) is an information network with multiple types of nodes and edges, denoted as a graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{N}, \mathcal{R})$, which consists of a node set $\mathcal{V}$, a relation set $\mathcal{E}$, a node type set $\mathcal{N}$, and a relation type set $\mathcal{R}$. Each node $v \in \mathcal{V}$ and each edge $e \in \mathcal{E}$ are associated with their type mapping function $\phi(v) : \mathcal{V} \rightarrow \mathcal{N}$ and $\varphi(e) : \mathcal{E} \rightarrow \mathcal{R}$, respectively. Node attribute of $v_i$ is denoted as $x_i \in \mathcal{X} \in \mathbb{R}^{\lambda}$.

**Example:** Figure 1-(a) illustrates a small HIN, in which $\mathcal{N}$ = {*Institution, Author, Paper, Venue* } and $\mathcal{R}$ = {*Work, Hire, Write, Written, Cite, Cited, Publish, Published* }.

**Definition 2.** (**Meta-path**): Given an HIN $G$, a meta-path $\Omega$ is defined as a sequence in the form of $\omega_1 \xrightarrow{r_1} \omega_2 \xrightarrow{r_2} \dots \xrightarrow{r_n} \omega_{n+1}$, where $\omega_j \in \mathcal{N}$ denotes the type of a node, and $r_i \in \mathcal{R}$ denotes a relation type.

[1]Code and data are available at https://github.com/zhiqiangzhongddu/RL-HGNN

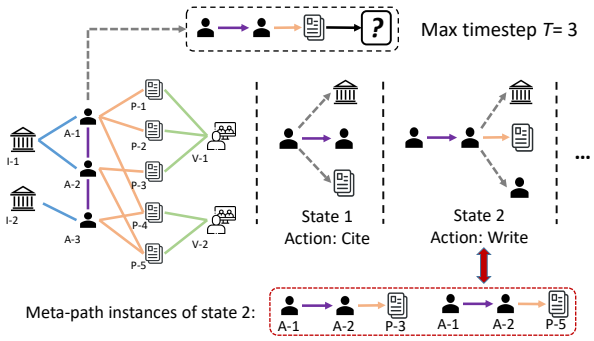


**Figure 2: An illustration of RL-HGNN designs meta-path for node A-1 as a Markov Decision Process (MDP). State 1 is the attribute of node A-1 and RL agent takes action to choose the relation *Cite* to extend the meta-path. Then, state 1 is updated to state 2, which is the average value of attributes of nodes A-1 and A-2. RL agent further takes action upon state 2 to extend the meta-path. Generated meta-path instances according to state 2 and action 2 are also shown in the figure.**

**Example:** Figure 1-(b) gives some meta-path examples, e.g., *Institution* $\xrightarrow{Hire}$ *Author*, *Venue* $\xrightarrow{Published}$ *Paper* $\xrightarrow{Written}$ *Author*, and *Author* $\xrightarrow{Cite}$ *Author* $\xrightarrow{Write}$ *Paper* $\xrightarrow{Written}$ *Author*.

**Definition 3.** (**Meta-path Instance**): Given a meta-path $\Omega$ of an HIN $G$, a meta-path instance $p$ of $\Omega$ is defined as a node sequence in $G$ following the schema defined by $\Omega$. $P = \{p_1, p_2, \ldots, p_{n+1}\}$ represents a generated meta-path instance of $\Omega$, where $p_i \in \mathcal{V}$ and $\phi(p_i) = \omega_i$ for $1 \le i \le n+1$, and $\varphi(\langle p_i, p_{i+1}\rangle) = r_i$ for $1 \le i \le n$.

**Example:** Following the meta-path $\Omega$ : *Venue* $\xrightarrow{Published}$ *Paper* $\xrightarrow{Written}$ *Author* in Figure 1-(b), we obtain several meta-path instances from $G$ in Figure 1-(a), e.g., *V-2* $\xrightarrow{Published}$ *P-5* $\xrightarrow{Written}$ *A-3*.

**Heterogeneous Graph Representation Learning with Reinforcement Learning**. Given an HIN $G = (\mathcal{V}, \mathcal{E}, \mathcal{N}, \mathcal{R})$, the problem of heterogeneous graph representation learning (HGRL) aims at learning the $d$-dimensional node representations $H[i] \in \mathbb{R}^d$ for all $v_i \in \mathcal{V}$ that are able to capture rich structure and semantic information provided in $G$. Due to the complexity and heterogeneity of $G$, existing approaches manually design meta-paths to represent the structure and semantic in $G$ and learn node representation based on these meta-paths. Our goal is to avoid the manual meta-path definition by adopting Reinforcement Learning (RL) algorithms to adaptively design appropriate meta-path for each node to capture comprehensive structure and semantic information in $G$ and learn valid node representation with the automatically designed meta-paths to unleash the power of HGRL.

## 2.2 Markov Decision Process

Markov Decision Process (MDP) is a mathematical idealised form to describe sequential decision process that satisfies the *Markov Property* [30]. An MDP can be formally represented with the quartuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{P}$ is a decision policy to identify the probability distribution of the next action according to current observed state, which is defined as a map function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to (0, 1)$. More specifically, the policy encodes the current state and action to output a probability distribution $\mathcal{P}(a_t|s_t)$, where $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$. $R$ is a reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, evaluating the result of taking action $a_t$ on the observed state $s_t$.

**Modelling HGRL with MDP**. As illustrated in Figure 2, the meta-path design process of HGRL can be naturally modelled as an MDP. In order to design a meta-path with maximum of 3 timesteps for node A-1, take the first timestep as an example, state $s_1$ can be the attribute of A-1 and action set includes the connected relations starting from A-1, i.e., {*Work*, *Cite*, *Write*}. Decision policy chooses one relation from the action set to extend the meta-path as $a_1 = argmax_a(\mathcal{P}(a \mid s_1))$. Then, the selected meta-path can be fed into HGRL to learn node representation $H^1$[A-1] and apply it to the downstream task to obtain a reward score ($R(s_1, a_1)$) which can be used to update $\mathcal{P}$. We refer to Section 3 for more details about modelling HGRL with MDP.

## 2.3 Solving MDP with Reinforcement Learning

Deep Reinforcement Learning (RL) is a family of algorithms that optimise the MDP with deep neural networks. At each timestep $t$, the RL agent takes action $a_t \in \mathcal{A}$ based on the current state $s_t \in \mathcal{S}$, and observes the next state $s_{t+1}$ as well as a reward $r_t = R(s_t, a_t)$. Looking at the definition of MDP, the agent acts as the decision policy $\mathcal{P}$. We aim to search for the optimal decisions so as to maximise the expected discounted cumulative reward, i.e., we would like to learn an agent network $\pi : \mathcal{S} \to \mathcal{A}$ to maximise $\mathrm{E}_\pi[\sum_{t'=t}^{\mathrm{T}} \gamma^{t'} \mathrm{r}_{t'}]$, where $T$ is the max timesteps and $\gamma \in (0, 1)$ is a discount factor to balance short-term and long-term gains, lower values place more emphasis on immediate rewards [1].

Existing RL algorithms are mainly classified into two series: model-based and model-free algorithms. Compared with model-based RL algorithms, model-free RL algorithms have better flexibility, as there is always the risk of suffering from model understanding errors, which in turn affects the learned policy [1]. We adopt a classic model-free RL algorithm, i.e., Deep Q-learning (DQN) [21, 22]. The basic idea of DQN is to estimate the action-value function by using the Bellman equation [22] as an interactive update:

$$Q(s_t, a_t; \theta) = \mathbb{E}_{s_{t+1}}[R(s_t, a_t) + \gamma \max_{a_{t+1}}(Q(s_{t+1}, a_{t+1}; \theta))], \quad (1)$$

where $s_{t+1}$ is the next state, $a_{t+1}$ is the next action and $\theta$ stands for the trainable parameters in the neural network that is used to estimate the decision policy. The proposition of Equation 1 is based on the following intuition: if the optimal value $Q(s_{t+1}, a_{t+1})$ of the state $s_{t+1}$ at the next timestep was known for all possible actions $a_{t+1}$, then the optimal policy is to select the action $a_{i+1}$ maximising the expected value $R(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1})$. A value iteration algorithm will coverage to the optimal action-value function, i.e., $Q \to \widetilde{Q}$ as $t \to \infty$.

DQN introduces two techniques to stabilise the training: (i) a *memory buffer* $\mathcal{D}$ which stores the agent's experience at each timestep in a replay memory that is accessed to perform the weight updates; (ii) a *separate target Q-network* $\hat{Q}$ for generating the targets in the Q-learning and is periodically updated. The ability to

use each stored experience for many updates allows DQN to learn more efficiently from its experiences and reduces the variance of the updates. Moreover, after every number of c updates, we clone the network $Q$ to obtain a separate target network $\hat{Q}$ and use $\hat{Q}$ for generating the Q-learning targets for the following c updates to $Q$. In such a way, the fluctuation becomes less severe, resulting in more stable training.

**Solving HGRL with RL**. As discussed in the Section 2.2, HGRL can be treated as an MDP, and combining with the content of this section, we can naturally utilise RL to optimise the HGRL through solving the meta-path design process of HGRL. More details about optimising HGRL with RL can be found in the next section.

## 3 THE RL-HGNN FRAMEWORK

Figure 3-(a) illustrates an overview of our framework RL-HGNN. There are two components in RL-HGNN, an RL agent module and an HGNN module. The RL agent aims at learning the relations between meta-path states and the future step in a meta-path. The HGNN module performs information aggregation on the generated meta-path instances to learn node representations.

Through combining these two modules, we unleash the power of HGRL by adopting the RL agent to adaptively design an optimal meta-path for each node instead of using the manual defined meta-paths. First, in RL-HGNN the RL agent summarises the attributes of nodes that involved in the meta-path as meta-path's state and maps the state into action (the next relation to extend the meta-path) as shown in Figure 2. Next, generated meta-path instances according to the meta-path can be utilised by the HGNN module to perform information aggregation to update the node representations. Last, the HGNN module applies the learned node representations for the downstream task and gets reward scores for updating the RL agent.

In what follows, we elaborate on details of our RL-HGNN framework. We first describe how we can design meta-path with the RL agent and how to train the RL agent. Then, we show how to perform information aggregation with the designed meta-paths. Last, we further propose an extension model of RL-HGNN, RL-HGNN++, which enables the meta-path design to explore both structural semantics of HIN and HGNN's status. Besides, it can also significantly accelerate the representation learning process.

### 3.1 Meta-path Design with RL

The key components of a MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R)$ include a set of states, a set of actions, a decision policy and a reward function as described in Section 2.2. As illustrated in Figure 2, a meta-path design process with max $T$ timesteps can be modelled as a $T$-round decision-making process which can be naturally treated as an MDP. We summarise the attribute of the starting node as state and choose one relation as the first step to extend the meta-path. Following the extended meta-path, we further generate corresponding meta-path instances for HGRL to learn node representations which can be applied to a downstream prediction task to obtain reward scores. In detail, we formalise the meta-path design process as follows.

- **State ($\mathcal{S}$)**: The state is used to assist the decision policy to choose a relation to extend the meta-path. It is crucial to capture all useful information in an existing meta-path. Therefore, we let the state to memorise all nodes involved in the meta-path. We take a meta-path $\Omega$ starting from node $v_i$ as an example, the state $s_t$ of $\Omega$ at timestep $t$ is formally defined as:

$$s_t = \frac{\frac{1}{|D(i)|}\sum_{j \in D(i)} x_j + s_{t-1}}{2}, \tag{2}$$

where $D(i)$ represents the arrived nodes at timestep $t$ with action $a_t$, $\frac{1}{|D(i)|}\sum_{j \in D(i)} x_j$ represents the average value of arrived nodes' attributes at timestep $t$. $s_{t-1}$ stands for the previous state at timestep $t-1$. Note that $s_0 = x_i$.

- **Action ($\mathcal{A}$)**: The action space $\mathcal{A}_t$ for a state $s_t$ is the set of available relation types $\{r_0, r_1, \dots\}$ in the HIN together with a special action *STOP*. Beginning with the starting node $v_i$, the decision policy iteratively predicts the most promising relation to extend the current meta-path to get a better reward score. It may either reach the maximal timestep $T$, or find a node representation that any extra relation in the meta-path will reduce the performance of node presentations on the downstream task, then the decision policy chooses the action *STOP* to finish the path design process.
- **Decision Policy ($\mathcal{P}$)**: The decision policy of an MDP aims at mapping a state in $\mathcal{S}$ into an action in $\mathcal{A}$. In the process of meta-path design, the state space and action space are continuous space and discrete space, respectively, i.e., $\mathcal{S} \sqsubset \mathbb{R}^{\lambda}$, $\mathcal{A} \sqsubset \mathbb{N}$. Thus we use a deep neural network to approximate the action-value function: $\mathcal{P}(a_t|s_t;\theta) : \mathcal{S} \times \mathcal{A} \rightarrow (0,1)$. Besides, due to any arbitrary action $a \in \mathcal{A}$ is always a positive integer, we use the Q-network of DQN [22] as the decision policy network, which was formally defined in Equation 1. We utilise a MLP [14] as the deep neural network of the Q-network, which is defined as:

$$\begin{aligned} z_1 &= W_1^T s_t + c_1, \\ z_2 &= W_2^T z_1 + c_2, \\ &\dots \\ \hat{P} &= Softmax(\phi_m(W_m^T z_{m-1} + c_m)), \end{aligned} \tag{3}$$

where $W_m$ and $c_m$ denote the weight matrix and bias vector for the $m$-th layer's perceptron, respectively. The output $\hat{P} \in (0,1)$ stands for the possibilities of choosing different relations $a_t \in \mathcal{A}_t$ to extend the meta-path. Note that, it is possible to adopt other RL algorithms as the policy network if they are able to approximate continuous state space and discrete space. Here, we utilise a basic RL algorithm to illustrate the main idea of our framework and demonstrate its effectiveness.

- **Reward function ($R$)**: Reward is a key factor in guiding the RL agent, we aim to design a reward function that can encourage the RL agent to achieve better and stable performance. Thus, we define the reward $R$ as the performance improvement on the specific task comparing with the history performances. More precisely, the reward function is defined as:

$$R(s_t, a_t) = \mathcal{M}(s_t, a_t) - \frac{\sum_{j=t-b}^{t-1} \mathcal{M}(s_j, a_j)}{b}, \tag{4}$$

where $\frac{\sum_{i=t-b}^{t-1} \mathcal{M}(s_j, a_j)}{b}$ is the baseline performance value at timestep $t$, which contains the history performances of last $b$ steps. $\mathcal{M}(s_t, a_t)$ is the performance of the learned node representation $H^t[i]$ on

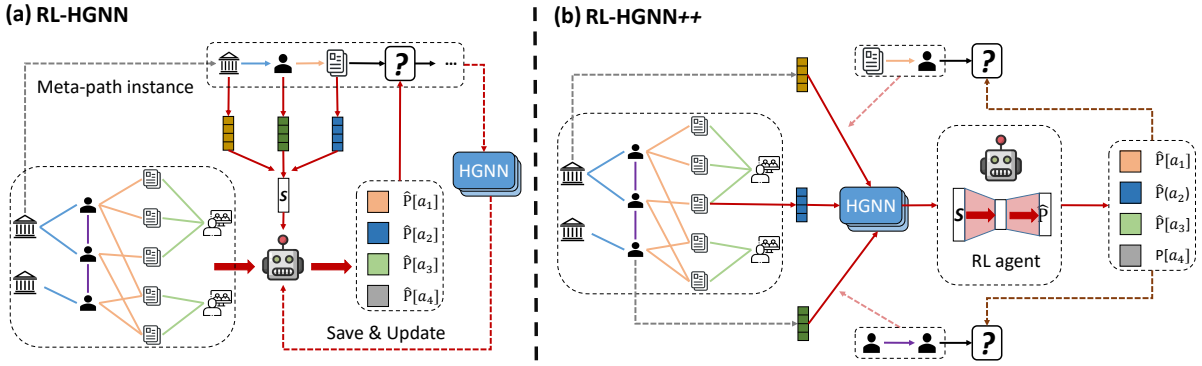


**Figure 3: An illustration of RL-HGNN and RL-HGNN++. (a) Overview of the RL-HGNN framework; (b) an illustrative example of how RL-HGNN++ defines State and makes Action.**

the downstream task. In this work, we adopt the node classification as the target task, and use its accuracy on the validation set as the evaluation performance.

**Optimisation.** Based on the definitions above, the proposed meta-path design process of node $v_i$ at timestep $t$ consists of three phases: (i) obtaining the state $s_t$ at timestep $t$; (ii) predicting an action (relation) $a_t = argmax_a(Q(s_t, a; \theta))$ to extend the meta-path according to the current state $s_t$; (iii) updating state $s_t$ to $s_{t+1}$. Moreover, we train the policy network Q-function by optimising Equation 1 with the reward function as defined in Equation 4 and the loss function defined in [22]:

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{\mathcal{T}\sim U(D)}[(R(s_t, a_t) + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}; \theta^-) - Q(s_t, a_t; \theta))^2], \quad (5)$$

where $\mathcal{T} = (s_t, a_t, s_{t+1}, R_t)$ is randomly sampled replay memory from the *memory buffer* $\mathcal{D}$, $\theta^-$ is the parameters of the *separate target Q-network* $\hat{Q}$, $\max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}; \theta^-)$ stands for the optimal target value and $Q(s_t, a_t; \theta)$ is the prediction value of the training Q-network. We optimise the Q-network' parameters by minimising the loss function via back-propagation and gradient descent.

## 3.2 Information Aggregation with Meta-path

Due to the heterogeneity of nodes in HIN, different types of nodes have different attribute spaces. Before the information aggregation, we first design a node type-specific transformation to project the attributes of different types of nodes into one same attribute space. The projections process can be formally presented as follows:

$$H^0[i] = M_{\omega_i} \cdot x_i, \quad (6)$$

where $x_i$ and $H^0[i]$ are the original and projected attribute of node $v_i$, and $M_{\omega_i}$ is the transformation matrix for $\omega_i$ type nodes.

After that, we discuss how to perform the information aggregation following these meta-path instances that were generated based on the meta-paths, to learn valid node representations for the downstream task. Here, we take the node $v_i$ as an example, and the designed meta-path for $v_i$ is assumed to be $\Omega$.

We first generate meta-path instances $P = \{p_1, p_2, \ldots, p_{n+1}\}$ following the meta-path $\Omega$. As the example shown in Figure 4, there could be many meta-path instances that follow the meta-path $\Omega$,

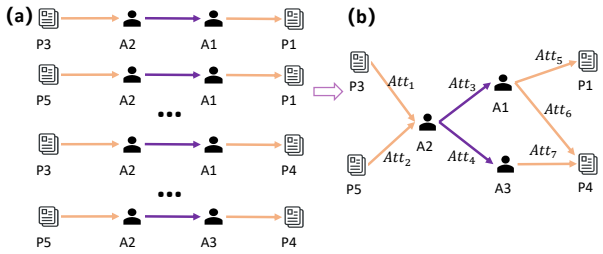


**Figure 4: Comparison between information aggregation by meta-path instances and our redundancy-free computation (A: Author, P: Paper). (a) The sequential aggregation path instances generated from the meta-path $\Omega$ : $Paper \xrightarrow{Written} Author \xrightarrow{Cite} Author \xrightarrow{Write} Paper$; (b) optimised aggregation paths by our redundancy-free computation method.**

e.g, a meta-path $Paper \xrightarrow{Written} Author \xrightarrow{Cite} Author \xrightarrow{Write} Paper$. An intuitive approach to perform the information aggregation is to adopt the sequential aggregator [42]. Examples include N-ary Tree-LSTM [31] and the LSTM variant of GraphSAGE [13]. But we argue these aggregation paths contain computation redundancy and it could be further optimised. For example, $P_3 \rightarrow A_2$ and $P_5 \rightarrow A_2$ are repeatedly calculated in the first aggregation step, hence we summarise these two into one aggregation process $\{P_3, P_5\} \rightarrow A_2$ to reduce additional computations, as *redundancy-free* information aggregation. Besides, we add attention scores $Att(s, j)$ for each aggregation path to distinguish different path instances [34]. We performed empirical study in Section 4.7 to compare the number of aggregations with and without this redundancy-free optimisation. Suppose that $H^t[i]$ is the learned representation of node $v_i$ and $\hat{H}[j]$ is the representation of node $v_j$ that was involved in $P$ at timestep $t$, we first formally describe the update procedure of $\hat{H}[j]$ as the following:

$$\hat{H}^l[j] = \underset{s\in N(j)}{Aggregate}(Att(s, j) \cdot \hat{H}^{l-1}[j]), \quad (7)$$

where $N(j)$ includes the connected nodes of $v_j$ in $P$. $l \in \{1, 2, \ldots, t\}$ is the id of aggregator to perform informatin aggregation, and $\hat{H}^0[j] = H^0[j]$. Similarly, $H^t[i]$ also can be learned with Equation 7.

$t$ aggregators stacked together forms our HGNN, which can update the node representations at timestep $t$ with $t$ times of aggregations.

The two basic operators $Att(\cdot)$ and $Aggregate(\cdot)$ are defined as:

$$Att(s,j) = \underset{s \in N(j)}{Softmax}(Leakrelu(W\hat{H}^{l-1}[s] \parallel W\hat{H}^{l-1}[j])),$$

$$Aggregate(\cdot) = Relu(Mean(\cdot)),$$

where $W$ contains the trainable parameters. The use of $Att(\cdot)$ can help us to distinguish different path instances in the optimal aggregation mechanism. For instance, two meta-path instances: $P_3 \xrightarrow[Att_1]{Written} A_2 \xrightarrow[Att_3]{Cite} A_1$ and $P_3 \xrightarrow[Att_1]{Written} A_2 \xrightarrow[Att_4]{Cite} A_3$, even their first half are integrated together but the attention scores help to distinguish them.

**Parameter Sharing.** We reduce the number of training parameters via the parameter sharing mechanism. In particular, we first initialise the max timestep $T$ of HGNN at the beginning and in each timestep, we repeatedly stack the aggregators by the initialisation order for the action $a_t$ to perform information aggregation. In this way, if the number of trainable parameters of each aggregator is $u$, we only need to train $T \times u$ parameters, instead of $\frac{T(T+1)}{2} \times u$ parameters.

**Training of HGNN.** Finally, the updated node representation $H^t[i]$ can be applied to downstream tasks and design different loss functions to update the HGNN. For semi-supervised node classification, with the help of a small fraction of labelled nodes, we can optimise the HGNN' parameters by minimising the Cross-Entropy via back-propagation and gradient descent. The Cross-Entropy loss can be formally defined as:

$$\mathcal{L} = -\sum_{v \in \mathcal{V}_L} \sum_{c=0}^{C-1} y_v[c] \cdot log(H^t[v][c]), \tag{8}$$

where $\mathcal{V}_L$ stands for the set of nodes with labels, $C$ is the number of classes, $y_v$ is the one-hot label vector of node $v$ and $H^t[v]$ is the learned node representation vector of node $v$. Meanwhile, the learned node representation can be evaluated by task specific evaluation matrix $\mathcal{M}_{eval}$, and fed into Equation 4 ($\mathcal{M}(s_t, a_t) = \mathcal{M}_{eval}(H^t)$) for obtaining a reward score. We summarise our RL-HGNN framework in Algorithm 1.

**Algorithm 1:** RL-HGNN

**Input:** HIN $G = (\mathcal{V}, \mathcal{E}, \mathcal{N}, \mathcal{R})$, Max timesteps $T$, Number of RL agent training rounds $K$, Number of HGNN training rounds $B$,
**Output:** Node representations $H^T$.

```
Initialise memory buffer D and Q-function
for k ← {1, 2, ..., K} do
    Initialise information aggregators (HGNN)
    for t ← {0, 1, 2, ..., T} do
        Sample a batch of nodes V_j from V
        for v_i ∈ V_j do
            Obtain state s_t[i] via Equation 2;
            Get the action a_t[i] = argmax_a Q(s_t[i], a; θ);
            Extend the meta-path Ω_t[i] with a_t[i] and
              generate meta-path instances P_t[i].
        end
        for β ← {1, 2, ..., B} do
            Perform information aggregation on P_t via
              Equation 7, and optimise HGNN' parameters
              via Loss-function (Equation 8)
        end
        Perform information aggregation on P_t with trained
          HGNN via Equation 7;
        Obtain the learned node representations H^t;
        Obtain R_t on validation dataset via Equation 4;
        Store the quartuple T = (s_t, a_t, s_{t+1}, R_t) into D;
        Optimise Q-function using the data in D via
          Loss-function-Q (Equation 5).
    end
end
```

## 3.3 RL-HGNN++

As seen from previous sections, RL-HGNN mainly adopts the RL agent to adaptively design meta-paths for HGRL. However, reviewing the overall workflow of RL-HGNN as illustrated in Figure 5-(a) and Algorithm 1, we identify three limitations: (i) it ignores the impact of the randomness of HGNN on HGRL; (ii) it neglects the structures of HIN while designing meta-paths; (iii) HGNN requires a large number of rounds to train their parameters and obtain node representation which restricts the efficiency of RL-HGNN. To be more specific, RL-HGNN designs meta-paths by only considering the node attribute information from the HIN, i.e., state $s_t$ summarising the attributes of nodes involved in the meta-path. However, the quality of learned node representation is closely related to the HGNN as well but so far RL-HGNN doesn't take this into account. For instance, Scardapane and Wang [24] discussed the randomness of deep neural network models and demonstrated various random factors that may influence the model's performance. In addition, similar to the training process of other deep neural networks, the encoder of RL-HGNN needs a number of rounds to train their parameters to learn valid node representations in terms of a set of meta-path instances. In this way, if HGNN needs $B$ rounds to train their parameters then we need $T \times B$ rounds to complete a meta-path design process with the max $T$ timesteps. Meanwhile, theRL agent also needs a large number of quartuple samples ($\mathcal{T} = (s_t, a_t, s_{t+1}, R_t)$) to train the policy network which can result in a combinatorial scale explosion.

**New State ($\mathcal{S}$).** To address the identified limitations, we propose an extension of RL-HGNN which can utilise the comprehensive information of both the HIN and the HGNN module. Our solution is to define a novel state to capture such information. In particular, we utilise the latest node representation vector $H^{t-1}[i]$ of meta-path's starting node $v_i$ as the state. Following this way, the structure and node attribute information are summarised in $H^{t-1}[i]$ with the help of HGNN, meanwhile the status of HGNN is also reflected in $H^{t-1}[i]$. The new state ($\mathcal{S}$) can be formally described as:

$$s_t = Normalise(H^{t-1}[i]), \tag{9}$$

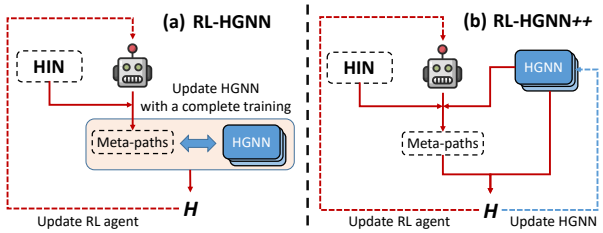


**Figure 5: Overview of the workflow of RL-HGNN and RL-HGNN++. (a) The workflow of RL-HGNN: the RL agent designs meta-paths for nodes of HIN according to the node attribute information, then trains an HGNN model with the designed meta-paths and learns node representations with the trained HGNN. Reward scores can be obtained by applying learned representations on the downstream task and then can be used to update the RL agent. (b) The workflow of RL-HGNN++: the RL agent designs meta-paths for nodes of HIN according to the HIN's structures and semantic information as well as the status of HGNN, then learns node representations with the HGNN. The learned representations can be used for optimising the HGNN and updating the RL agent after being applied to the downstream task.**

where *Normalise* is a normaliser trained on the first $B$ generated states, to convert the states into the same distribution. Note that *normaliser* is a common trick used in deep RL algorithms for stabilising the training when the state space is very sparse.

**New Training Process**. Following the new state definition, the training process can be further optimised as well. The new process is presented in Figure 5-(b). Compared with the process of RL-HGNN (shown in Figure 5-(a)), the major difference relies in that we can directly utilise the HGNN to learn a valid node representation instead of updating the HGNN with a complete training process before learning the node presentations over meta-path. The learned node representation can be further used to optimise HGNN's parameters and update the RL agent. It is because the meta-paths are designed for the current status of HGNN. With this novel training process, we only need $T$ rounds to complete a meta-path design procedure, instead of $T \times K$ rounds. We perform an empirically analysis to compare the time consuming of RL-HGNN and RL-HGNN++ in Section 4.7. We summarise RL-HGNN++ in Algorithm 2.

## 4 EXPERIMENTS

We evaluate RL-HGNN and RL-HGNN++ through experiments to address the following research questions:

- RQ1 How do RL-HGNN and RL-HGNN++ perform compared with state-of-the-art models for HGRL?
- RQ2 What are the main differences between RL-HGNN explored meta-paths and human-designed meta-paths?
- RQ3 What is the impact of the major components of RL-HGNN?

### 4.1 Datasets

We adopt two widely used HIN datasets from different domains to evaluate the performance of our models. The detailed description of the datasets is presented in Table 1.

**Algorithm 2:** RL-HGNN++

**Input:** HIN $G = (\mathcal{V}, \mathcal{E}, \mathcal{N}, \mathcal{R})$, Max timesteps $T$, Number of RL agent training rounds $K$,
**Output:** Node representations $H^T$.

Initialise information aggregators (HGNN), memory buffer $\mathcal{D}$ and Q-function,
**for** $k \leftarrow \{0, 1, K-1\}$ **do**
  Timestep $t$ = mode$(k, T) + 1$,
  Sample a batch of nodes $\mathcal{V}_j$ from $\mathcal{V}$,
  **for** $v_i \in \mathcal{V}_j$ **do**
    Obtain state $s_t[i]$ via Equation 9;
    Get the action $a_t[i] = argmax_a Q(s_t[i], a; \theta)$;
    Extend the meta-path $\Omega_t[i]$ with $a_t[i]$ and generate meta-path instances $P_t[i]$.
  **end**
  Perform information aggregation on $P_t$ via Equation 7;
  Obtain the learned node representations $H^t$;
  Obtain $R_t$ on validation dataset via Equation 4;
  Store the quartuple $\mathcal{T} = (s_t, a_t, s_{t+1}, R_t)$ into $\mathcal{D}$;
  Optimise HGNN's parameters via Loss-function (Equation 8);
  Optimise Q-function using the data in $\mathcal{D}$ via Loss-function-Q (Equation 5).
**end**

**Table 1: Statistics of the datasets.**

| Dataset | Node | Relation |
|---|---|---|
| IMDb | # Movie (M): 4,278<br># Director (D): 2,081<br># Actor (A): 5,257 | # M-D, # D-M: 4,278<br># M-A, # A-M: 12,828 |
| DBLP | # Author (A): 4,057<br># Paper (P): 14,328<br># Term (T): 7,723<br># Venue (V): 20 | # A-P # P-A: 19,645<br># P-T # T-P: 85,810<br># P-V # V-P: 14,328 |

- **IMDB**[2] is an online dataset about movies and television program, including information such as cast, production crew and plot summaries. We extract a subset of IMDB which contains 4,278 movies, 2,081 directors and 5,257 actors. The movies are labelled as one of the three classes, i.e., *Action*, *Comedy* and *Drama*, according to their genre information. The feature of each movie corresponds to elements of a bag-of-words (i.e., their plot keywords, 1,232 in total).
- **DBLP**[3] is an online computer science bibliography. We extract a subset of DBLP which contains 4,057 movies, 1,4328 papers, 7,723 terms and 20 venues. The authors are labelled as one of the following four research areas: *Database*, *Data mining*, *Machine learning* and *Information retrial*. Each author can be described by a bag-of-words (i.e., their paper keywords, 334 in total).

[2] https://www.imdb.com/
[3] https://dblp.uni-trier.de/

## 4.2 Competing Methods

To validate the effectiveness of the proposed models, we compare their performance against different kinds of state-of-the-art models, including 5 homogeneous graph representation learning models and 7 heterogeneous graph representation learning models:

- **LINE** [32] is a homogeneous model that exploits the first-order and second-order proximity between nodes. We apply it to the datasets by ignoring HIN heterogeneity and node attributes.
- **DeepWalk** [23] is a random walk based graph representation learning method for homogeneous graphs, we apply it to the datasets by ignoring HIN heterogeneity and node attributes.
- **ESim** [27] is a HGRL method that learns different semantics from multiple meta-paths. It requires a pre-defined weight for each meta-path, we assign all meta-paths with a same weight.
- **metapath2vec** [6] is a HGRL approach that performs random walk on HIN with the guidance of pre-defiend meta-paths and utilises skip-gram to generate node representations.
- **HERec** [28] is a HGRL method which designs a type constraint strategy to filter the node sequence and utilises skip-gram to generate node representations.
- **MLP** [14] is a class of feed-forward neural networks that learns information from node attributes without structural information.
- **GCN** [19] is a homogeneous GNN that extends the convolution operation to graphs.
- **GAT** [34] is a homogeneous GNN that performs convolution on graphs with an attention mechanism.
- **RGCN** [25] is a heterogeneous GNN model which keeps a different weight for each relation to perform convolution on graphs.
- **HAN** [38] is a heterogeneous GNN model which learns meta-path guided node representations from different meta-path based homogeneous graphs and integrates them with using attention.
- **MAGNN** [10] is a heterogeneous GNN model which defines metapath instance encoders to extract the structural and semantics in meta-paths to improve generated representations.
- **HGT** [16] is a heterogeneous GNN model which uses each meta-relation to parameterise transformer-like self-attention architecture to capture common and specific patterns of relationships.

## 4.3 Implementation Details

For the DQN of the proposed RL-HGNN and RL-HGNN++, we use the implementation in [20] with a few modifications to fit it with our models. We develop a 5-layers of MLP with (32, 64, 128, 64, 32) as the hidden units for $Q$ function. The memory size is $50 \times b$, where $b$ is the number of validation nodes in the dataset. For the HGNN model of the proposed RL-HGNN and RL-HGNN++, we randomly initialise parameters and optimise the model with Adam optimiser [18]. We set the learning rate to 0.005, the regularisation parameter to 0.0001, the representation vector dimension is 128, the dimension of the attention vector $Att(\cdot)$ to 16 the training batch size to 256 and the number of attention head is 8, with a dropout ratio to 0.5. The max timesteps ($T$) for IMDB and DBLP are 2 and 4, respectively. Code and data are available at: https://github.com/zhiqiangzhongddu/RL-HGNN.

For random walk based competing methods, including DeepWalk, Esim, metapath2vec and HERec, we set the window size to 5, walk length to 100, walks per node to 40 and the number of negative samples to 5. For meta-path guided competing methods, including Esim, metapath2vec, HERec, HAN and MAGNN, we give them the pre-defined meta-paths as in [38]. For IMDB, there are two meta-paths: $Movie \xrightarrow{M-A} Actor \xrightarrow{A-M} Movie$ and $Movie \xrightarrow{M-D} Director \xrightarrow{D-M} Movie$. For DBLP, there are three meta-paths: $Author \xrightarrow{A-P} Paper \xrightarrow{P-A} Author$, $Author \xrightarrow{A-P} Paper \xrightarrow{P-T} Term \xrightarrow{T-P} Paper \xrightarrow{P-A} Author$, and $Author \xrightarrow{A-P} Paper \xrightarrow{P-V} Venue \xrightarrow{V-P} Paper \xrightarrow{P-A} Author$. For GNN based models, we test them with the same parameters as RL-HGNN on the same data split. We re-implement all competing models in PyG[4] following the authors' official code.

## 4.4 Experimental Settings

We evaluate our models under semi-supervised and supervised settings. For the semi-supervised setting, we adopt the same settings as in [10] to use 400 instances for training and another 400 instances for validation, and the rest nodes for testing. The generated node representations are further fed into a *support vector machine* (SVM) classifier to get the prediction results. For the supervised setting, we use the same number of nodes for training and validation, respectively, and use the rest of the nodes for testing. We employ an end-to-end mechanism, namely the model is trained and optimised on the training and validation sets, respectively, and generates the predictions on the testing set directly.

## 4.5 Experimental Results (RQ1)

**Semi-supervised node classification**. We present the results of semi-supervised node classification in Table 2, where competing methods that don't support semi-supervised settings keep their unsupervised settings. We report the average *Micro-F1* and *Macro-F1* of 10 runs with different seeds of each model. As shown in the table, RL-HGNN++ performs consistently better than other competing methods across different proportions and datasets. On IMDB, the performance gain obtained by RL-HGNN++ over the best competing method (MAGNN) is around (3.7% - 5.08%). And the GNNs, including all GNN models designed for homogeneous and heterogeneous graphs, perform better than shallow heterogeneous network embedding models, which demonstrates that the usage of heterogeneous node attributes benefits the performance significantly. On DBLP, the performances of all models are overall better compared with results on IMDB. It is interesting to observe that different from the results on IMDB, the shallow heterogeneous network methods obtain better performances than homogeneous GNNs. That said, the heterogeneous relations on DBLP is useful for the node classification task. RL-HGNN++ outperforms the strongest competing method up to 2.4%, it demonstrates the generality of RL-HGNN++.

In addition, we also have an interesting observation that RL-HGNN outperforms the competing methods in most cases on the IMDB dataset, but performs less outstanding than the strongest baseline in most cases on the DBLP dataset. We argue the reason is that the labels in the DBLP datasets are more accurate than those in IMDB and DBLP contains much richer heterogeneous relations. Since RL-HGNN can only generate one meta-path for each node,

[4] https://pytorch-geometric.readthedocs.io/en/latest/

**Table 2: Experiment results (%) on the IMDb and DBLP datasets for the node classification task with unsupervised and semi-supervised settings.**

| Dataset | Metrics | Train % | Unsupervised | | | | | Semi-supervised | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | LINE | DeepWalk | ESim | metapath2vec | HERec | GCN | GAT | HAN | MAGNN | RL-HGNN | RL-HGNN++ |
| IMDb | Micro-F1 | 20% | 45.21 | 49.94 | 49.32 | 47.22 | 46.23 | 52.80 | 53.64 | 56.32 | 59.60 | 62.14 | **62.53** |
| | | 40% | 46.92 | 51.77 | 51.21 | 48.17 | 47.89 | 53.76 | 55.56 | 57.32 | 60.50 | 62.18 | **63.10** |
| | | 60% | 48.35 | 52.79 | 52.53 | 49.87 | 48.19 | 54.23 | 56.47 | 58.42 | 60.88 | 62.69 | **63.13** |
| | | 80% | 48.98 | 52.72 | 52.54 | 50.50 | 49.11 | 54.63 | 57.40 | 59.24 | 61.53 | 62.86 | **64.07** |
| | Macro-F1 | 20% | 44.04 | 49.00 | 48.37 | 46.05 | 45.61 | 52.73 | 53.64 | 56.19 | 59.35 | 61.87 | **62.37** |
| | | 40% | 45.45 | 50.63 | 50.09 | 47.57 | 46.80 | 53.67 | 55.50 | 56.15 | 60.27 | 62.17 | **62.95** |
| | | 60% | 47.09 | 51.65 | 51.45 | 48.17 | 46.84 | 54.24 | 56.46 | 57.29 | 60.66 | 62.50 | **62.98** |
| | | 80% | 47.49 | 51.49 | 51.37 | 49.99 | 47.73 | 54.77 | 57.43 | 58.51 | 61.44 | 62.65 | **63.98** |
| DBLP | Micro-F1 | 20% | 87.68 | 87.21 | 91.21 | 89.02 | 91.49 | 88.51 | 91.61 | 92.33 | 93.61 | 91.73 | **94.34** |
| | | 40% | 89.25 | 88.51 | 92.05 | 90.36 | 92.05 | 89.22 | 91.77 | 92.57 | 93.68 | 92.01 | **95.78** |
| | | 60% | 89.34 | 89.09 | 92.28 | 90.94 | 92.66 | 89.57 | 91.97 | 92.72 | 93.99 | 92.67 | **95.92** |
| | | 80% | 89.96 | 89.37 | 92.68 | 91.31 | 92.78 | 90.33 | 92.24 | 93.23 | 94.47 | 93.91 | **96.63** |
| | Macro-F1 | 20% | 87.16 | 86.70 | 90.68 | 88.47 | 90.82 | 88.00 | 91.05 | 91.69 | 93.13 | 91.63 | **94.94** |
| | | 40% | 88.85 | 88.07 | 91.61 | 89.91 | 91.44 | 89.00 | 91.24 | 91.96 | 93.23 | 92.06 | **95.42** |
| | | 60% | 88.93 | 88.69 | 91.84 | 90.50 | 92.08 | 89.43 | 91.42 | 92.14 | 93.57 | 92.76 | **95.57** |
| | | 80% | 89.51 | 88.93 | 92.27 | 90.86 | 92.25 | 89.98 | 91.73 | 92.50 | 94.10 | 94.89 | **96.30** |

which is not sufficient compared with baselines with many human-defined meta-paths. In contrast, RL-HGNN++ can arrive at the best performances at both datasets, thanks to its design of defining meta-paths with comprehensive information.

**Supervised node classification**. We present the results of supervised node classification in Table 3. We report the average *Micro-F1* and *Macro-F1* of 10 runs of each model with different seeds. For the table, we can see that RL-HGNN++ consistently outperforms all competing models on the IMDB and DBLP datasets, with up to 5.6% in terms of *Micro-F1*. Note that all GNN models that contain the topology information obtain better performances than MLP, which only works on node attributes. Heterogeneous GNNs outperform homogeneous GNNs and our RL enhance model, RL-HGNN++, achieves the best performance. This demonstrates the usefulness of heterogeneous relations and the advantages of designing different appropriate meta-paths for each node according to their features.

## 4.6 Meta-paths Analysis (RQ2)

In addition to the quantitative evaluation of our proposed framework, we also visualise the process of how RL agent designs meta-path for each node to answer RQ2. As shown in Figure 6, it summarises the actions made by the RL agent on IMDB with different max timesteps. They follow the semi-supervised training settings and note that since RL-HGNN++ may generate different meta-paths in different rounds, here we only present the actions of the round that arrives the best validation performance. It is interesting to see that the RL agent picks the relation *M-A* at the first timestep more often than the other one *M-D*, which means Movie's characteristic is more related to its actors than its director. Besides, when timestep $t$=2, the RL agent picks more and more *STOP* actions (indicated as 'S' in Figure 6). This shows that two short meta-paths, i.e., *Movie* $\xrightarrow{M-A}$ *Actor* and *Movie* $\xrightarrow{M-D}$ *Director* are already sufficient for the major part of nodes to learn their valid representations. Moreover, there are 6.7%-9,4% nodes that don't need any meta-paths to learn their representations, which means their attributes can provide enough information. This has also been reflected in the results of MLP in Section 4.5, which doesn't utilise any structural information but only node attributes.

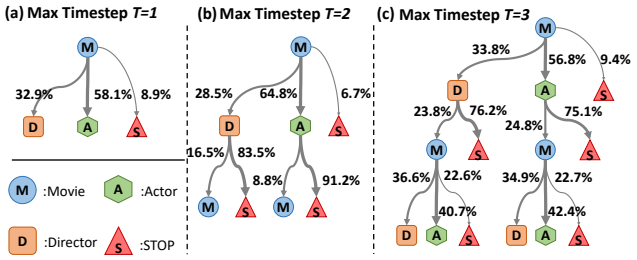


**Figure 6: Elaboration of the actions of the RL agent on IMDB: (a), (b) and (c) correspond to RL-HGNN++ with max timestep T=1, T=2 and T=3, respectively. The thickness of the line represents the ratio of the corresponding action.**

Moreover, we present the most relevant meta-paths in Table 4[5] generated by RL-HGNN++, with different max timesteps $T$. Similar to Figure 6, here we only present the meta-paths of the round that arrives the best validation performance. From the table, we can find that the RL agent is able to design various meta-paths for different nodes. The manually defined meta-path i.e., *Movie* $\xrightarrow{M-A}$ *Actor* $\xrightarrow{A-M}$ *Movie*, is discovered by our model as well. However, these meta-paths defined by experts are not the most useful ones to learn node representations for Movie nodes. On the contrary, two short meta-paths *Movie* $\xrightarrow{M-A}$ *Actor* and *Movie* $\xrightarrow{M-D}$ *Director* are the most useful. It indicates that participants (director and actors) of the movie largely determine its type (since the target task is to predict the class of movies).

Moreover, considering the performances of RL-HGNN++ with different max timesteps ($T$) that presented in Figure 7, there is a

[5]We have omitted the relation types for all the meta-paths in Table 4.

**Table 3: Experiment results (%) on the IMDb and DBLP datasets for the node classification task with supervised settings.**

| Dataset | Metrics | Train % | Supervised | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | MLP | GCN | GAT | RGCN | HAN | HGT | RL-HGNN | RL-HGNN++ |
| IMDb | Micro-F1 | 5% | 47.60 | 52.18 | 53.32 | 54.09 | 55.74 | 55.80 | 54.90 | **57.09** |
| | | 10% | 49.75 | 55.81 | 56.53 | 58.22 | 59.21 | 60.06 | 60.97 | **62.27** |
| | | 15% | 54.62 | 58.55 | 59.13 | 60.13 | 60.25 | 60.67 | 61.12 | **63.77** |
| | | 20% | 55.22 | 59.13 | 60.22 | 60.15 | 60.70 | 61.71 | 62.33 | **64.65** |
| | Macro-F1 | 5% | 47.67 | 51.89 | 53.17 | 54.20 | 55.53 | 55.36 | 54.31 | **56.47** |
| | | 10% | 49.07 | 54.96 | 56.32 | 58.17 | 58.57 | 60.13 | 61.02 | **62.04** |
| | | 15% | 54.86 | 57.23 | 58.82 | 60.02 | 60.03 | 60.42 | 60.93 | **63.81** |
| | | 20% | 55.02 | 58.56 | 59.90 | 60.14 | 60.44 | 61.43 | 62.07 | **63.89** |
| DBLP | Micro-F1 | 5% | 69.25 | 78.94 | 78.01 | 81.98 | 90.82 | 90.08 | 89.11 | **92.52** |
| | | 10% | 75.75 | 82.53 | 82.69 | 82.73 | 92.17 | 92.20 | 90.43 | **93.87** |
| | | 15% | 76.87 | 83.50 | 84.61 | 83.29 | 92.62 | 92.82 | 91.20 | **94.72** |
| | | 20% | 77.95 | 83.94 | 85.07 | 84.67 | 93.16 | 93.21 | 91.98 | **95.13** |
| | Macro-F1 | 5% | 68.50 | 78.23 | 76.74 | 81.11 | 90.24 | 89.93 | 88.63 | **92.20** |
| | | 10% | 74.85 | 81.46 | 81.59 | 82.65 | 91.59 | 91.21 | 89.70 | **93.34** |
| | | 15% | 76.17 | 82.77 | 82.00 | 83.79 | 91.90 | 92.90 | 90.84 | **94.14** |
| | | 20% | 76.86 | 83.05 | 84.34 | 84.11 | 92.13 | 92.97 | 91.31 | **94.33** |

significant performance improvement comparing $T \geq 2$ with $T = 1$, we argue that meta-paths with richer semantics can bring benefits of learning better node representations. At last, RL-HGNN++ has designed some meta-paths that were ignored by experts, e.g., $Movie \xrightarrow{M-A} Actor \xrightarrow{A-M} Movie \xrightarrow{M-A} Actor$, and they are more effective than $Movie \xrightarrow{M-A} Actor \xrightarrow{A-M} Movie$.

**Table 4: Meta-paths designed by RL-HGNN++ in IMDB.**

| Max Timestep ($T$) | Meta-paths Designed by RL-HGNN++ | Percentage (%) |
|---|---|---|
| $T = 1$ | *Movie* → *Actor* | 58.1 |
| | *Movie* → *Director* | 32.9 |
| $T = 2$ | *Movie* → *Actor* | 59.1 |
| | *Movie*→ *Director* | 23.8 |
| | *Movie*→*Actor*→*Movie* | 5.7 |
| | *Movie*→*Director*→*Movie* | 4.7 |
| $T = 3$ | *Movie*→*Actor* | 42.7 |
| | *Movie*→*Director* | 25.8 |
| | *Movie*→*Actor*→*Movie*→*Actor* | 5.9 |
| | *Movie*→*Actor*→*Movie*→*Director* | 4.9 |
| | *Movie*→*Actor*→*Movie* | 3.2 |
| Manually Defined Meta-paths | *Movie*→*Actor*→*Movie* | |
| | *Movie*→*Director*→*Movie* | |

## 4.7 Model Analysis (RQ3)

We conduct experiments to study the influence of hyper-parameters and evaluate the effectiveness of some components in RL-HGNN.

**Influence of Max Timestep $T$.** Following the same experimental procedure in semi-supervised node classification, we study the influence of one key hyper-parameter, i.e., max timestep $T$. Specifically, we investigate the following two aspects: (i) its influence on the performance of the classification task which is measured by *Micro F1*; (ii) its influence on the number of information aggregations. The results are presented in Figure 7. We can see that the cases when $T \geq 2$ lead to significant improvements over $T = 1$ in terms of *Micro F1*, and as $T$ increases the number of information aggregations

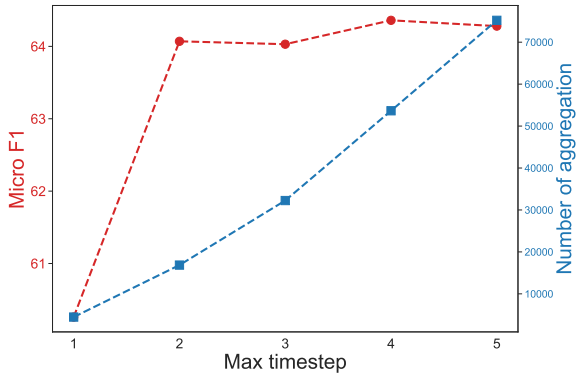


**Figure 7: The performance (in terms of Micro F1) and the number of aggregations of RL-HGNN++ on IMDB, with different max timesteps.**

also increases. Considering more information aggregations leads to higher computational cost, we choose $T = 2$ for all the experiments in Section 4.5, even though $T > 2$ can lead to better performance.

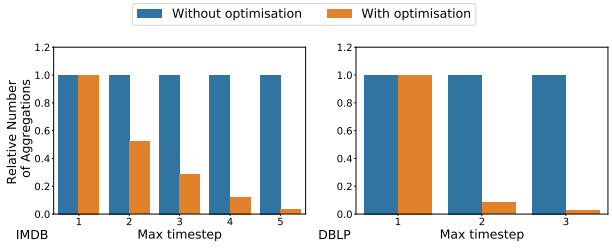


**Figure 8: Comparing the number of aggregations with and without the redundancy-free optimisation with different max timesteps (lower is better).**

**Influence of Redundancy-free Optimisation.** Here, we investigate the influence of the redundancy-free optimisation by comparing the number of information aggregations with and without the

optimisation. Figure 8 shows the comparison results. We can find that this optimisation strategy significantly reduces the number of information aggregations. Besides, as the max timestep $T$ increases, the reduced ratio becomes higher. For instance, the optimisation reduces around 50% aggregations on IMDB with $T = 2$, and the reduced ratio is more than 80% when $T = 4$.

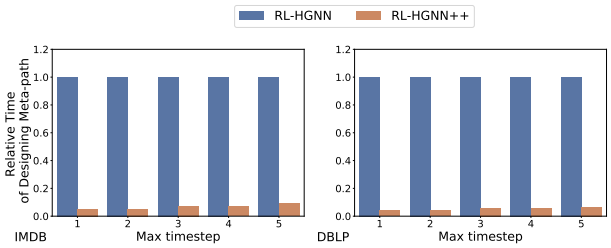


**Figure 9: Comparing the running time of meta-path design of RL-HGNN and RL-HGNN++ with different max timesteps (lower is better).**

**Comparison of Time Consuming for Meta-path Design.** As we introduced in Section 3.3, RL-HGNN++ significantly accelerates the meta-path design process by optimising the training process. Here, we conduct comparisons of time-consuming of RL-HGNN and RL-HGNN++ for meta-path design. We calculate the running time of RL-HGNN++ relative to RL-HGNN, and the results are shown in Figure 9. From the figure, we can find that RL-HGNN++ consume less than 5% of RL-HGNN's running time both datasets.

## 5 RELATED WORK

**Graph Neural Networks**. Existing Graph Neural Networks (GNNs) generalise the convolutional operations of deep neural networks to deal with arbitrary graph-structured data [40]. Generally, a GNN model can be regarded as using the input graph structure to generate nodes' computation graph for message passing [11], during which the local neighbourhood information is aggregated to get a more contextural representation of a node in the network. Following this idea, spectral-based GNN models were first proposed to perform graph convolution in the Fourier domain of a graph [3, 5]. Defferrard et al. proposed ChebNet which utilises Chebyshev polynomials to filter node features in the graph [5]. Later on, another series of spatial approaches drawn remarkable attention due to their effectiveness and efficiency [13, 19, 34, 41]. The spatial-based GNN models define convolution directly in the graph domain by aggregating node features from the neighbours for each node and further combining with its own features. For information aggregation, GCN [19] adopts mean pooling and GraphSAGE [13] concatenates on nodes' features with supplementary max/LSTM [15] pooled neighbour information. Inspired by the Transformer [33], GAT [34] aggregates neighbourhood information based on trainable attention weights. Several other spatial-based GNN variants have been proposed to further extend this idea. DGI [35] alternatively drives local network embeddings to capture global structural information by maximising local mutual information. Cluster-GCN [4] samples a subgraph using a graph clustering algorithm and performs graph convolutions to nodes within the sampled subgraph to improve the ability of handling large graphs.

However, the complex graph-structured data in real world are commonly associated with multiple types of objects and relations. All of the GNN models mentioned above assume homogeneous graphs, thus it is difficult to directly apply them to HINs.

**Heterogeneous Graph Representation Learning**. HGRL aims to project nodes in a HIN into a low-dimensional vector space while preserving the heterogeneous node features and relations. A recent survey presents a comprehensive overview on HGRL [7], and it focuses on shallow heterogeneous network embedding methods [6, 8, 9], and heterogeneous GNN-based approaches that are empowered by rather complex deep encoders [10, 16, 25, 38, 42]. The "shallow" methods are characterised as an embedding lookup table, meaning that they directly encode each node in the network as a vector, and this embedding table is the parameter to optimise. For example, Metapath2vec [6] designs a meta-path based random walk and utilises skip-gram [2] to perform heterogeneous graph embedding. HIN2Vec [9] carries out multiple prediction training tasks which learn the latent vectors of nodes and meta-paths simultaneously. However, Metapath2vec and HIN2Vec only utilise a few meta-paths, both of them cannot utilise the node attributes, and do not support the end-to-end training strategy. On the other hand, inspired by the recent outstanding performance of GNN models, some studies have attempted to extend GNNs for HINs. The R-GCNs model keeps a distinct linear projection weight for each relation type [25]. HetGNN [42] adopts different Recurrent Neural Networks for different node types to integrate multi-modal features. HAN [38] extends GAT by maintaining different weights for different meta-path-defined edges, and MAGNN [10] defines meta-path instance encoders, which are used to extract the structural and semantic information ingrained in the meta-path instances.

However, all of these models require manual effort and domain expertise to define meta-paths to capture the semantics underlying the given heterogeneous structure. This leads to two main limitations: (i) the design of meta-paths requires rich domain knowledge that is extremely difficult to obtain in complex and semantic rich HINs; (ii) existing meta-path guided methods simply assume that different nodes/relations of the same type share the same meta-path, which ignores the differences among individual nodes. There is one recently proposed model HGT [16] attempting to avoid the the dependency on human-defined meta-paths by design transferable relation scores, but its exploration range is limited by the number of layers of HGNN model and it introduces a large number of parameters to optimise. Therefore, we intend to design a new HGRL framework, which adaptively designs different meta-paths for each node in the network with the help of reinforcement learning.

## 6 CONCLUSIONS AND FUTURE WORK

We have studied in this paper the *heterogeneous graph representation learning* (HGRL) problem and identified the limitation of existing HGRL methods mainly due to their dependency on manually defined meta-paths. In order to fully unleash the power of HGRL, we presented a novel framework **R**einforcement **L**earning enhanced **H**eterogeneous **G**raph **N**eural **N**etwork (RL-HGNN) and proposed one extension model RL-HGNN++. Comparing with existing HGRL models, the biggest improvement of our framework is avoiding

the manual efforts in defining meta-paths of HGRL. The experimental results have demonstrated that our framework generally out-performs the competing approaches and discovers many useful meta-paths which have been ignored by human design.

In the future, we aim to adopt a multi-agent RL algorithm to further enable the RL-HGNN to design multiple meta-paths for each node, instead of only focusing one optimal meta-path. In addition, it is also interesting to study how to extend our framework to other tasks on HINs, such as the online recommendation system, knowledge graph completion, and etc.

## ACKNOWLEDGMENT

This work is supported by the Luxembourg National Research Fund through grant PRIDE15/10621687/SPsquared. This work is also supported by Ministry of Science and Technology (MOST) of Taiwan under grants 109-2636-E-006-017 (MOST Young Scholar Fellowship) and 108-2218-E-006-036, and also by Academia Sinica under grant AS-TP-107-M05. The authors thank Zhe Xie for her help in designing the code.

## REFERENCES

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine* (2017).

[2] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Proceedings of the 2013 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2787–2795.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *Proceedings of the 2014 International Conference on Learning Representations (ICLR)*.

[4] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *Proceedings of the 2019 ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 257–266.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering.. In *Proceedings of the 2016 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 3837–3845.

[6] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *Proceedings of the 2017 ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 135–144.

[7] Yuxiao Dong, Ziniu Hu, Kuansan Wang, Yizhou Sun, and Jie Tang. 2020. Heterogeneous Network Representation Learning. In *Proceedings of the 2020 International Joint Conferences on Artifical Intelligence (IJCAI)*. IJCAI, 4861–4867.

[8] Yujie Fan, Shifu Hou, Yiming Zhang, Yanfang Ye, and Melih Abdulhayoglu. 2018. Gotcha - Sly Malware!: Scorpion A Metagraph2vec Based Malware Detection System. In *Proceedings of the 2018 ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 253–262.

[9] Tao-Yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. HIN2Vec: Explore Meta-paths in Heterogeneous Information Networks for Representation Learning. In *Proceedings of the 2017 ACM International Conference on Information and Knowledge Management (CIKM)*. ACM, 1797–1806.

[10] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph embedding. In *Proceedings of the 2020 International Conference on World Wide Web (WWW)*. ACM, 2331–2341.

[11] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. [n.d.]. Neural Message Passing for Quantum Chemistry.

[12] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 2016 ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 855–864.

[13] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 2017 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 1025–1035.

[14] Simon Haykin. 1999. Neural Networks: A Comprehensive Foundation. *Knowledge Engineering Review* (1999).

[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* (1997).

[16] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *Proceedings of the 2020 International Conference on World Wide Web (WWW)*. ACM, 2704–2710.

[17] Jiarui Jin, Jiarui Qin, Yuchen Fang, Kounianhua Du, Weinan Zhang, Yong Yu, Zheng Zhang, and Alexander J. Smola. 2020. An Efficient Neighborhood-based Interaction Model for Recommendation on Heterogeneous Graph. In *Proceedings of the 2020 ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 75–84.

[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 2015 International Conference on Learning Representations (ICLR)*.

[19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 2017 International Conference on Learning Representations (ICLR)*.

[20] Kwei-Herng Lai, Daochen Zha, Kaixiong Zhou, and Xia Hu. 2020. Policy-GNN: Aggregation Optimization for Graph Neural Networks. In *Proceedings of the 2020 ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 461–471.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR abs/1312.5602* (2013).

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (2015), 529–533.

[23] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the 2014 ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 701–710.

[24] Simone Scardapane and Dianhui Wang. 2017. Randomness in neural networks: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* (2017).

[25] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2019. Modeling Relational Data with Graph Convolutional Networks. In *European Semantic Web Conference (ESWC)*. ACM, 593–607.

[26] Dominic Seyler, Praveen Chandar, and Matthew Davis. 2018. An Information Retrieval Framework for Contextual Suggestion Based on Heterogeneous Information Network Embeddings. In *Proceedings of the 2018 International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 953–956.

[27] Jingbo Shang, Meng Qu, Jialu Liu, Lance M. Kaplan, Jiawei Han, and Jian Peng. 2016. Meta-Path Guided Embedding for Similarity Search in Large-Scale Heterogeneous Information Networks. *CoRR abs/1610.09769* (2016).

[28] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and Philip S. Yu. 2019. Heterogeneous Information Network Embedding for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[29] Yizhou Sun and Jiawei Han. 2012. Mining heterogeneous information networks: a structural analysis approach. *ACM SIGKDD Explorations Newsletter* (2012).

[30] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. MIT Press.

[31] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 2015 Annual Meeting of the Association for Computational Linguistics (ACL)*. ACL, 1556–1566.

[32] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proceedings of the 2015 International Conference on World Wide Web (WWW)*. ACM, 1067–1077.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Proceedings of the 2017 Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 5998–6008.

[34] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the 2018 International Conference on Learning Representations (ICLR)*.

[35] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *Proceedings of the 2019 International Conference on Learning Representations (ICLR)*.

[36] Xuan-Son Vu, Addi Ait-Mlouk, Erik Elmroth, and Lili Jiang. 2019. Graph-based Interactive Data Federation System for Heterogeneous Data Retrieval and Analytics. In *Proceedings of the 2019 International Conference on World Wide Web (WWW)*. ACM, 3595–3599.

[37] Guojia Wan, Bo Du, Shirui Pan, and Gholamreza Haffari. 2020. Reinforcement Learning Based Meta-Path Discovery in Large-Scale Heterogeneous Information Networks. In *Proceedings of the 2020 AAAI Conference on Artificial Intelligence (AAAI)*. AAAI.

[38] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *Proceedings of the 2019 International Conference on World Wide Web (WWW)*. ACM, 2022–2032.

[39] Xin Wang, Ying Wang, and Yunzhi Ling. 2020. Attention-Guide Walk Model in Heterogeneous Information Network for Multi-Style Recommendation Explanation. In *Proceedings of the 2020 AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 6275–6282.

[40] Zonghan Wu, Shirui Pan, Fengwen Chen, and Guodong Long. 2020. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[41] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *Proceedings of the 2019 International Conference on Machine Learning (ICML)*. JMLR.

[42] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous Graph Neural Network. In *Proceedings of the 2019 ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 793–803.