Data Mining and Knowledge Discovery (2022) 36:2299–2333
https://doi.org/10.1007/s10618-022-00862-z

Personalised meta-path generation for heterogeneous graph neural networks

Zhiqiang Zhong[1] · Cheng-Te Li[2] · Jun Pang[1,3]

Received: 9 December 2021 / Accepted: 18 August 2022 / Published online: 2 October 2022

Abstract
Recently, increasing attention has been paid to heterogeneous graph representation learning (HGRL), which aims to embed rich structural and semantic information in heterogeneous information networks (HINs) into low-dimensional node representations. To date, most HGRL models rely on hand-crafted meta-paths. However, the dependency on manually-defined meta-paths requires domain knowledge, which is difficult to obtain for complex HINs. More importantly, the pre-defined or generated meta-paths of all existing HGRL methods attached to each node type or node pair cannot be personalised to each individual node. To fully unleash the power of HGRL, we present a novel framework, Personalised Meta-path based Heterogeneous Graph Neural Networks (PM-HGNN), to jointly generate meta-paths that are personalised for each individual node in a HIN and learn node representations for the target downstream task like node classification. Precisely, PM-HGNN treats the meta-path generation as a Markov Decision Process and utilises a policy network to adaptively generate a meta-path for each individual node and simultaneously learn effective node representations. The policy network is trained with deep reinforcement learning by exploiting the performance improvement on a downstream task. We further propose an extension,

Responsible editor: Albrecht Zimmermann.

✉ Cheng-Te Li
chengte@mail.ncku.edu.tw

✉ Jun Pang
jun.pang@uni.lu

Zhiqiang Zhong
zhiqiang.zhong@uni.lu

1 Faculty of Science, Technology and Medicine, University of Luxembourg, Esch-sur-Alzette, Luxembourg

2 Institute of Data Science and the Department of Statistics, National Cheng Kung University, Tainan, Taiwan

3 Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Esch-sur-Alzette, Luxembourg

PM-HGNN++, to better encode relational structure and accelerate the training during the meta-path generation. Experimental results reveal that both PM-HGNN and PM-HGNN++ can significantly and consistently outperform 16 competing baselines and state-of-the-art methods in various settings of node classification. Qualitative analysis also shows that PM-HGNN++ can identify meaningful meta-paths overlooked by human knowledge.

Keywords Meta-path generation · Heterogeneous graph neural networks

1 Introduction

The complex interactions in real-world data, such as social networks, biological networks and knowledge graphs, can be modelled as Heterogeneous Information Networks (HINs) (Sun and Han 2012), which are commonly associated with multiple types of nodes and relations. Take the academia HIN depicted in Fig. 1a as an example; it involves 4 types of nodes, including *Papers* (P), *Authors* (A), *Institutions* (I) and *publication venues* (V), and 8 types of relations. Due to the capability of HINs to depict the complex inter-dependency between nodes, they have attracted increasing attention in the research community and have been applied in the fields of relational learning (Van Otterlo 2005), recommender systems (Zheng et al. 2021), information retrieval (Wan et al. 2020), etc. However, the complex semantics and non-Euclidean nature of HINs render them challenging to modulate by conventional machine learning algorithms designed for tabular or sequential data.

Over the past decade, a significant line of research on HINs is *Heterogeneous Graph Representation Learning* (HGRL). The goal of HGRL is to learn latent node representations, which encode complex graph structure and multi-typed nodes, for downstream tasks, including link prediction (Wan et al. 2020), node classification (Wang et al. 2019) and node clustering (Fu et al. 2020). As discussed in a recent survey (Dong et al. 2020), one of the paradigms on HGRL is to manually define and use meta-paths to model HIN's rich semantics and to leverage random walks to transform the graph structure into a set of sequences (Dong et al. 2017; Fu et al. 2017; Shi et al. 2019), which can be further exploited by shallow embedding learning algorithms (Mikolov et al. 2013; Le and Mikolov 2014). A meta-path scheme is defined as a sequence of relations over HIN's network schema. For instance, an illustrative meta-path in the academia HIN in Fig. 1a is $A \xrightarrow{Cite} A \xrightarrow{Write} P$. Some follow-up shallow HGRL models try to avoid the requirement of manually defined meta-paths by developing jump and stay random walk strategies (Hussein et al. 2018), performing random walk with the guide of node contexts (Jiang et al. 2020), or switching to utilising network schema (Zhao et al. 2020). Nevertheless, these "shallow" methods neither support end-to-end training to learn more effective representations for a specific task nor fully utilise node attributes due to the limitation of the embedding algorithms.

Recently, in view of the impressive success of Graph Neural Networks (GNNs) (Kipf and Welling 2017; Velickovic et al. 2018), the second paradigm of HGRL attempts to devise Heterogeneous Graph Neural Networks (HGNNs) for HGRL (Schlichtkrull

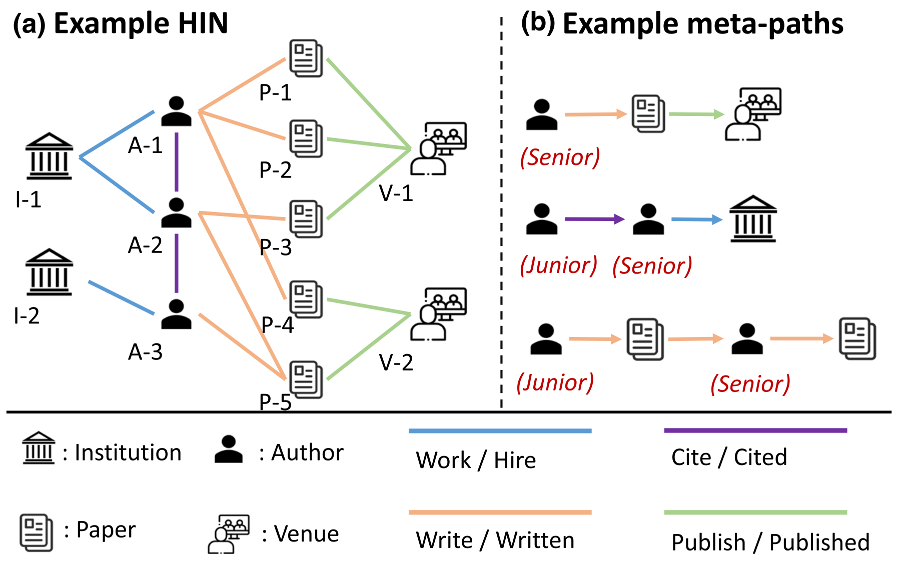

Fig. 1 An example HIN and few meth-paths: **a** an academia HIN; **b** meta-paths designed for *senior* vs. *junior* authors

et al. 2019; Zhang et al. 2019; Fu et al. 2020), which extend various graph convolutions on HIN. Compared with "shallow" HGRL methods, HGNNs support an end-to-end training mechanism that can learn node representations with some labelled nodes' assistance and are also empowered by more complex encoders instead of using the shallow embedding learning methods. HGNNs can model both structure and node attributes in HINs with the guidance of meta-paths. However, they still rely on the hand-crafted meta-paths to explicitly model the semantics of HINs, and obtaining meaningful and useful meta-paths for nodes in HINs to guide HGNNs is still highly non-trivial.

More precisely, existing meta-path guided HGRL methods simply assume that nodes with the same type share the same meta-path. Take the academia HIN as an example (Fig. 1a) and assume we plan to learn node representations to determine the research area of *Author*s. A meta-path Ω_1: $A \xrightarrow{Write} P \xrightarrow{Published} V$ 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: Ω_2: $A \xrightarrow{Cite} A \xrightarrow{Write} P$, because Ω_1 retains little information in the case of junior PhD candidates. Hence, we argue that we should generate a *personalised* meta-path for each "individual node" according to its attributes and neighbouring relational structures instead of giving each "node type" several pre-defined meta-paths in general.

Motivated by the outstanding success of Reinforcement Learning (RL) in strategy selection problems (Mnih et al. 2015), previous methods attempt to apply RL techniques to find paths between given node pairs which model the similarity between two nodes (Meng et al. 2015; Yang et al. 2018; Wan et al. 2020). The found paths are then fed into the encoder to learn representations for pairwise tasks like link prediction. Nevertheless, challenges still remain in designing personalised meta-paths of individual nodes for node-wise tasks.

Key challenges in personalised meta-paths generation. First, the definition of meta-paths requires rich domain knowledge that is extremely difficult to obtain in complex and semantic rich HINs (Dong et al. 2020). Specifically, given a HIN G with a node type set N, a relation type set R and a fixed meta-path length T. The possible meta-paths are contained in a set of size $(N \times R)^T$. Such a huge set can result in a combinatorial explosion when increasing the scales of N, R and T. Second, the representation capacities of manually-defined meta-paths are limited to a specific task on specific HIN since different G with the same N and R may have different node attributes and relation types distributions. It requires defining appropriate meta-paths for each task on each HIN, which is extremely difficult for practical applications.

In light of these challenges, we propose to investigate HGRL with the objectives of (i) learning to generate a personalised meta-path for each individual node in HINs automatically, (ii) learning node representations effectively and efficiently with personalised meta-paths, and (iii) retaining the end-to-end training strategy to achieve task-oriented optimisation. To achieve these objectives, we present a novel **P**ersonalised **M**eta-path based **H**eterogeneous **G**raph **N**eural **N**etwork (PM-HGNN), to unleash the power of HGRL.

Key Ideas of PM-HGNN. Generally, we aim to replace the human efforts on meta-path generation with an RL agent to address the limitations of the dependency on hand-crafted meta-paths of HGNNs. Compared with experts with domain knowledge, the RL agent can adaptively generate personalised meta-paths for each individual node in terms of a specific task/HIN through sequential exploration and exploitation. That said, the obtained meta-paths are no longer for specific types of nodes but are personalised for each individual node. Both graph structure and node attributes are considered in the meta-path generation process, and it is practicable for HINs with complex semantics.

As illustrated in Fig. 2, the meta-path generation process can be naturally considered as a Markov Decision Process (MDP), in which the next relation to extending the meta-path depends on the current state in a meta-path. Moreover, an HGNN model is proposed to learn node representations from the derived meta-paths that can be applied to a downstream task, such as node classification. 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 agent. The reward function is defined as the performance improvement over historical performance, which encourages the RL agent to achieve better and more stable performance. In addition, we find that there exists a large computational redundancy during the information aggregation on meta-paths, thus we develop an efficient strategy to achieve redundancy-free aggregation.

We showcase an instance of our framework, i.e. PM-HGNN, by implementing it with a classic RL algorithm, i.e., Deep Q-learning (Mnih et al. 2015). Besides, we further propose an extension of PM-HGNN++ to deal with the issues of high computational cost and ignoring relational information in PM-HGNN. Specifically, PM-HGNN generates a meta-path for each node according to node attributes, while PM-HGNN++

further enables the meta-path generation to explore the structural semantics of HIN. PM-HGNN++ is able to not only significantly accelerate the HGRL process but also improve the effectiveness of learned node representations with promising performance on downstream tasks.

Main contributions. We summarise our contributions below:

- We present a framework, PM-HGNN,[1] to learn node representations in a HIN without hand-crafted meta-paths. An essential novelty of PM-HGNN is that the generated meta-paths are personalised to every individual node rather than general to each node type.
- We propose an attention-based redundancy-free mechanism to reduce redundant computation during heterogeneous information aggregation on the derived meta-path instances.
- We further develop an extension of PM-HGNN, PM-HGNN++, which not only improves the meta-path generation by incorporating node attributes and relational structure but also accelerates the training process.
- Experiments conducted on node classification tasks with unsupervised and (semi-)supervised settings exhibit that our framework can significantly and consistently outperform 16 competing methods (up to 5.6% *Micro-F1* improvements). Advanced studies further reveal that PM-HGNN++ can identify meaningful meta-paths that human experts have ignored.

2 Related work

Relational learning In the past decades, research focused on using frameworks that could represent a variable number of entities and the relationships that hold amongst them. The interest in learning using this expressive representation formalism soon resulted in the emergence of a new subfield of machine learning that was described as relational learning (Van Otterlo 2005; Raedt 2008). For instance, TILDE (Blockeel and Raedt 1998) learns decision trees within inductive logic programming systems. Serafino et al. (2018) proposed an ensemble learning-based relational learning model for multi-type classification tasks in HINs. Petkovic et al. (2020) proposed a relational feature ranking method based on the gradient-boosted relational trees towards relational data. Lavrac et al. (2020) presented a unifying methodology combining propositionalisation and embedding techniques, which benefit from the advantages of both in solving complex relational learning tasks. Nevertheless, most of them are not in virtue of neural networks, which fall behind in automatically mining complex HINs.

Graph neural networks Existing GNNs generalise the convolutional operations of deep neural networks to deal with arbitrary graph-structured data. Generally, a GNN model can be regarded as using the input graph structure to generate the computation graph of nodes for message passing, the local neighbourhood information is aggregated to get more effective contextual node representations in the network (Bruna et al. 2014; Defferrard et al. 2016; Kipf and Welling 2017; Velickovic et al. 2018). However,

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

the complex graph-structured data in the 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 apply them to HINs directly.

Heterogeneous graph representation learning HGRL aims to project nodes in a HIN into a low-dimensional vector space while preserving the heterogeneous node attributes and edges. A recent survey presents a comprehensive overview of HGRL (Dong et al. 2020), covering shallow heterogeneous network embedding methods (Dong et al. 2017; Fu et al. 2017; Fan et al. 2018), and heterogeneous GNN-based approaches that are empowered by rather complex deep encoders (Meng et al. 2015; Schlichtkrull et al. 2019; Zhang et al. 2019; Wang et al. 2019; Fu et al. 2020; Hu et al. 2020; Zheng et al. 2021). 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 be optimised. However, they 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. R-GCNs (Schlichtkrull et al. 2019) keep a distinct linear projection weight for each relation type. HetGNN (Zhang et al. 2019) adopts different recurrent neural networks for different node types to incorporate multi-modal attributes. HAN (Wang et al. 2019) extends GAT (Velickovic et al. 2018) by maintaining weights for different meta-path-defined edges. MAGNN (Fu et al. 2020) 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 in order to capture the semantics underlying the given heterogeneous graph.

A recent model, HGT (Hu et al. 2020), attempts to avoid the dependency on handcrafted meta-paths by devising transferable relation scores, but the number of layers limits its exploration range, and it introduces a large number of additional parameters to optimise. GTN Yun et al. (2019) selects meta-paths from a group of adjacency matrices with learnable weights. The weights are shared among all nodes and are thus not flexible to generate node-specific meta-paths for each individual node. In addition, FSPG Meng et al. (2015), AutoPath Yang et al. (2018) and MPDRL Wan et al. (2020) attempt to employ RL technologies to discover paths between pairs of nodes and further learn node representations for predicting the possibility of the existing edges between node pairs. They assume the founded paths explicitly represent the similarity between two nodes. However, they can only identify meta-paths that describe two nodes' similarities instead of generating meta-path for individual nodes to learn their representations for node-wise tasks. Moreover, some work (Tanon et al. 2018; Ahmadi et al. 2020) concerns the discovery of frequent patterns in a HIN and the subsequent transformation of these patterns into rules, aka rule mining. But the found patterns are not designed for specific tasks or nodes. Consequently, we believe it is necessary and essential to developing a new HGRL framework that can support the adaptive generation of personalised meta-paths for each node in HIN for node-wise tasks.

Discussion Table 8 summarises the key advantages of PM-HGNN and compares it with a number of recent state-of-the-art methods. PM-HGNN is the first HGRL model that can adaptively generate personalised meta-paths for each individual node to support node-wise tasks and maintain the end-to-end training mechanism.

3 Preliminaries

3.1 Problem Statement

Definition 1 (*Heterogeneous information network*): A HIN is defined as a directed graph $G = (V, E, N, R)$, associated with a node type mapping function $\phi : V \rightarrow N$ and a relation type mapping function $\varphi : E \rightarrow R$, where N and R are the sets of node and edge types, respectively. Node v_i's attribute vector is denoted as $x_i \in \mathbb{R}^{\lambda}$ (with the dimensionality λ).

Definition 2 (*Meta-path*): Given a HIN G, a meta-path Ω with length T is defined as: $\omega_0 \xrightarrow{r_1} \omega_1 \xrightarrow{r_1} \ldots \xrightarrow{r_T} \omega_T$, where $\omega_j \in N$ denotes a certain node type, and $r_i \in R$ denotes a relation type.

Definition 3 (*Meta-path Instance*): Given a meta-path Ω, a meta-path instance p is defined as a node sequence following the schema defined by Ω.

Problem: Heterogeneous graph representation learning. We formulate heterogeneous graph representation learning (HGRL) as an information integration optimisation problem. For a given HIN G, $f : (V, E) \rightarrow \mathbb{R}^d$ be a function to integrate information from node attributes and network structure as node representations. Without manually specifying any meta-paths, we aim to jointly generate ρ (e.g., ρ=1) meta-paths $\{\Omega_j\}_{j=1}^{\rho}$ for each node $v_i \in V$ to guide f to encode rich structural and semantic information in HIN, and accordingly learn representations for all nodes in G.

3.2 Markov decision process

Markov Decision Process (MDP) is an idealised mathematical form to describe a sequential decision process that satisfies the *Markov Property* (Sutton and Barto 1998). An MDP can be formally represented with the quadruple $(S, A, \mathcal{P}, \mathcal{R})$, where S is a finite set of states, A is a finite set of actions, and $\mathcal{P} : S \times A \rightarrow (0, 1)$ is a decision policy function to identify the probability distribution of the next action according to the current state. Specifically, the policy encodes the state and the available action at step t to output a probability distribution $\mathcal{P}(a_t|s_t)$, where $s_t \in S$ and $a_t \in A$. $\mathcal{R}$ is a reward function $\mathcal{R} : S \times A \rightarrow \mathbb{R}$, evaluating the result of taking action a_t on the observed state s_t.

Modelling HGRL with MDP. As illustrated in Fig. 2, the meta-path generation process of HGRL can be naturally modelled as an MDP. To generate a meta-path with maximum of 3 steps for $v_{A\text{-}1}$, we take the first step as an example, the state s_1 is identifiable information of $v_{A\text{-}1}$ and the action set includes relations in the HIN, i.e., $\{Work, Cite, \ldots\}$. The decision maker selects one relation from the action set to extend the meta-path as $a_1 = argmax_{a \in A}(\mathcal{P}(a \mid s_1))$. Then, the selected meta-path is fed into HGNN to learn node representation and apply it to the downstream task to obtain a reward score $\mathcal{R}(s_1, a_1)$ that can be used to update $\mathcal{P}$. We refer to Sect. 4 for more details about modelling HGRL with MDP.

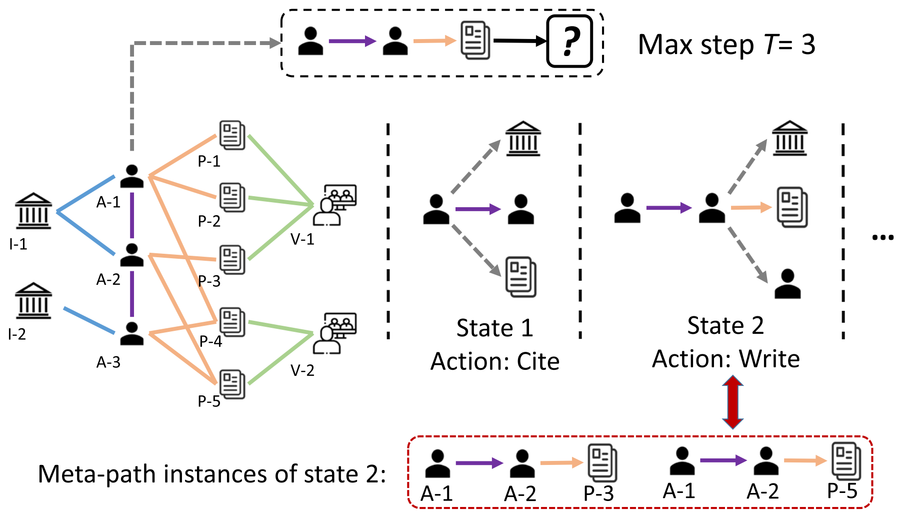

Fig. 2 Illustration of generating meta-paths as an MDP

3.3 Solving MDP with reinforcement learning

Deep Reinforcement Learning (RL) is a family of algorithms that optimise the MDP with deep neural networks. At each step t, the RL agent takes action $a_t \in A$ based on the current state $s_t \in S$, and observes the next state s_{t+1} as well as a reward $r_t = \mathcal{R}(s_t, a_t)$. Looking at the definition of MDP, the agent acts as the decision policy with $\mathcal{P}$. We aim to search for the optimal decisions that maximise the expected discounted cumulative reward, i.e., we aim to learn an agent network $\pi : S \to A$ to maximise $\mathbb{E}_\pi[\sum_{t'=t}^{T} \gamma^{t'} r_{t'}]$, where T is the maximum number of steps, and $\gamma \in [0, 1]$ is a discount factor to balance short-term and long-term gains, and smaller γ values place more emphasis on immediate rewards (Arulkumaran et al. 2017).

Existing RL algorithms are mainly classified into two series: model-based and model-free algorithms. Compared with model-based algorithms, model-free algorithms have better flexibility, as there is always the risk of suffering from model understanding errors, which in turn affects the learned policy (Arulkumaran et al. 2017). We adopt a classic model-free RL algorithm, i.e., Deep Q-learning (DQN) (Mnih et al. 2015). The basic idea of DQN is to estimate the action-value function by using the Bellman equation (Mnih et al. 2015) ($\mathcal{Q}^*$) as an interactive update based on the following intuition: if the optimal value $\mathcal{Q}^*(s_t, a_t)$ of the state s_t was known for all possible actions $a_t \in A$, then the optimal policy is to select the action a_i maximising the expected value $\mathcal{R}(s_t, a_t) + \gamma \mathcal{Q}^*(s_{t+1}, a_{t+1})$. And it is common to use a function approximator $\mathcal{Q}$ to estimate $\mathcal{Q}^*$ (Mnih et al. 2015):

$$\mathcal{Q}(s_t, a_t; \theta) = \mathbb{E}_{s_{t+1}}\left[\mathcal{R}(s_t, a_t) + \gamma \max_{a_{t+1}\in A} (\mathcal{Q}(s_{t+1}, a_{t+1}; \theta))\right], \tag{1}$$

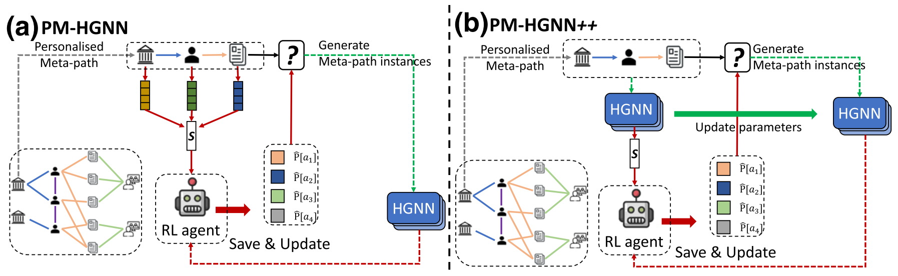

Fig. 3 Overview of PM-HGNN and PM-HGNN++

where θ stands for the trainable parameters in the neural network that is used to estimate the decision policy. A value iteration algorithm will approach the optimal action-value $\widetilde{\mathcal{Q}}$, i.e., $\mathcal{Q} \rightarrow \widetilde{\mathcal{Q}}$, as $t \rightarrow \infty$.

DQN exploits two techniques to stabilise the training: (1) the *memory buffer* $\mathcal{D}$ that stores the agent's experience in a replay memory, which can be accessed to perform the weight updating. (2) the *separate target Q-network* ($\hat{\mathcal{Q}}$) to generate the target for Q-learning, which is periodically updated.

4 The PM-HGNN Framework

We present the overview of the proposed PM-HGNN in Fig. 3a, which consists of two components: the RL agent module and an HGNN module. According to states, the RL agent aims at predicting actions for each individual node to arrive at better rewards. Next, we generate meta-path instances based on generated personalised meta-paths to support the information aggregation of HGNN to learn effective node representations. Finally, we apply generated representations on the downstream task for performance evaluation to obtain reward scores and save states, actions, and reward scores into the RL agent for the subsequent updating.

4.1 Personalised Meta-path Generation with RL

A personalised meta-path generation process with maximum T steps for each node can be modelled as a T-round decision-making process that can be naturally treated as an MDP. We elaborate on the alignment between each MDP component and the personalised meta-path generation process in the following.

State (S): The state is a vector used to assist the decision policy to select a relation type to extend the personalised meta-paths for each node. Hence, it is crucial to comprehensively encode existing parts of a meta-path into a state. We adopt a gating mechanism to adaptively update the state. Take v_i's meta-path Ω (starting from node type N_{v_i}) as an example, the state s_t of Ω at step t is formally defined as:

$$s_t = q \circ \left(\frac{1}{|D(v_i)|} \sum_{j \in D(v_i)} x_j \right) + (1 - q) \circ s_{t-1}, \tag{2}$$

where $\circ$ stands for the Hadamard product, $D(v_i)$ represents a set of past nodes at step t, and $\frac{1}{|D(v_i)|}\sum_{j\in D(v_i)} x_j$ represents the average vector of past nodes' attributes. s_{t-1} stands for the state at step $t-1$. q is the update gate that can determine whether to update a state with past nodes' attributes, and we estimate it by exploring the relationship between past nodes' attributes and states. It is formally defined as: $q = Sigmoid\left(f_\varphi\left((\frac{1}{|D(v_i)|}\sum_{j\in D(v_i)} x_j) \parallel s_{t-1}\right)\right)$. f_φ can be seen as a shared convolutional kernel (Velickovic et al. 2018), and $1-q$ is the reset gate.

Action (A): The action space is a set of relation types to be chosen by the policy network to extend the meta-path, and each relation type is labelled with a positive integer. Note that we add a special action *STOP* to allow each node to have flexible-length meta-paths. Beginning with the starting node v_i, the decision policy iteratively predicts which relation can lead to higher reward scores and accordingly use it to extend the current meta-path. The decision policy selects the action *STOP* to finish the path generation process if encountering a state that includes any extra relationship to the current meta-path hurts the performance on the downstream task.

Decision policy ($\mathcal{P}$): The decision policy aims at mapping a state in S into action in A. The state space and the action space are continuous and discrete, respectively. Thus we use a deep neural network to approximate the action-value function: $\mathcal{P}(a_t|s_t;\theta) : S \times A \to (0, 1)$. Besides, since any arbitrary action $a \in A$ is always a positive integer, we use DQN (Mnih et al. 2015) to optimise the meta-path generation problem. We utilise an MLP (Haykin 1999) as the decision policy network in the DQN, defined as: $z_1 = W_1^T s_t + b_1$, $z_2 = W_2^T z_1 + b_2$, ..., $\hat{P} = Softmax(\phi_m(W_m^T z_{m-1} + b_m))$, where W_m and b_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 selecting different relations $a_t \in A$ to extend the meta-path. Note that it is possible to adopt other RL algorithms to optimise the policy network. Here, we utilise a basic RL algorithm to illustrate our framework's main idea and demonstrate its effectiveness.

Reward function ($\mathcal{R}$): We devise a reward function to encourage the RL agent to achieve better and more stable performance on downstream tasks. We define the reward function $\mathcal{R}$ as the performance improvement on the specific downstream task comparing with the historical performances, given by:

$$\mathcal{R}(s_t, a_t) = \mathcal{M}(s_t, a_t) - \frac{\sum_{j=t-b}^{t-1} \mathcal{M}(s_j, a_j)}{b}, \tag{3}$$

where $\frac{\sum_{i=t-b}^{t-1} \mathcal{M}(s_j, a_j)}{b}$ is the baseline performance value at step t, which contains the historical performances of the last b steps. $\mathcal{M}(s_t, a_t)$ is the performance based on the learned node representation $H^t[v_i]$ on the downstream task (e.g., node classification). And use its accuracy on the validation set as the evaluation performance $\mathcal{M}$.

Optimisation The proposed meta-path generation at step t consists of three phases: (i) obtaining state s_t; (ii) predicting an action $a_t = \arg\max_{a\in A}(Q(s_t, a; \theta))$ 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 Eq. 1 with the reward function as defined in Eq. 3, and the loss function is given by:

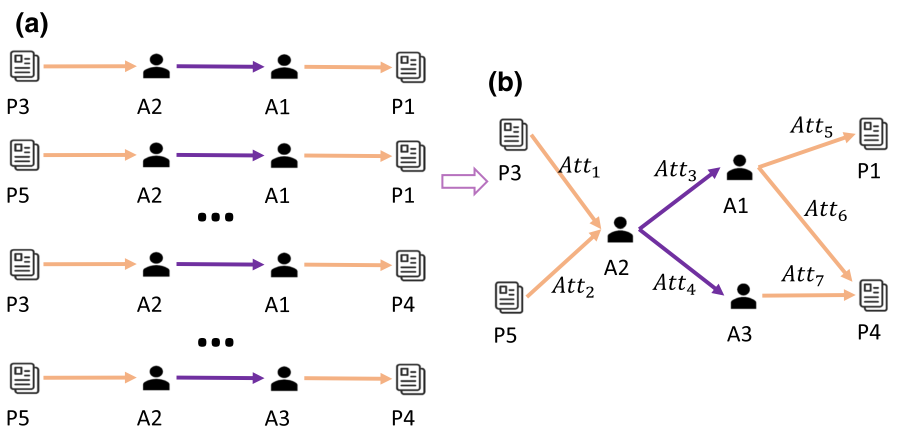

Fig. 4 Comparison between information aggregation by conventional meta-path instances and the proposed redundancy-free computation. **a** The sequential aggregation path instances generated from the meta-path Ω: $P \xrightarrow{Written\_by} \mathrm{A} \xrightarrow{Cite} \mathrm{A} \xrightarrow{Write} \mathrm{P}$ (*A*: *Author*, *P*: *Paper*). **b** The improved aggregation structure by our redundancy-free computation method with attention scores to distinguish messages from different nodes

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{\mathcal{T}\sim U(\mathcal{D})}[(\mathcal{R}(s_t, a_t) \quad + \gamma \max_{a_{t+1}\in A} \hat{Q}(s_{t+1}, a_{t+1}; \theta^-) - Q(s_t, a_t; \theta))^2], \tag{4}$$

where $\mathcal{T} = (s_t, a_t, s_{t+1}, \mathcal{R}(s_t, a_t))$ is randomly sampled replay memory from the *memory buffer* $\mathcal{D}$, θ^- is the set of parameters in the *separate target Q-network* $\hat{Q}$, $\max_{a_{t+1}\in A} \hat{Q}(s_{t+1}, a_{t+1}; \theta^-)$ stands for the optimal target value, and $Q(s_t, a_t; \theta)$ is the predicted reward value based on the training Q-network. We optimise the Q-network's parameters by minimising the loss function.

4.2 Information aggregation with personalised meta-paths

Due to the heterogeneity of nodes in HIN, various types of nodes have different attribute spaces. Before the information aggregation, we first do a node type-specific transformation to project the latent features of different node types into the same space, given by: $H^0[v_i] = W_{\omega_i} \cdot x_i$, where x_i and $H^0[v_i]$ are the original attributes and projected features of node v_i, and W_{ω_i} is a learnable transformation matrix for the type node $\omega_i = \phi(v_i)$. Next, we perform the information aggregation using the generated meta-path instances to learn effective node representations for the downstream task. Take node v_i as an example, and its corresponding obtained meta-path is assumed to be Ω.

We first generate meta-path instances $\{p_1, p_2, \ldots\}$ based on meta-path Ω. As the example shown in Fig. 4, there could be many meta-path instances that follow meta-path Ω, (e.g., Ω : *Paper* $\xrightarrow{Written\_by}$ *Author* $\xrightarrow{Cite}$ *Author* $\xrightarrow{Write}$ *Paper*). An intuitive approach to performing information aggregation is to adopt sequential aggregation, e.g., HetGNN Zhang et al. (2019) and HAN Wang et al. (2019). But we argue that these aggregated paths have computation redundancy, and the approach to aggregation can be further improved. For example, $P_3 \to A_2$ and $P_5 \to A_2$ are repeatedly calculated in the first aggregation step. We aim to merge these two instances into one process $\{P_3, P_5\} \to A_2$ to reduce redundant computations, termed as *redundancy-free* aggregation. Besides, we learn the attention scores $Att(v_i, v_j)$, where nodes v_i and v_j are two ends of a link, for each link in the aggregation path so that messages on different nodes can be distinguished from each other.

Let $H^0[v_i]$ be the projected features of node v_i involved in the instance set $\{p_1, p_2, \dots\}$ of meta-path Ω. We give the updating procedure of $H[v_i]$:

$$H^l[v_i] = \underset{v_j \in I(v_i)}{Aggregate}\left(Att(v_i, v_j) \cdot H^{l-1}[v_j]\right), \tag{5}$$

where $I(v_i)$ indicates the set of past nodes of v_i in $\{p_1, p_2, \dots\}$, $l \in \{1, 2, \dots, t\}$ is the id of aggregator to perform information aggregation, and $Aggregate(\cdot) = Relu(Mean(\cdot))$. The operator $Att(\cdot)$ calculates the importance of the received messages using the relation between past messages and node features, given by:

$$Att(i, j) = \underset{j \in I(v_i)}{Softmax}\left(LeakyRelu\left(WH^{l-1}[v_j] \parallel WH^{l-1}[v_i]\right)\right), \tag{6}$$

where W are the trainable parameters, and $\parallel$ is the concatenation operator. Sect. 5.4 presents an empirical study to reveal how the redundancy-free aggregation affects the time efficiency.

Algorithm 1 PM-HGNN

```
**Input:** HIN $G = (V, E)$, Max step $T$, Number of RL agent training epochs $K$, Number of HGNN training epochs $B$,
**Output:** Node representations $H^T$
1: Initialise memory buffer $\mathcal{D}$ and Q-function
2: for $k \leftarrow \{1, 2, \dots, K\}$ do Initialise information aggregators (HGNN)
3:     for $t \leftarrow \{0, 1, 2, \dots, T\}$ do
4:         Sample a batch of nodes $V_j$ from $V$
5:         for $v_i \in V_j$ do
6:             Obtain state $s_t[v_i]$ via Eq. 2;
7:             Get action $a_t[v_i] = argmax_{a \in A} \mathcal{Q}(s_t[v_i], a; \theta)$
8:             Extend $\Omega_t[v_i]$ with $a_t[v_i]$ and generate $\{p_0, p_1, \dots\}$
9:         end for
10:        for $\beta \leftarrow \{1, 2, \dots, B\}$ do
11:            Perform information aggregation on $\{p_0, p_1, \dots\}$ via Eq. 5
12:            Optimise HGNN' parameters via Loss-function $\mathcal{L}$
13:        end for
14:        Information aggregation on $p_t$ with trained HGNN via Eq. 5
15:        Obtain the learned node representations $H^t$
16:        Obtain $\mathcal{R}(s_t, a_t)$ on validation dataset via Eq. 3
17:        Store the quartuple $\mathcal{T} = (s_t, a_t, s_{t+1}, \mathcal{R}(s_t, a_t))$ into $\mathcal{D}$
18:        Optimise Q-function using the data in $\mathcal{D}$ via Loss-function-Q (Eq. 4)
19:    end for
20: end for
```

4.3 Model training

Training of HGNN Finally, the updated node representation $H^t[v_i]$ can be applied to downstream tasks. Take semi-supervised node classification as an example, with a small fraction of labelled nodes, we can optimise the HGNN's parameters by minimising the Cross-Entropy loss $\mathcal{L}$ via back-propagation and gradient descent: $\mathcal{L} = -\sum_{v \in \mathcal{V}_L} \sum_{c=0}^{C-1} y_v[c] \cdot \log(H^t[v][c])$, 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 representation vector of node v. Meanwhile, the learned node representation can be evaluated by task-specific evaluation matrix $\mathcal{M}_{eval}$ for obtaining a reward score. We summarise our PM-HGNN framework in Algorithm 1. Note that PM-HGNN framework maintains the ability of HGNN in learning node representations for newly added nodes to the graph. The trained RL agent can adaptively generate a meta-path for the new node to compute its representations.

4.4 PM-HGNN++

As seen in previous sections, PM-HGNN adopts the RL agent to generate meta-paths for HGRL adaptively. However, reviewing the overall workflow of PM-HGNN, we identify two limitations. (1) PM-HGNN neglects the relational structures of HIN while generating meta-paths; (2) HGNN requires a large number of epochs to train their parameters and obtain node representations, which restricts the efficiency of PM-HGNN. To be more specific, PM-HGNN generates 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 as Eq. 2. However, HIN's semantic structure is also important information to assist the meta-path generation. In addition, similar to the training process of other deep neural networks (Kipf and Welling 2017), the encoder of PM-HGNN needs a number of epochs to train their parameters to learn effective node representations in terms of a set of meta-path instances. In this way, if HGNN needs B epochs to train their parameters, then we need $T \times B$ epochs to complete a meta-path generation process with the maximum number of steps T. Meanwhile, the RL agent also needs lots of explorations to obtain numerous samples $(\mathcal{T} = (s_t, a_t, s_{t+1}, \mathcal{R}(s_t, a_t)))$ to train the policy network that can result in a combinatorial scale explosion. In the end, following the average aggregation approach to defining node states, PM-HGNN is not able to distinguish node states with categorical attributes, as shown in Eq. 2.

New State (S). To address the identified limitations, we propose an extension of PM-HGNN, PM-HGNN++, which can utilise the structure information of HIN and significantly accelerate the meta-path generation process. Our solution is to define a novel state instead of Eq. 2. Because we notice that the encoder (i.e., HGNN) of PM-HGNN can summarise node attributes and topological structure information involved in meta-paths to assist meta-path generation. In particular, we utilise the latest node representation vector $H^{t-1}[v_i]$ of meta-path's starting node v_i as the state. The new state (S) can be formally described as:

$$s_t = Normalise(H^{t-1}[i]), \tag{7}$$

where *Normalise* is a normaliser trained on the first B generated states to convert the states into the same distribution. Hence, input will be normalized as $\frac{H[v_i]-H_{mean}}{H_{std}}$, where H_{mean} and H_{std} is calculated by the first B states. Note that *normaliser* is a common trick used in deep RL for stabilising the training when the S is very sparse (Hou et al. 2017). In addition, $H^t[i]$ contains not only node attributes but also the relevant graph structure information of each node, hence s_t of Eq. 7 can distinguish different nodes with categorical attributes. So, it endows PM-HGNN++ with the ability to handle graphs with diverse node attributes.

Algorithm 2 PM-HGNN++

```
   Input: HIN $G = (V, E)$, Max timesteps $T$, Number of RL agent training epochs $K$,
   Output: Node representations $H^T$
1: Initialise information aggregators (HGNN), memory buffer $\mathcal{D}$ and Q-function
2: for $k \leftarrow \{0, 1, K - 1\}$ do
3:     Step $t$ = mode($k$, $T$) + 1
4:     Sample a batch of nodes $V_j$ from $V$
5:     for $v_i \in V_j$ do
6:         Obtain state $s_t[v_i]$ via Eq. 7
7:         Get the action $a_t[v_i] = argmax_{a \in A} Q(s_t[v_i], a; \theta)$
8:         Extend the meta-path $\Omega_t[v_i]$ with $a_t[v_i]$
9:         Generate meta-path instances $\{p_0, p_1, \dots\}$
10:    end for
11:    Perform information aggregation on $\{p_0, p_1, \dots\}$ via Eq. 5
12:    Obtain the learned node representations $H^t$
13:    Obtain $\mathcal{R}(s_t, a_t)$ on validation dataset via Eq. 3
14:    Store the quartuple $\mathcal{T} = (s_t, a_t, s_{t+1}, R_t)$ into $\mathcal{D}$
15:    Optimise HGNN's parameters via Loss-function $\mathcal{L}$
16:    Optimise Q-function using the data in $\mathcal{D}$ via Loss-function-Q (Eq. 4)
17: end for
```

New training process. The training process can be further improved based on the new definition of the state. The learned node representation can be further used to optimise HGNN's parameters and update the RL agent. We achieve such a kind of mutual optimisation between the RL agent and HGNN since the meta-paths are generated based on the current status of HGNN. With this novel training process, we only need T epochs to complete a meta-path generation procedure because we do not need to wait for the encoder to complete an entire training process, instead of $T \times B$ epochs. We summarise PM-HGNN++ in Algorithm 2. We perform empirical analysis to compare the time consuming of PM-HGNN and PM-HGNN++ in Sect. 5.4. Note that, similar to PM-HGNN, PM-HGNN++ framework can adaptively generate a meta-path for the new node to compute its representation, which maintains the ability of HGNN to learn representations for newly added nodes.

4.5 Model analysis

The proposed models can deal with various types of nodes and relations and fuse rich semantics in HINs. Benefiting from the RL agent, personalised meta-paths are generated for different nodes, and the HGNN encoder allows information to transfer between nodes via diverse relations. The RL agent that we adopted in this paper, i.e., Deep Q-learning (DQN), is hard to give the precise computation complexity (Fan et al. 2020), hence we give the empirical meta-path generation time in IMDb and DBLP datasets in Sect. 5.4 (Fig. 8). Given a meta-path Ω and the dimensionality of (hidden) node representations is d. The time complexity of the node representation learning process is $\mathcal{O}(V_\Omega d^2 + E_\Omega d)$, where V_Ω is the number of nodes following the meta-path Ω and E_Ω is the number of meta-paths based node pairs. $\mathcal{O}(V_\Omega d^2)$ accounts for the representation transformation process and $\mathcal{O}(E_\Omega d)$ represents the $Att(\cdot)$ computation process of relevant edges. To further unfold the relationship between the complexity of generated meta-paths and the performance, we also report how the maximum step T and the number of aggregation paths affect the node classification performance based on the IMDb dataset in Sect. 5.4 (Table 6).

5 Experiments

5.1 Experimental settings

Datasets We adopt three HIN datasets (IMDb and DBLP, ACM) (Wang et al. 2019; Fu et al. 2020) from different domains to evaluate our models' performance. Statistic information is summarised in Table 4 and the meta-relation schema are shown in Fig. 9. The detailed description of datasets refers to Appendix A.

Competing methods and model configuration We compare their performance against various state-of-the-art models. They include 5 homogeneous graph representation learning models: LINE Tang et al. (2015), DeepWalk Perozzi et al. (2014), MLP, GCN Kipf and Welling (2017), GAT Velickovic et al. (2018); and 10 HGRL models: Esim Shang et al. (2016), metapath2vec Dong et al. (2017), JUST Hussein et al. (2018), HERec Shi et al. (2019), NSHE Zhao et al. (2020), RGCN Schlichtkrull et al. (2019), HAN Wang et al. (2019), GTN Yun et al. (2019), MAGNN Fu et al. (2020) and HGT Hu et al. (2020); and 1 state-of-the-art relational learning model PropStar (Lavrac et al. 2020). Note that FSPG Meng et al. (2015) AutoPath Yang et al. (2018) and MPDRL Wan et al. (2020) do not support node-wise tasks since they learn paths that explicitly represent the similarity between pairs of nodes (as discussed in Sec. 2). Therefore we cannot compare them. The detailed model description and configuration for implementation refer to Appendix C and Appendix E.

Evaluation settings We evaluate our models under node classification. For the semi-supervised setting, we adopt the same settings as in MAGNN Fu et al. (2020) to use 400 instances for training and another 400 instances for validation, 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 percentage of nodes for training and validation, respectively. Use the rest of the nodes for testing. We report the average *Micro-F1* and *Macro-F1* scores of 10 runs with different seeds.

Table 1 Experiment results (%) on the IMDb, DBLP and ACM datasets for the node classification task with unsupervised and semi-supervised settings. MP2V stands for metapath2vec

			Unsupervised (w/ SVM)							Semi-supervised							
Dataset	Train	LINE	DeepWalk	ESim	MP2V	JUST	HERec	NSHE	GCN	GAT	HAN	GTN	HGT	PropStar	MAGNN	Ours	Ours++
IMDb Micro-F1	20%	45.21	49.94	49.32	47.22	50.62	46.23	51.69	52.80	53.64	56.32	58.64	57.49	58.23	59.60	62.14	**62.53**
	40%	46.92	51.77	51.21	48.17	52.25	47.89	53.17	53.76	55.56	57.32	59.73	58.27	59.29	60.50	62.18	**63.10**
	60%	48.35	52.79	52.53	49.87	52.28	48.19	54.04	54.23	56.47	58.42	60.64	59.22	59.80	60.88	62.69	**63.13**
	80%	48.98	52.72	52.54	50.50	53.54	49.11	54.25	54.63	57.40	59.24	61.58	60.16	60.24	61.53	62.86	**64.07**
IMDb Macro-F1	20%	44.04	49.00	48.37	46.05	50.33	45.61	51.93	52.73	53.64	56.19	58.39	57.32	58.03	59.35	61.87	**62.37**
	40%	45.45	50.63	50.09	47.57	52.04	46.80	53.75	53.67	55.50	56.15	59.82	58.19	58.06	60.27	62.17	**62.95**
	60%	47.09	51.65	51.45	48.17	53.22	46.84	54.52	54.24	56.46	57.29	60.35	59.31	59.31	60.66	62.50	**62.98**
	80%	47.49	51.49	51.37	49.99	53.67	47.73	54.38	54.77	57.43	58.51	61.17	60.14	59.93	61.44	62.65	**63.98**
DBLP Micro-F1	20%	87.68	87.21	91.21	89.02	90.40	91.49	92.20	88.51	91.61	92.33	93.41	92.60	91.79	93.61	91.73	**94.34**
	40%	89.25	88.51	92.05	90.36	91.27	92.05	92.63	89.22	91.77	92.57	93.50	93.01	92.04	93.68	92.01	**95.78**
	60%	89.34	89.09	92.28	90.94	91.82	92.66	92.77	89.57	91.97	92.72	93.68	93.20	92.49	93.99	92.67	**95.92**
	80%	89.96	89.37	92.68	91.31	92.15	92.78	93.20	90.33	92.24	93.23	94.32	93.97	93.16	94.47	93.91	**96.63**
DBLP Macro-F1	20%	87.16	86.70	90.68	88.47	90.18	90.82	91.89	88.00	91.05	91.69	92.97	91.74	91.35	93.13	91.63	**94.94**
	40%	88.85	88.07	91.61	89.91	90.44	91.44	91.82	89.00	91.24	91.96	92.88	92.21	92.07	93.23	92.06	**95.42**
	60%	88.93	88.69	91.84	90.50	91.10	92.08	92.15	89.43	91.42	92.14	93.30	92.81	92.41	93.57	92.76	**95.57**
	80%	89.51	88.93	92.27	90.86	91.96	92.25	92.39	89.98	91.73	92.50	94.02	93.16	92.82	94.10	94.89	**96.30**
ACM Micro-F1	20%	84.41	84.22	87.10	85.00	87.76	88.13	87.10	86.12	88.45	88.89	89.06	88.67	87.82	90.16	89.62	**91.38**
	40%	85.14	86.33	88.53	87.36	88.36	89.28	88.53	86.64	88.68	89.06	89.71	89.03	88.36	90.53	90.51	**91.53**

Table 1 continued

			Unsupervised (w/ SVM)							Semi-supervised							
Dataset	Train	LINE	DeepWalk	ESim	MP2V	JUST	HERec	NSHE	GCN	GAT	HAN	GTN	HGT	PropStar	MAGNN	Ours	Ours++
	60%	86.03	86.95	89.64	88.75	89.85	89.90	89.08	87.00	88.81	89.92	90.95	90.09	88.65	90.87	90.90	**91.98**
	80%	87.16	87.12	89.75	88.97	90.05	90.09	89.74	88.04	89.41	90.04	91.60	90.62	89.21	91.01	91.01	**92.99**
ACM Macro-F1	20%	83.41	83.13	86.12	84.36	86.49	87.58	86.62	85.61	87.16	87.93	88.12	87.20	87.64	89.06	88.78	**90.03**
	40%	84.06	85.37	87.82	86.63	87.15	88.28	87.12	85.93	87.43	88.05	88.67	88.14	87.95	89.38	89.46	**91.08**
	60%	85.98	85.73	88.08	87.48	88.39	88.95	88.85	86.66	87.71	88.26	90.85	89.09	87.48	89.49	89.61	**91.12**
	80%	86.88	85.85	89.64	88.37	88.70	89.06	89.50	87.00	88.02	88.88	90.90	90.02	88.59	90.71	90.25	**92.62**

Best performance are given in bold

5.2 Experimental results

Semi-supervised node classification. We present the results in Table 1, in which competing methods that cannot support semi-supervised settings utilise the unsupervised settings. PM-HGNN++ performs consistently better than all competing methods across different proportions and datasets. On IMDb, the performance gain obtained by PM-HGNN++ over the best competing method (MAGNN) is around $(3.7\% - 5.08\%)$. GNN models designed for homogeneous and heterogeneous graphs perform better than shallow HGRL models. This demonstrates that the modelling of heterogeneous node attributes improves the performance significantly. On DBLP and ACM, the performances of all models are overall better compared with the results on IMDb. It is interesting to observe that different from the results on IMDb, the shallow heterogeneous network methods, i.e., JUST and NSHE obtain better performances than a few number of homogeneous GNNs. That said, the heterogeneous relations on DBLP and ACM are useful for the node classification task. PM-HGNN++ apparently outperforms the strongest competing method (i.e., MAGNN) up to 2.4%, showing the generality and superiority of PM-HGNN++. In addition, among unsupervised competing methods, NSHE obtains better performance than other unsupervised methods, showing that network schema in HIN is helpful to obtain better representations.

Supervised node classification. We present the results in Table 2. We can see that PM-HGNN++ consistently outperforms all competing models on IMDb, DBLP and ACM datasets, with up to 5.6% in terms of *Micro-F1*. Heterogeneous GNNs outperform homogeneous GNNs, and our PM-HGNN++ achieves the best performance. This demonstrates the usefulness of heterogeneous relations and the advantages of generating appropriate personalised meta-paths for each node according to its attributes and relational structure.

5.3 Meta-path analysis

Meta-path generation process visualisation We visualise how the RL agent in PM-HGNN++ generates personalised meta-paths for each target node in Figs. 10,11 and Figs. 12.10 summarises the actions made by the RL agent on IMDb with different max steps under the semi-supervised setting. The percentages marked in the figure present the fraction of nodes choosing the corresponding relation to extend the meta-path at that step. For example, the meta-path $M \xrightarrow{64.8\%} A \xrightarrow{8.8\%} M$ with $T = 2$. 64.8% means there are 64.8% *Movie* nodes select *MA* relation at step-1, and 8.8% means among those *Movie* nodes who selected *MA* at step-1, there are 8.8% nodes selecting *AM* to extend the meta-path. It is interesting to see that the RL agent selects the relation *MA* at the first step more often than the other one *MD*. It means that a movie's characteristic is more related to its actors than its director. Besides, when step $T = 2$, the RL agent selects plenty of *STOP* actions. This shows that two short meta-paths, i.e., $Movie \xrightarrow{MA} Actor$ and $Movie \xrightarrow{MD} Director$ are informative enough for the majority of nodes to learn effective representations. Moreover, there are $6.7\% - 9.4\%$ nodes that do not need any meta-paths to learn their representations. This implies that their attributes can provide enough information. This has also been reflected in the results

Table 2 Experiment results (%) on the IMDb, DBLP and ACM datasets for the node classification task with supervised settings

Dataset	Train	MLP	GCN	GAT	RGCN	HAN	GTN	HGT	PropStar	MAGNN	Ours	Ours++
IMDb Micro-F1	5%	47.60	52.18	53.32	54.09	55.74	55.42	55.80	55.28	54.77	54.90	**57.09**
	10%	49.75	55.81	56.53	58.22	59.21	60.11	60.06	59.82	59.26	60.97	**62.27**
	15%	54.62	58.55	59.13	60.13	60.25	60.76	60.67	60.83	60.03	61.12	**63.77**
	20%	55.22	59.13	60.22	60.15	60.70	61.58	61.71	61.77	60.21	62.33	**64.65**
IMDb Macro-F1	5%	47.67	51.89	53.17	54.20	55.53	58.39	55.36	55.16	54.36	54.31	**56.47**
	10%	49.07	54.96	56.32	58.17	58.57	59.82	60.13	60.19	58.22	61.02	**62.04**
	15%	54.86	57.23	58.82	60.02	60.03	60.35	60.42	60.60	59.76	60.93	**63.81**
	20%	55.02	58.56	59.90	60.14	60.44	61.17	61.43	61.86	60.28	62.07	**63.89**
DBLP Micro-F1	5%	69.25	78.94	78.01	81.98	90.82	90.90	90.08	90.80	90.22	89.11	**92.52**
	10%	75.75	82.53	82.69	82.73	92.17	92.25	92.20	92.57	91.96	90.43	**93.87**
	15%	76.87	83.50	84.61	83.29	92.62	92.77	92.82	92.93	92.31	91.20	**94.72**
	20%	77.95	83.94	85.07	84.67	93.16	93.24	93.21	93.45	93.02	91.98	**95.13**
DBLP Macro-F1	5%	68.50	78.23	76.74	81.11	90.24	90.32	89.93	89.98	90.06	88.63	**92.20**
	10%	74.85	81.46	81.59	82.65	91.59	91.99	91.21	91.12	91.31	89.70	**93.34**
	15%	76.17	82.77	82.00	83.79	91.90	92.45	92.90	92.30	92.09	90.84	**94.14**
	20%	76.86	83.05	84.34	84.11	92.13	93.03	92.97	93.05	92.57	91.31	**94.33**
ACM Micro-F1	5%	70.30	76.23	77.35	79.35	88.83	88.27	88.50	88.35	88.29	88.86	**90.53**
	10%	73.55	80.85	80.28	80.95	90.08	90.18	90.16	89.52	90.37	89.31	**92.72**
	15%	75.22	82.33	82.26	82.46	91.84	91.66	91.87	90.11	90.76	89.77	**92.88**
	20%	76.18	82.96	84.32	83.08	92.56	92.06	91.96	91.56	91.22	90.99	**93.93**

Table 2 continued

Dataset	Train	MLP	GCN	GAT	RGCN	HAN	GTN	HGT	PropStar	MAGNN	Ours	Ours++
ACM Macro-F1	5%	71.80	77.01	77.53	80.85	88.73	87.57	89.51	89.79	89.46	88.62	**90.47**
	10%	72.56	79.23	79.60	81.37	89.40	88.99	90.12	90.15	90.02	89.06	**91.20**
	15%	73.73	80.42	80.39	81.79	90.73	89.58	90.80	90.87	90.42	89.21	**92.04**
	20%	74.86	81.07	81.64	82.86	91.80	90.71	91.40	91.13	91.32	90.87	**93.55**

Best performance are given in bold

Table 3 Meta-paths designed by PM-HGNN++ on IMDb

T	Meta-paths Designed by PM-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
	Movie→*Actor*	42.7
	Movie→*Director*	25.8
$T = 3$ *Movie*	→*Actor*→*Movie*→*Actor*	5.9
	Movie→*Actor*→*Movie*→*Director*	4.9
	Movie→*Actor*→*Movie*	3.2
Manual	Movie→Actor→Movie,Movie→Director→Movie	
GTN Yun et al. (2019)	Movie→ Director, Movie→ Actor	
	Movie→Director→Movie	

of MLP in Sect. 5.2, which does not utilise any structural information but only node attributes. More analyses on Figs. 11 and12 can be found in Appendix B.

Overall meta-path statistics. We further present meta-paths generated for most of the target nodes in Table 3,[2] which are summarised from Fig. 10. Table 6 and7 present the generated meta-paths on DBLP and ACM, respectively. We can find from Table 3 that the RL agent can generate the same meta-paths as manually-defined ones. Besides, we find that these meta-paths specified by human experts are not the most useful ones to learn node representations for *Movie* nodes. Two short meta-paths $Movie \xrightarrow{MA} Actor$ and $Movie \xrightarrow{MD} Director$ are the most useful ones. This indicates that participants (director and actors) of the movie largely determine its type as the task is to predict the class of a movie. The top three meta-paths generated by the most potent competing method, GTN, confirm our observation that two shorter meta-paths are more valuable to learn *Movie* nodes' representations. However, GTN can only select several meta-paths for every node type; our model can identify personalised meta-paths for each node. More analyses on Tables 6 and 7 can be found in Appendix B.

Meta-path comparison In order to understand how does PM-HGNN++ generate personalised meta-paths for different nodes, we hereby present another analysis to investigate the designed meta-path lengths for nodes that can be correctly classified according to raw node attributes. Specifically, we devide nodes into two groups, *pos* and *neg*, according to whether applying multi-layer perceptron (MLP) with node attributes can produce correct classification under supervised settings. That said, correctly-classified nodes and incorrectly-classified ones are grouped into *pos* and *neg*, respectively. Then, we calculate the average lengths of PM-HGNN++ generated meta-paths for nodes belonging to two groups. The results are reported in Fig. 5. We can

[2] We do not present the relation types for all meta-paths in3.

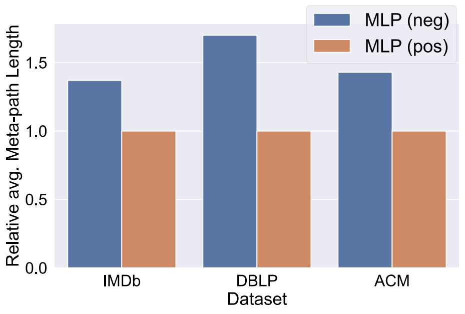

Fig. 5 The average meta-path length generated by PM-HGNN++ on different datasets. MLP (pos)/ MLP (neg) are groups of nodes that MLP makes correct and wrong classifications, respectively

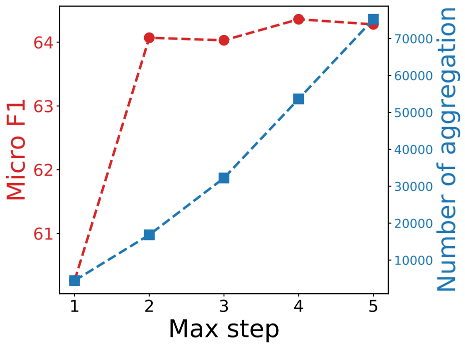

Fig. 6 Micro-F1 and the number of aggregations of PM-HGNN++ on IMDb, with different max steps

find that nodes that cannot be correctly classified by MLP are associated with longer personalised meta-paths. Such results deliver an important insight: PM-HGNN++ tends to explore deeper relational neighbouring structure to enhance node representation learning, i.e., discriminating nodes with different labels from one another, if node attributes themselves cannot provide sufficient information. Moreover, in the DBLP dataset, which has more complicated heterogeneous semantic patterns (4 node types and 6 relation types), node group MLP (*neg*) has relatively longer meta-paths because PM-HGNN++ can have more options to explore and generate personalised meta-paths.

5.4 Model snalysis on PM-HGNN++

Maximum step T. We study how the influence of maximum step T and the number of aggregation paths affect on the classification performance. The results on IMDb dataset are presented in Fig. 6. Setting $T \geq 2$ leads to the most significant performance improvement in terms of Micro F1 over $T = 1$. As T increases, the number of aggregation paths also increases. Considering that more aggregation paths bring higher computational cost, we choose $T = 2$ for the experiments in Sect. 5.2, even though $T > 2$ can produce better performance.

Redundancy-free aggregation We investigate the influence of the redundancy-free improvement according to the number of aggregations with/without the improvement. Fig. 7 shows the results. We can find that our redundancy-free aggregation strategy

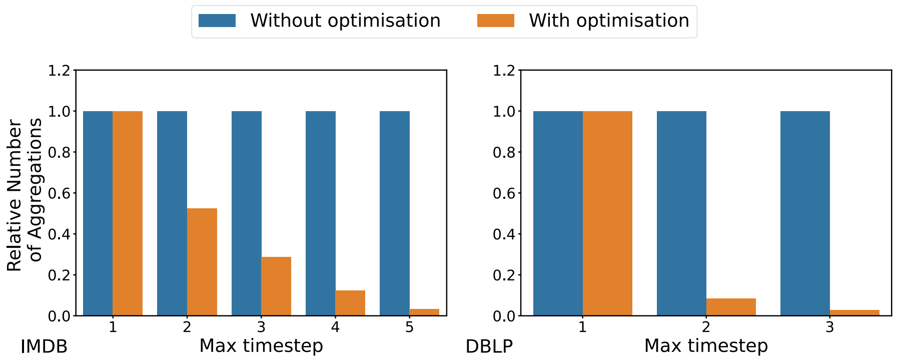

Fig. 7 The relative numbers of aggregations (# of without the redundancy-free aggregation divided by # of with the redundancy-free aggregation) under different maximum steps on IMDb (left) and DBLP (right)

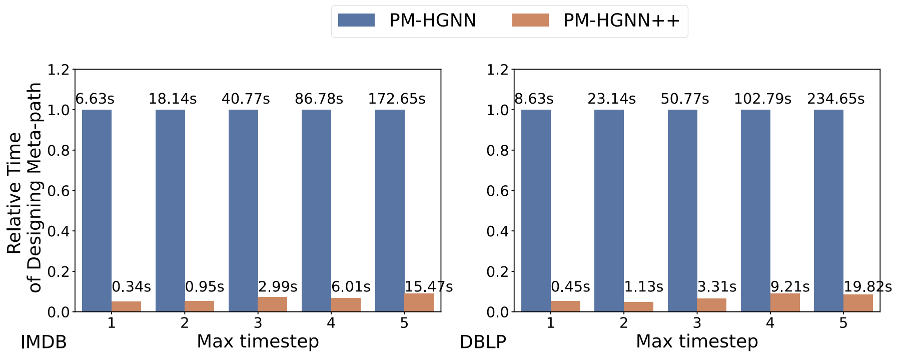

Fig. 8 Run time analysis of meta-path generation for PM-HGNN and PM-HGNN++ under different max steps based on IMDb (left) and DBLP (right) datasets. The x-axis is the number of maximum timestep in generating meta-paths, and the y-axis is the relative time (i.e., the run time of PM-HGNN++ divided by the run time of PM-HGNN). Besides, the numbers at tops of bars indicate the run time values in seconds

significantly reduces the number of information aggregations. Besides, as the maximum step T increases, the reduction effect is more obvious. On IMDb, our method reduces near 50% aggregations with $T = 2$, and the reduced ratio is more than 80% when $T = 4$.

Run time analysis of meta-path generation As described in Sect. 4.4, PM-HGNN++ can accelerate the meta-path generation process through the proposed novel training process. Here we compare the run time of PM-HGNN and PM-HGNN++ for personalised meta-paths generation. We report their actual run time in seconds, and calculate the run time of PM-HGNN++ relative to PM-HGNN (i.e., relative time). The results on IMDb and DBLP datasets are shown in Fig. 8. We can find that PM-HGNN++ is less than 5% of PM-HGNN's run time on both datasets. Such results exhibit the promising time efficiency of PM-HGNN++.

6 Conclusions and future work

We have studied in this paper the HGRL problem and identified the limitation of existing HGRL methods, i.e., mainly due to their dependency on hand-crafted meta-paths. In order to fully unleash the power of HGRL, we presented a novel framework PM-HGNN and proposed one extension model PM-HGNN++. Compared with existing HGRL models, the most significant advantages of our framework lie in avoiding manual efforts in defining meta-paths of HGRL and generating personalised meta-paths for each individual node. The experimental results demonstrated that our framework generally outperforms the competing approaches and discovers useful meta-paths that have been ignored by human expertise. In the future, we plan to extend our framework to other tasks on HINs, such as online recommendation and knowledge graph completion and understanding PM-HGNN's generated meta-paths is another promising direction.

Acknowledgements This work is supported by the Luxembourg National Research Fund through grant PRIDE15/10621687/SPsquared, and supported by Ministry of Science and Technology (MOST) of Taiwan under grants 110-2221-E-006-136-MY3, 110-2221-E-006-001, and 110-2634-F-002-051.

A Datasets

The statistics of datasets are summarised in Table 4. Detailed descriptions of two datasets are presented as follows:

IMDb[3] is an online dataset about movies and television programs, including information such as cast, production crew and plot summaries. We extract a subset of IMDb that 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 attribute of each movie corresponds to elements of a bag of words (i.e., their plot keywords, 1, 232 in total).

DBLP[4] is an online computer science bibliography. We extract a subset of DBLP that 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).

[3] https://www.imdb.com/

[4] https://dblp.uni-trier.de/

Table 4 Statistics of the datasets

Dataset	Node	Relation	# Attributes
	# Movie (M): 4,278	# M-D, # D-M: 4,278	1,232
IMDb	# Director (D): 2,081	# M-A, # A-M: 12,828	
	# Actor (A): 5,257		
DBLP	# Author (A): 4,057	# A-P # P-A: 19,645	334
	# Paper (P): 14,328	# P-T # T-P: 85,810	
	# Term (T): 7,723	# P-V # V-P: 14,328	
	# Venue (V): 20		
ACM	# Paper (P): 3,025	# A-P # P-A: 9,744	1,830
	# Author (A): 5,835	# P-S # S-P: 3,025	
	# Subject (S): 56		

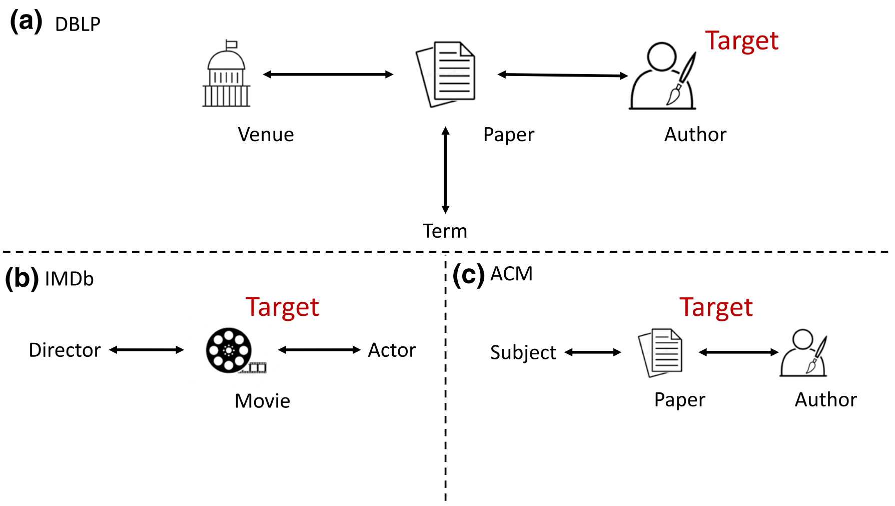

Fig. 9 Meta-relation schema of three datasets

ACM[5] is an online academic publication dataset. We extract papers published in KDD, SIGMOD, SIGCOMM, MobiCOMM and VLDB and divided the papers into three classes (*Database*, *Wireless Communication*, and *Data Mining*). Then, we construct a HIN that comprises $3,025$ papers, $5,835$ authors and 56 subjects. Paper features correspond to elements of a bag of words represented by keywords. We label the papers according to the conference they published.

5 https://dl.acm.org/

Table 5 All possible meta-paths on different datasets. IMDb: Movie (M), Director (D), Actor (A); DBLP: Author (A), Paper (P), Term (T), Venue (V); ACM: Paper (P), Author (A), Subject (S)

t	IMDb	DBLP	ACM
$t = 1$	$M \to A$	$A \to P$	$P \to A$
	$M \to D$		$P \to S$
$t = 2$	$M \to A \to M$	$A \to P \to A$	$P \to S \to P$
	$M \to D \to M$	$A \to P \to V$	$P \to A \to P$
		$A \to P \to T$	
	$M \to A \to M \to A$	$A \to P \to A \to P$	$P \to S \to P \to S$
$t = 3$	$M \to A \to M \to D$		$P \to S \to P \to A$
	$M \to D \to M \to A$	$A \to P \to V \to P$	$P \to A \to P \to S$
	$M \to D \to M \to D$	$A \to P \to T \to P$	$P \to A \to P \to A$
	$A \to P \to A \to P \to A$		
	$A \to P \to A \to P \to V$		
	$A \to P \to A \to P \to T$	$M \to A \to M \to A \to M$	$P \to S \to P \to S \to P$
	$M \to A \to M \to D \to M$	$A \to P \to V \to P \to A$	$P \to S \to P \to A \to P$
$t = 4$	$M \to D \to M \to A \to M$	$A \to P \to V \to P \to V$	$P \to A \to P \to S \to P$
	$M \to D \to M \to D \to M$	$A \to P \to V \to P \to T$	$P \to A \to P \to A \to P$
	$A \to P \to T \to P \to A$		
	$A \to P \to T \to P \to V$		
	$A \to P \to T \to P \to T$		

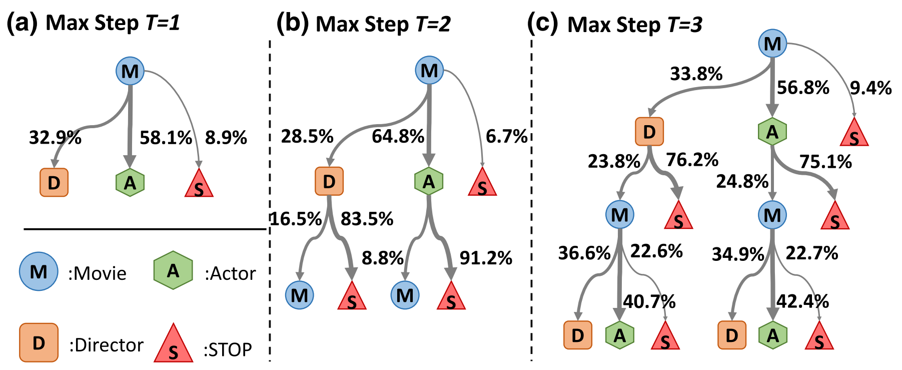

Fig. 10 Actions that the RL agent takes on IMDb: **a**, **b** and **c** correspond to PM-HGNN++ with max step $T = 1, 2, 3$, respectively. The red triangles with "S" indicate the "STOP" action. The thickness of links represents the ratio of the corresponding action (Color figure online)

B More Meta-paths Analysis

We have introduced three datasets that we used in the experimental Sections in Sect. 5.1 and Section A. Here we further present the set of meta-paths that are possibly generated in each step in Table 5.[6] It should be noted that as we discussed in Section 4, there is always an available *STOP* action at each step. That said, PM-HGNN variants allow to design meta-paths with flexible lengths at each step.

We present more meta-path analysis here. Figures 11[7] and12 visualise how the reinforcement learning agent in PM-HGNN++ generates personalised meta-paths for each target node on DBLP and ACM datasets with different max steps under the semi-supervised setting, respectively The percentages marked in the figure represent the fraction of nodes choosing the corresponding relation to extend the meta-path at that step. Tables 6 and 7 summarise the predefined meta-paths, the found important meta-paths by GTN Yun et al. (2019) and the top-frequent personalised meta-paths designed by PM-HGNN++ for the target nodes of DBLP and ACM datasets, respectively. The meta-path generation process is shown in Figures 11 and 12, the RL agent makes decisions to extend the meta-paths for different nodes according to the defined state S. Note that, we only present the top-5 frequent meta-paths if there are more possibilities.

C Competing Methods

We adopt 5 homogeneous graph representation learning models and 10 heterogeneous graph representation learning models and 1 state-of-the-art relational learning model as competing methods.

[6] We only discuss the meta-paths that started with target nodes of each dataset.

[7] We present the visualisation results of $T < 4$, since the figure will be too dense to see the content clearly with $T = 4$

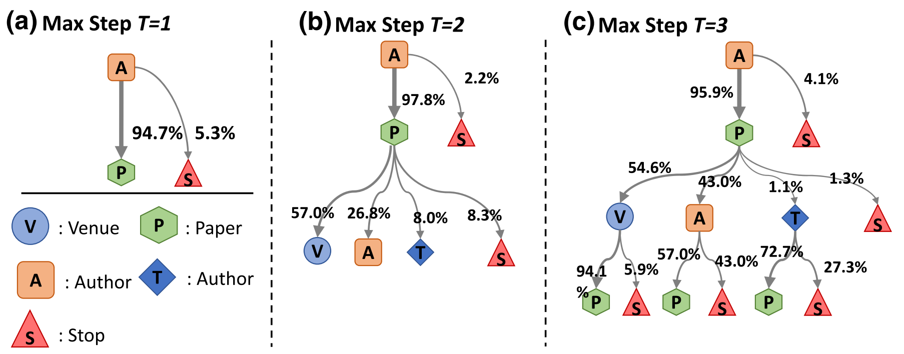

Fig. 11 Actions that the RL agent takes on DBLP: **a**, **b** and **c** correspond to PM-HGNN++ with max step $T = 1, 2, 3$, respectively. The red triangles with "S" indicate the "STOP" action. The thickness of links represents the ratio of the corresponding action (Color figure online)

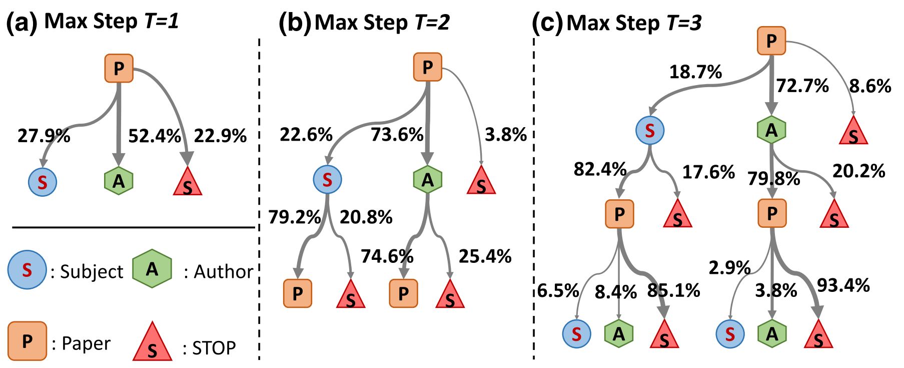

Fig. 12 Actions that the RL agent takes on ACM: **a**, **b** and (c) correspond to PM-HGNN++ with max step $T = 1, 2, 3$, respectively. The red triangles with a black "S" indicate the "STOP" action. The thickness of links represents the ratio of the corresponding action (Color figure online)

1. **LINE** Tang et al. (2015) is a traditional homogeneous model that exploits the first-order and second-order proximity between nodes. We apply it to the datasets by ignoring heterogeneous information network heterogeneity and node attributes.
2. **DeepWalk** Perozzi et al. (2014) is a random walk-based graph representation learning method for homogeneous graphs, we apply it to the datasets by ignoring heterogeneous information network heterogeneity and node attributes.
3. **ESim** Shang et al. (2016) is a heterogeneous graph representation learning 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 the same weight.
4. **metapath2vec** Dong et al. (2017) is a heterogeneous graph representation learning approach that performs random walks on heterogeneous information networks with the guidance of pre-defined meta-paths and utilises skip-gram to generate node representations.
5. **JUST** Hussein et al. (2018) is a heterogeneous graph representation learning method that does not rely on manually defined meta-paths.

6. **HERec** Shi et al. (2019) is a heterogeneous graph representation learning method that designs a type constraint strategy to filter the node sequence and utilises skip-gram to generate node representations.
7. **NSHE** Zhao et al. (2020) is a heterogeneous graph representation learning method that tries to maintain network schema during network representation learning.
8. **MLP** Haykin (1999) is a class of feed-forward neural networks that learns information from node attributes without structural information.
9. **GCN** Kipf and Welling (2017) is a homogeneous graph neural network that extends the convolution operation to graphs. Here we test GCN on meta-path-based homogeneous graphs and report the results from the best meta-path.
10. **GAT** Velickovic et al. (2018) is a homogeneous graph neural network that performs convolution on graphs with an attention mechanism. Similar to the implementation of GCN, here we test GAT on meta-path-based homogeneous graphs and report the results from the best meta-path.
11. **RGCN** Schlichtkrull et al. (2019) is a heterogeneous graph neural network model which keeps a different weight for each relation to perform convolution on graphs. The graph neural network encoder is the same as GCN.
12. **HAN** Wang et al. (2019) is a heterogeneous graph neural network model which learns meta-path guided node representations from different meta-path-based homogeneous graphs and integrates them with using attention.
13. **GTN** Yun et al. (2019) is a heterogeneous graph neural network model which transforms a heterogeneous graph into multiple new graphs defined by given meta-paths and selects from them by transition probability.
14. **PropStar** Lavrac et al. (2020) is state-of-the-art relational learning which uses embedding vectors to represent the features of the data set. Individual relational features obtained as the result of pro-positionalisation by Wordification, are used by supervised embeddings learners to obtain representations, co-located with instance labels.
15. **MAGNN** Fu et al. (2020) is a heterogeneous graph neural network model which defines meta-path instance encoders to extract the structure and semantics in meta-paths to improve generated representations.
16. **HGT** Hu et al. (2020) is a heterogeneous graph neural network model which uses each meta-relation to parameterise transformer-like self-attention architecture to capture common and specific patterns of relationships.

For LINE and DeepWalk, we utilise the integrated implementations from GraphEmbedding.[8] For metapath2vec, RGCN, HAN and HGT, we utilise the integrated implementations from Deep Graph Library.[9] For GCN and GAT, we use the implementation from PytorGeometric.[10] For other methods, we adopt the official implementation

[8] https://github.com/shenweichen/GraphEmbedding

[9] https://www.dgl.ai/

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

Table 6 Meta-paths generated by PM-HGNN++ on DBLP

T	Meta-paths Designed by PM-HGNN++	Percentage (%)
$T = 1$	$Author \rightarrow Paper$	94.7
	$Author \rightarrow Paper \rightarrow Venue$	55.7
$T = 2$	$Author \rightarrow Paper \rightarrow Author$	26.2
	$Author \rightarrow Paper$	8.1
	$Author \rightarrow Paper \rightarrow Term$	7.8
	$Author \rightarrow Paper \rightarrow Venue \rightarrow Paper$	49.3
	$Author \rightarrow Paper \rightarrow Author \rightarrow Paper$	23.5
$T = 3$	$Author \rightarrow Paper \rightarrow Author$	17.7
	$Author \rightarrow Paper \rightarrow Venue$	3.1
	$Author \rightarrow Paper$	1.2
	$Author \rightarrow Paper \rightarrow Venue \rightarrow Paper \rightarrow Author$	32.7
	$Author \rightarrow Paper \rightarrow Author$	25.8
$T = 4$	$Author \rightarrow Paper \rightarrow Author \rightarrow Paper \rightarrow Author$	21.9
	$Author \rightarrow Paper \rightarrow Venue \rightarrow Paper$	14.9
	$Author \rightarrow Paper \rightarrow Venue$	3.2
Manual	$Author \rightarrow Paper \rightarrow Venue \rightarrow Paper \rightarrow Author$	
	$Author \rightarrow Paper \rightarrow Author$	
	$Author \rightarrow Paper \rightarrow Term \rightarrow Paper \rightarrow Author$	
GTN Yun et al. (2019)	$Author \rightarrow Paper \rightarrow Venue \rightarrow Paper \rightarrow Author$	,
	$Author \rightarrow Paper \rightarrow Author \rightarrow Paper \rightarrow Author$	,
	$Author \rightarrow Paper \rightarrow Author$	

Table 7 Meta-paths generated by PM-HGNN++ on ACM

T	Meta-paths Designed by PM-HGNN++	Percentage (%)
$T = 1$	$Paper \rightarrow Author$	52.4
	$Paper \rightarrow Subject$	27.9
$T = 2$	$Paper \rightarrow Author \rightarrow Paper$	54.9
	$Paper \rightarrow Author$	18.7
	$Paper \rightarrow Subject \rightarrow Paper$	17.9
	$Paper \rightarrow Subject$	4.7
	$Paper \rightarrow Author \rightarrow Paper$	54.2
	$Paper \rightarrow Author$	14.7
$T = 3$	$Paper \rightarrow Subject \rightarrow Paper$	13.1
	$Paper \rightarrow Subject$	3.3
	$Paper \rightarrow Author \rightarrow Paper \rightarrow Author$	2.1
Manual	$Paper \rightarrow Author \rightarrow Paper, Paper \rightarrow Subject \rightarrow Paper$	
GTN Yun et al. (2019)	$Paper \rightarrow Author \rightarrow Paper, Paper \rightarrow Subject \rightarrow Paper$	

Table 8 Model comparison from various aspects: Heterogeneous Information Network (HIN), Node-wise Task (NT), End-to-end Training (E2E), Without Manual-defined Meta-paths (WMM), Adaptive Meta-path Generation (AMG), Meta-paths Attached (MPA) to each node-pair ($*$), each node type ($\circ$) or each node ($\bullet$)

	HIN	NT	E2E	WMM	AMG	MPA
LINE Tang et al. (2015)		✓		✓		
DeepWalk Perozzi et al. (2014)		✓		✓		
FSPG Meng et al. (2015)	✓			✓	✓	$*$
Esim Shang et al. (2016)	✓	✓		✓		$\circ$
metapath2vec Dong et al. (2017)	✓	✓		✓		$\circ$
JUST Hussein et al. (2018)	✓	✓		✓	✓	
HERec Shi et al. (2019)	✓	✓		✓		$\circ$
NSHE Zhao et al. (2020)	✓	✓		✓		$\circ$
GCN Kipf and Welling (2017)		✓	✓	✓		
GAT Velickovic et al. (2018)		✓	✓	✓		
RGCN Schlichtkrull et al. (2019)	✓	✓	✓			$\circ$
AutoPath Yang et al. (2018)	✓		✓	✓	✓	$*$
HAN Wang et al. (2019)	✓	✓	✓			$\circ$
GTN Yun et al. (2019)	✓	✓	✓		✓	$\circ$
MAGNN Fu et al. (2020)	✓	✓	✓			$\circ$
HGT Hu et al. (2020)	✓	✓	✓	✓	✓	$\circ$
MPDRL Wan et al. (2020)	✓		✓	✓	✓	$*$
PM-HGNN	✓	✓	✓	✓	✓	$\bullet$

provided in their published papers: ESim,[11] JUST,[12] HERec,[13] NSHE,[14] GTN,[15] PropStart[16] and MAGNN.[17]

D Model comparison

In Section 2, we have systematically discussed related work and highlighted the differences between PM-HGNN and them. Here, we further present Table 8 to summarise the key advantages of PM-HGNN and compares it with a number of recent state-of-the-art methods. PM-HGNN is the first HGRL model that can adaptively generate personalised meta-paths for each individual node to support node-wise tasks and maintain the end-to-end training mechanism.

[11] https://github.com/shangjingbo1226/ESim

[12] https://github.com/eXascaleInfolab/JUST

[13] https://github.com/librahu/HERec

[14] https://github.com/AndyJZhao/NSHE

[15] https://github.com/seongjunyun/Graph_Transformer_Networks

[16] https://github.com/SkBlaz/PropStar

[17] https://github.com/cynricfu/MAGNN

E Model configuration

For the DQN of the proposed PM-HGNN and PM-HGNN++, we use the implementation in Mnih et al. (2015) with a few modifications to fit it with our frameworks. We develop a 5-layers 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 module of PM-HGNN and PM-HGNN++, we randomly initialise parameters and optimise the model with Adam optimiser. 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 steps (T) for IMDb and DBLP are 2 and 4, respectively. For a fair comparison, we set the node representation dimension of all the models mentioned above to 64.

For random walk-based 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 GAT, HAN, and MAGNN, we set the number of attention heads to 8. For HAN and MAGNN, we set the dimension of the attention vector in inter-meta path aggregation to 128. For meta-path guided methods, including Esim, metapath2vec, HERec, HAN and MAGNN, we give them the pre-defined meta-paths as in Wang et al. (2019). For IMDb, there are two meta-paths: $Movie \rightarrow Actor \rightarrow Movie$ and $Movie \rightarrow Director \rightarrow Movie$. For DBLP, there are three meta-paths: $Author \rightarrow Paper \rightarrow Author$, $Author \rightarrow Paper \rightarrow Term \rightarrow Papper \rightarrow Author$, $Author \rightarrow Paper \rightarrow Venue \rightarrow Papper \rightarrow Author$, and $Venue \rightarrow Paper \rightarrow Author$. For the relational learning model, we use the implementation published with the official paper and adopt model settings the same as the official settings for the IMDb dataset. For relational learning and graph neural network-based models, we test them with the same parameters as PM-HGNN on the same data split. Competing models are implemented with Pytorch[18] following the published implementations .

References

Sun Y, Han J (2012) Mining heterogeneous information networks: a structural analysis approach. ACM SIGKDD Explorations Newsletter

Van Otterlo M (2005) A survey of reinforcement learning in relational domains. Centre for Telematics and Information Technology (CTIT) University of Twente, Tech, Rep

Zheng J, Ma Q, Gu H, Zheng Z (2021) Multi-view denoising graph auto-encoders on heterogeneous information networks for cold-start recommendation. In: Proceedings of the 2021 ACM Conference on Knowledge Discovery and Data Mining (KDD), pp. 2338–2348

Wan G, Du B, Pan S, Haffari G (2020) Reinforcement learning based meta-path discovery in large-scale heterogeneous information networks. In: Proceedings of the 2020 AAAI Conference on Artificial Intelligence (AAAI), pp. 6094–6101

Wang X, Ji H, Shi C, Wang B, Ye Y, Cui P, Yu PS (2019) Heterogeneous graph attention network. In: Proceedings of the 2019 International Conference on World Wide Web (WWW), pp. 2022–2032

Fu X, Zhang J, Meng Z, King I (2020) MAGNN: metapath aggregated graph neural network for heterogeneous graph embedding. In: Proceedings of the 2020 International Conference on World Wide Web (WWW), pp. 2331–2341

[18] https://pytorch.org/

Dong Y, Hu Z, Wang K, Sun Y, Tang J (2020) Heterogeneous network representation learning. In: Proceedings of the 2020 International Joint Conferences on Artifical Intelligence (IJCAI), pp. 4861–4867

Dong Y, Chawla N.V, Swami A (2017) metapath2vec: Scalable representation learning for heterogeneous networks. In: Proceedings of the 2017 ACM Conference on Knowledge Discovery and Data Mining (KDD), pp. 135–144

Fu T, Lee W, Lei Z (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), pp. 1797–1806

Shi C, Hu B, Zhao WX, Yu PS (2019) Heterogeneous information network embedding for recommendation. IEEE Transactions on Knowledge and Data Engineering

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionally. In: Proceedings of the 2013 Annual Conference on Neural Information Processing Systems (NIPS), pp. 3111–3119

Le QV, Mikolov T (2014) Distributed representations of sentences and documents. In: Proceedings of the 2014 International Conference on Machine Learning (ICML), pp. 1188–1196

Hussein R, Yang D, Cudré-Mauroux P (2018) Are meta-paths necessary?: Revisiting heterogeneous graph embeddings. In: Proceedings of the 2018 ACM International Conference on Information and Knowledge Management (CIKM), pp. 437–446

Jiang J, Li Z, Ju CJ-, Wang W (2020) MARU: meta-context aware random walks for heterogeneous network representation learning. In: Proceedings of the 2020 ACM International Conference on Information and Knowledge Management (CIKM), pp. 575–584

Zhao J, Wang X, Shi C, Liu Z, Ye Y (2020) Network schema preserving heterogeneous information network embedding. In: Proceedings of the 2020 International Joint Conferences on Artifical Intelligence (IJCAI), pp. 1366–1372

Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: Proceedings of the 2017 International Conference on Learning Representations (ICLR)

Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2018) Graph attention networks. In: Proceedings of the 2018 International Conference on Learning Representations (ICLR)

Schlichtkrull MS, Kipf TN, Bloem P, van den Berg R, Titov I, Welling M (2019) Modeling relational data with graph convolutional networks. In: European Semantic Web Conference (ESWC), pp. 593–607

Zhang C, Song D, Huang C, Swami A, Chawla NV (2019) Heterogeneous graph neural network. In: Proceedings of the 2019 ACM Conference on Knowledge Discovery and Data Mining (KDD), pp. 793–803

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller MA, Fidjeland A, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518:529–533

Meng C, Cheng R, Maniu S, Senellart P, Zhang W (2015) Discovering meta-paths in large heterogeneous information networks. In: Proceedings of the 2015 International Conference on World Wide Web (WWW), pp. 754–764

Yang C, Liu M, He F, Zhang X, Peng J, Han J (2018) Similarity modeling on heterogeneous networks via automatic path discovery. In: European Conference on Machine Learning and Knowledge Discovery in Databases (ECMLPKDD), pp. 37–54

Raedt LD (2008) Logical and Relational Learning. Cognitive Technologies

Blockeel H, Raedt LD (1998) Top-down induction of first-order logical decision trees. Artif Intell 101(1–2):285–297

Serafino F, Pio G, Ceci M (2018) Ensemble learning for multi-type classification in heterogeneous networks. IEEE Trans Knowl Data Eng 30(12):2326–2339

Petkovic M, Ceci M, Kersting K, Dzeroski S (2020) Estimating the importance of relational features by using gradient boosting. In: International Symposium on Foundations of Intelligent Systems (ISMIS). Lecture Notes in Computer Science, vol. 12117, pp. 362–371

Lavrac N, Skrlj B, Robnik-Sikonja M (2020) Propositionalization and embeddings: two sides of the same coin. Mach Learn 109(7):1465–1507

Bruna J, Zaremba W, Szlam A, LeCun Y (2014) Spectral networks and locally connected networks on graphs. In: Proceedings of the 2014 International Conference on Learning Representations (ICLR)

Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Proceedings of the 2016 Annual Conference on Neural Information Processing Systems (NIPS), pp. 3837–3845

Fan Y, Hou S, Zhang Y, Ye Y, Abdulhayoglu M (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), pp. 253–262

Hu Z, Dong Y, Wang K, Sun Y (2020) Heterogeneous graph transformer. In: Proceedings of the 2020 International Conference on World Wide Web (WWW), pp. 2704–2710

Yun S, Jeong M, Kim R, Kang J, Kim HJ (2019) Graph transformer networks. In: Proceedings of the 2019 Annual Conference on Neural Information Processing Systems (NeurIPS), pp. 11960–11970

Tanon TP, Stepanova D, Razniewski S, Mirza P, Weikum G (2018) Completeness-aware rule learning from knowledge graphs. In: Proceedings of the 2018 International Joint Conferences on Artifical Intelligence (IJCAI), pp. 5339–5343

Ahmadi N, Huynh V, Meduri VV, Ortona S, Papotti P (2020) Mining expressive rules in knowledge graphs. ACM Journal of Data and Information Quality

Sutton RS, Barto AG (1998) Reinforcement learning: An introduction. IEEE Transactions on Neural Networks and Learning Systems 9(5):1054–1054

Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA (2017) Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine

Haykin S (1999) Neural networks: A comprehensive foundation. Knowledge Engineering Review

Hou Y, Liu L, Wei Q, Xu X, Chen C (2017) A novel DDPG method with prioritized experience replay. In: Proceedings of the 2017 International Conference on Systems Man and Cybernetics (SMC), pp. 316–321

Fan J, Wang Z, Xie Y, Yang Z (2020) A theoretical analysis of deep q-learning. In: Proceedings of the 2020 Annual Conference on Learning for Dynamics and Control (L4DC). Proceedings of Machine Learning Research, vol. 120, pp. 486–489

Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: Large-scale information network embedding. In: Proceedings of the 2015 International Conference on World Wide Web (WWW), pp. 1067–1077

Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: Online learning of social representations. In: Proceedings of the 2014 ACM Conference on Knowledge Discovery and Data Mining (KDD), pp. 701–710

Shang J, Qu M, Liu J, Kaplan LM, Han J, Peng J (2016) Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. CoRR **abs/1610.09769**