mg2vec: Learning Relationship-Preserving Heterogeneous Graph Representations via Metagraph Embedding

Wentao Zhang, Yuan Fang, *Member, IEEE,* Zemin Liu,
Min Wu, *Member, IEEE,* and Xinming Zhang, *Senior Member, IEEE*

Abstract—Given that heterogeneous information networks (HIN) encompass nodes and edges belonging to different semantic types, they can model complex data in real-world scenarios. Thus, HIN embedding has received increasing attention, which aims to learn node representations in a low-dimensional space, in order to preserve the structural and semantic information on the HIN. In this regard, metagraphs, which model common and recurring patterns on HINs, emerge as a powerful tool to capture semantic-rich and often latent relationships on HINs. Although metagraphs have been employed to address several specific data mining tasks, they have not been thoroughly explored for the more general HIN embedding. In this paper, we leverage metagraphs to learn relationship-preserving HIN embedding in a self-supervised setting, to support various relationship mining tasks. In particular, we observe that most of the current approaches often under-utilize metagraphs, which are only applied in a pre-processing step and do not actively guide representation learning afterwards. Thus, we propose the novel framework of mg2vec, which learns the embeddings for metagraphs and nodes jointly. That is, metagraphs actively participates in the learning process by mapping themselves to the same embedding space as the nodes do. Moreover, metagraphs guide the learning through both first- and second-order constraints on node embeddings, to model not only latent relationships between a pair of nodes, but also individual preferences of each node. Finally, we conduct extensive experiments on three public datasets. Results show that mg2vec significantly outperforms a suite of state-of-the-art baselines in relationship mining tasks including relationship prediction, search and visualization.

Index Terms—heterogeneous information networks, network embedding, relationship mining.

1 Introduction

Real world objects often interact with each other to form large-scale networks, such as social, biological and transportation networks. Therefore, network analysis becomes an important research area, with many crucial applications such as personalized recommendations [1], disease protein predictions [2] and information retrieval [3], which can often be cast as node classification or link prediction tasks. These tasks ultimately boil down to deriving effective node representations. While traditional approaches rely heavily on manual feature engineering, network embedding [4] has emerged as a promising family of algorithms to learn node representations automatically.

Earlier network embedding algorithms [5], [6], [7] are mostly designed for homogeneous networks, where there is only a single type of node and edge. However, in real-world scenarios, objects are often organized into *Heterogeneous Information Networks* (HIN) [8], where both nodes and edges belong to different types. As illustrated in Figure 1, nodes in the network denote different types of objects such as user, address and company. These objects further interact with (*i.e.*, link to) each other via various explicit relationships: user u_1 lives at address 123 Green St., and u_4 inherits the family name Smith, among other examples. Note that depending on the contexts, networks with multiple types of relationship are also variously called multi-view [9], multiplex [10], multi-dimensional [11] or multi-layered networks [12] in the literature.

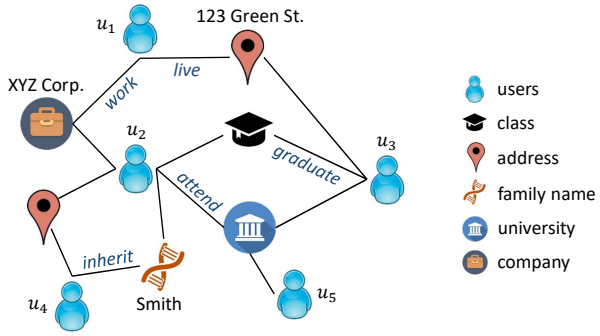
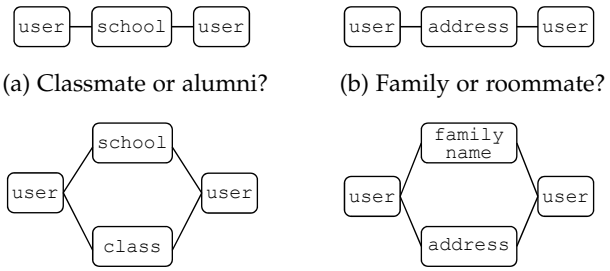


Fig. 1: A toy heterogeneous information network (HIN).

To capture the rich semantics on HINs, recent methods [13], [14], [15], [16] distinguish different types of explicit relationship (*i.e.*, edge) when handling neighboring nodes in the learning process. For instance, while both XYZ Corp. and 123 Green St. are neighboring nodes of u_1 on the toy HIN, they convey distinct semantics via

- *W. Zhang is with the University of Science and Technology of China, Hefei, China, and was a visiting research student with Singapore Management University, Singapore. E-mail: zwt@mail.ustc.edu.cn*
- *Y. Fang (co-corresponding author) and Z. Liu are with Singapore Management University, Singapore. E-mail: {yfang,zmliu}@smu.edu.sg*
- *M. Wu is with the Institute for Infocomm Research, Singapore. E-mail: wumin@i2r.a-star.edu.sg*
- *X. Zhang (co-corresponding author) is with the University of Science and Technology of China, Hefei, China. E-mail: xinming@ustc.edu.cn*

Manuscript received xxx; revised xxx.



(a) Classmate or alumni? (b) Family or roommate?

(c) Likely classmate (d) Likely family

Fig. 2: From metapaths to metagraphs.

work and live relationships, respectively. Thus, it becomes imperative to differentiate them. Moreover, composite and often latent relationships also exist, such as colleagues working together, or peer researchers in the same field, which can be modeled to some extent by *metapath* structures [17] such as user—company—user and author—paper—venue—paper—author, respectively. A number of studies [18], [19], [20] have leveraged such metapaths to learn representations for HINs.

However, metapaths still fall short of expressing more intricate relationships between nodes. Consider the metapath shown in Figure 2(a). The underlying relationship between the two users is unclear, which could be either classmate or fellow almuni. To reduce such ambiguity, *metagraphs* [21], [22] have been proposed, which can express finer-grained semantics. For instance, the metagraph in Figure 2(c) is able to capture the classmate relationship with higher confidence than its metapath cousin in (a). Similarly, the metapath in Figure 2(b) cannot tell apart the family and roommate relationship, whereas the metagraph in (d) can better characterize the family relationship. In particular, given two metapaths, namely, user—family name—user and user—address—user, each of them cannot confidently characterize the family relationship independently. In contrast, their conjunction, which is equivalent to the metagraph in (d), is able to express the family relationship well, implying that metagraphs exhibit a higher expressive power than its constituent metapaths. In fact, metagraphs can be regarded as nonlinear models of metapaths [21], [23].

Problem. Given their expressiveness, we leverage metagraphs to address relationship-preserving HIN embedding in a self-supervised manner. It aims to map the nodes to a low-dimensional vector space that preserves the latent composite relationships between nodes, in addition to explicit heterogeneous relationships. Ultimately, the learned representations can support different downstream relationship mining tasks. For one, we can predict the relationship class between two nodes [24], which can be cast as an instance of classification. Note that this task is different from link prediction—in relationship prediction, we model the latent relationships rather than the direct links between nodes. In particular, two nodes could be already linked, but their specific relationship is still unknown, such as predicting if two linked users are family or classmates on a social network. Another task is to search for and rank nodes involved in a target relationship with a query node [21], [25], which can be cast as an instance of learning to rank [26]. Lastly, relationship visualization can also be performed to identify different latent relationships. All these relationship mining tasks leverage the relationship-preserving embeddings as their input features.

Challenges and insights. Although metagraphs have been utilized in a number of data mining problems including search [21], classification [2] and recommendation [27] with considerable performance gains, their research have been limited in HIN embedding for learning more universal representations. In particular, two major questions remain in effectively exploiting metagraphs.

First, how to take the full advantage of metagraphs in HIN embedding? Most existing approaches utilize metagraphs only in a precomputation step to derive or sample various forms of feature, such as computing node similarities [28], extracting neighborhoods [29], projecting the HIN into multiple homogeneous networks [30], [31], and sampling random walks [32]. In these studies, the metagraphs become irrelevant after the precomputation, since only their derived features, but not metagraphs themselves, contributes to representation learning. This is an "underutilization" of metagraphs—given their ability to express rich semantics, their active and direct participation in learning may better preserve the latent relationships between nodes in the embedding space.

In our approach, we treat metagraphs as first class citizens, by mapping them into the same embedding space as nodes. As such, metagraphs can directly constrain node representations in a relationship-preserving manner. Thus, metagraphs actively contribute to representation learning, instead of just being a passive tool for deriving features during precomputation.

Second, how to effectively employ metagraphs to constrain node representations in the common embedding space? While nodes interact with each other on a network, each node also exists as an individual object. Most previous network or HIN embedding methods have been primarily concerned with exploiting the interactions between two nodes through various constructs, such as the center-context node pairs in skip-grams [5], [18], proximity between two nodes [6], as well as metapaths or other structures connecting two nodes [16], [19], [33]. In general, they capture the interaction between two nodes, but largely neglect the role of individual nodes. In the context of relationship-preserving embedding, node individuality is also crucial, which can be deemed as a preference or prior for the node to interact with other nodes in a certain way.

In our approach, we leverage metagraphs to guide node representation learning through both first- and second-order metagraph embedding. Specifically, in the first-order metagraph embedding, we utilize the set of metagraphs associated with each node to learn the preferences of individual nodes, as different nodes exhibit varying tendency to interact with other nodes via different latent relationships. In the second order, we directly model the interactions between two nodes through the set of metagraphs connecting them. Both orders of metagraph embedding lend to more robust and effective node representations.

Contribution. In this paper, we study the methodology of relationship-preserving HIN embedding. Specifically, we

propose a novel framework called mg2vec by exploiting metagraph embeddings in a self-supervised manner, to support various relationship mining tasks. In summary, we make the following contributions.

- We propose to fully utilize metagraphs for learning relationship-preserving representations on HINs.
- We devise a novel self-supervised framework mg2vec, which actively guide representation learning via first- and second-order metagraph embedding.
- We evaluate mg2vec extensively on three datasets, and achieve promising empirical performance across relationship prediction, search and visualization.

TABLE 1: Summary of main notations.

Notation	Description
G	a heterogeneous information network (HIN)
V, E	the set of nodes and edges on G, resp.
$\mathcal{M}$	the set of metagraphs on G
$M_i \in \mathcal{M}$	a metagraph on G
$\mathcal{S}_i$	the set of subgraphs instantiated by metagraph M_i
$\mathcal{S}^{(v)}$	the set of subgraphs containing node v
$\mathcal{S}^{(u,v)}$	the set of subgraphs containing both nodes u and v
$\mathbf{m}_i \in \mathbb{R}^D$	the vector representation of metagraph M_i
$\mathbf{v} \in \mathbb{R}^D$	the vector representation of node v
$\alpha \in [0, 1]$	trade-off between the first and second order

2 Related Work

Network embedding [4], [34] has been extensively studied in recent years, due to its impressive ability in deriving effective network representations without manual feature engineering. A significant number of approaches deal with homogeneous graphs, mainly to preserve network structures in the embedding space such as DeepWalk [5] and node2vec [7], and to preserve higher-order proximities such as LINE [6], SDNE [35] and HOPE [36]. Generative adversarial networks [37], [38] have also been employed for better sampling, such as GraphGAN [37] and ANE [38]. Meanwhile, graph neural networks such as GCN [39] and GAT [40] have emerged as competitive approaches, but they often address a specific task in a supervised or semi-supervised manner, which depart from our self-supervised setting to support different relationship mining tasks. Nonetheless, GraphSAGE [41] also presents a self-supervised variant.

Beyond homogeneous graphs, HIN embedding [8] has also attracted considerable attention due to their prevalence in real-world applications. On the one hand, explicit relationships on HINs are differentiated in various models, such as PTE [42], HEP [13], PME [14], HEER [15] and HeGAN [16]. On the other hand, to model latent relationships between nodes, an important tool is metapath [17], which is a path pattern joining multiple explicit relationships. Due to their composite nature, metapaths are able to capture long-range latent relationships, and have been employed in several HIN embedding approaches including metapath2vec [18], hin2vec [19] and HERec [20]. However, metapaths are still inadequate to capture more intricate relationships.

Thus, the more powerful metagraphs [21], [22], which are nonlinear functions of metapaths [21], [23], have been proposed for several data mining tasks [2], [27], although their adoption in the more general HIN embedding has been limited. Fang *et al.* [23] directly utilize metagraphs as the representations for HINs, where each metagraph constitutes one dimension and its number of instances containing a node form the feature value for that node. While such metagraph-based representations are reasonable, they could still be high dimensional and sparse on a type-rich HIN with a large number of metagraphs. Other metagraph-based methods often adopt an embedding approach to learn low-dimensional representations. However, most of these studies [28], [29], [30], [31], [32] only utilize metagraphs in a precomputation step, without fully exploiting metagraphs to actively capture and preserve latent relationships during representation learning. A more recent approach M-HIN [33] jointly learns metagraph and node embeddings based on complex tensor factorization [43], which is tantamount to modeling the second-order interactions between two nodes but not the first-order preferences of individual nodes.

On another line, relationship mining has been extensively researched in text mining and natural language processing, ranging from the early work using handcrafted patterns [44] or automatically mined patterns [45], to recent deep neural network models [46], [47]. However, these studies focus on mining relationships from texts, whereas we deal with HINs. There is also a series of work on knowledge graph embedding including TransE [48] and TransR [49], but their primary goal is to preserve the subject-predicate-object triples (*i.e.*, explicit heterogeneous relationships), instead of latent composite relationships. Thus, they are not adequately designed for HIN embedding, and achieve inferior empirical performance compared to our model. Lastly, proximity embedding approaches [25], [50] also exist on HINs, although they focus on learning only one target relationship class in a supervised manner. In contrast, we aim to learn relationship-preserving embeddings that are able to capture different latent relationships in a self-supervised setting.

3 Preliminaries

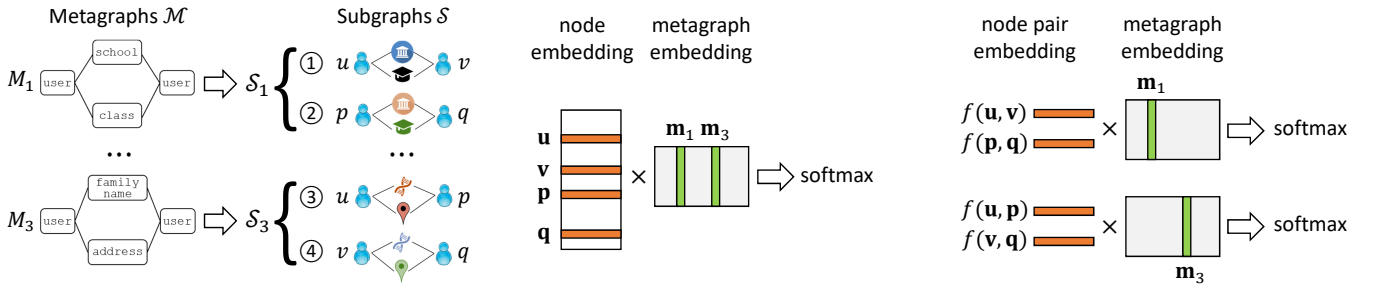
In this section, we introduce the definitions of HINs and metagraphs, and formalize the problem of relationship-preserving HIN embedding. Main notations are summarized in Table 1.

3.1 Definitions

An example of HIN is illustrated in Figure 1 on a toy social network. Observe that it consists of multiple node types (*e.g.*, user, company or address) and edge types (*e.g.*, work or live).

Definition 1 (Heterogeneous information network [8])**.** A *heterogeneous information network* or *HIN* is a graph $G = (V, E, T, R, \phi, \varphi)$, where V and E denote the sets of nodes and edges, respectively. It is also associated with a node type mapping function $\phi : V \to T$ and an edge type mapping function $\varphi : E \to R$, where T and R denote the sets of node and edge types such that $|T| > 1$ or $|R| > 1$. □

Metagraphs have emerged as a valuable tool to capture rich semantics on HINs. As Figure 1 shows, nodes often



(a) Metagraph and subgraph enumeration (b) First-order metagraph embedding (c) Second-order metagraph embedding

Fig. 3: Overall framework of mg2vec. (a) Enumeration of metagraphs and their subgraph instances from the input HIN. (b) First-order metagraph embedding to constrain the representations of individual nodes. (c) Second-order metagraph embedding to constrain the representations of node pairs.

interact with each other in common patterns that can be summarized by metagraph structures shown in Figure 2.

Definition 2 (Metagraph [21])**.** A *metagraph* can be denoted as a graph $M = (V_M, E_M)$, where V_M is the set of nodes in the metagraph M, and E_M is the set of edges in M. Moreover, each node $v \in V_M$ denotes a type from the node types T and each edge $(u, v) \in E_M$ denotes a type from the edge types R. □

Note that a node on the HIN typically represents a physical object with both an intrinsic identity (*e.g.*, XYZ Corp.) and a type (*e.g.*, company), whereas a node on the metagraph is only a type descriptor without correspondence to any concrete object. In particular, each metagraph instantiates a set of subgraphs on the HIN, and these subgraphs contain various physical objects. For example, the subgraph u_1—XYZ Corp.—u_2 is an instance of the metagraph user—company—user in Figure 1. The instances of a metagraph are formalized below.

Definition 3 (Metagraph instance [21])**.** Consider a subgraph S with nodes $V_S \subset V$ and edges $E_S \subset E$, and a metagraph $M = (V_M, E_M)$. S is an *instance* of M if there exists a bijection between their nodes, $\psi : V_S \to V_M$, such that the following hold:

- $\forall v \in V_S, \phi(v) = \psi(v)$; and
- $\forall u, v \in V_S, \langle u, v\rangle \in E_S$ iff $\langle \psi(u), \psi(v)\rangle \in E_M$, and $\varphi(\langle u, v\rangle) = \langle \psi(u), \psi(v)\rangle$. □

Note that a subgraph can only be instantiated by one metagraph, although a metagraph can instantiate many different subgraphs on a HIN.

3.2 Relationship-Preserving HIN Embedding

Using metagraphs and their instances, we aim to capture the latent relationships between a set of *core nodes* $C \subseteq V$, whose relationships with each other are the target of investigation. For instance, in Figure 1, if we are interested in the latent relationships between users, the set of all user nodes form the core nodes. In the most general case, all nodes on the HIN can be core nodes.

The problem of relationship-preserving HIN embedding is thus to learn a mapping that projects each core node $v \in C$ to a low-dimensional space $\mathbb{R}^D$, where D is the number of dimensions. Node representations in the new space should ideally preserve the semantic relationships between the core nodes. In our proposal, we further aim to map metagraphs into the same space, to actively constrain node representations. In particular, we assume a self-supervised setting, so that the learned node embeddings can be utilized for various relationship mining tasks including relationship prediction, search and visualization.

4 Proposed Approach: mg2vec

In this section, we introduce our framework mg2vec, which hinges on metagraphs to learn relationship-preserving HIN embedding. An overview of the framework is presented in Figure 3. First, we enumerate the set of metagraphs and their subgraph instances. Next, based on these metagraphs, we resort to both first- and second-order metagraph embedding, which treat metagraphs as first class citizens by mapping them into a common embedding space as the nodes reside. Finally, we present the overall model.

4.1 Enumeration of Metagraphs and Instances

In the initial step shown in Figure 3(a), we enumerate the set of metagraphs $\mathcal{M}$ on the HIN, and for each metagraph $M_i \in \mathcal{M}$ we enumerate its set of subgraph instances $\mathcal{S}_i$. While the subgraph instances depict the interactions between the objects, metagraphs summarize them into common patterns. For instance, the four subgraphs involving the nodes u, v, p, q can be abstracted by two metagraphs M_1 and M_3, where the subgraphs in $\mathcal{S}_1$ are instantiated by metagraph M_1, and the subgraphs in $\mathcal{S}_3$ are instantiated by M_3. Each subgraph can only be instantiated by one and only one metagraph.

There is extensive literature [51], [52], [53], [54] on the efficient enumeration of metagraphs and their instances up to a given size. The problem can be reduced to the well known NP-hard subgraph isomorphism. A brute-force approach incurs a time complexity of $O(|V|d^{|V_M|-1})$, where $|V|$ is the number of nodes in the HIN G, d is the average degree of G, and $|V_M|$ is the number of nodes in the metagraph. Nevertheless, the complexity can be significantly reduced through various pruning techniques and special data structures. For example, Bi *et al.* [53] proposed an auxiliary data structure called compact path-index (CPI) with a polynomial construction time of $O(|E||E_M|)$ where E is the number of edges in G and E_M is the number of

edges in the metagraph. CPI is able to drastically decrease the enumeration time to the point that CPI construction often dominates the total processing time.

4.2 First-order Metagraph Embedding

We guide the embedding of each core node to express their individual preference, as shown in Figure 3(b). Specifically, a core node can be found in many subgraphs, and their instantiating metagraphs further characterize the latent relationships the node tends to participate in. For ease of discussion, we only consider symmetric metagraphs [21] containing only two core nodes here, such as those shown in Figure 2. We will discuss how to handle other metagraphs in Section 4.5.

Let $\mathbf{m}_i \in \mathbb{R}^D$ and $\mathbf{v} \in \mathbb{R}^D$ denote the embedding of metagraph M_i and node v, respectively, which are embedded into the same space. Let $\mathcal{S}^{(v)}$ denote the set of subgraphs containing node v. For example, in Figure 3(a) with four subgraphs labeled ①②③④, $\mathcal{S}^{(u)} = \{①③\}$, $\mathcal{S}^{(v)} = \{①④\}$, *etc.* Note that each of the four nodes u, v, p and q appears in at least one instance of the metagraph M_1 and one instance of M_3. Therefore, both M_1 and M_3 characterize the preference of each node. It further implies that their embeddings $\mathbf{m}_1$ and $\mathbf{m}_3$ will constrain node embeddings $\mathbf{u}, \mathbf{v}, \mathbf{p}$ and $\mathbf{q}$ individually, as illustrated in Figure 3(b).

Specifically, for a core node v, we adopt the self-supervised goal of predicting its containing subgraphs $\mathcal{S}^{(v)}$. More formally, we maximize $P(\mathcal{S}^{(v)}|v;\Theta)$, where Θ are model parameters consisting of all node and metagraph embeddings, *i.e.*, $\Theta = \{\mathbf{v} : v \in V, \mathbf{m}_i : M_i \in \mathcal{M}\}$. Since the set of subgraphs containing v can be instantiated by many different metagraphs, $\mathcal{S}^{(v)}$ can be broken down into several disjoint subsets,

$$\mathcal{S}^{(v)} = \bigcup_{M_i \in \mathcal{M}} \{\mathcal{S}^{(v)} \cap \mathcal{S}_i\}, \tag{1}$$

where each subset $\{\mathcal{S}^{(v)} \cap \mathcal{S}_i\}$ contains the subgraphs instantiated by a common metagraph M_i. These subsets are disjoint since each subgraph can only be instantiated by exactly one metagraph.

Treating all subgraphs of v instantiated by a common metagraph as sharing the same underlying distribution, we obtain the following.

$$\begin{aligned} P(\mathcal{S}^{(v)}|v;\Theta) &= \prod_{M_i \in \mathcal{M}} P(\mathcal{S}^{(v)} \cap \mathcal{S}_i|v;\Theta) \\ &= \prod_{M_i \in \mathcal{M}} P(M_i|v;\Theta)^{|\mathcal{S}^{(v)} \cap \mathcal{S}_i|}, \end{aligned} \tag{2}$$

where $P(M_i|v;\Theta)$ can be materialized using a softmax function:

$$P(M_i|v;\Theta) = \frac{\exp(\mathbf{m}_i \cdot \mathbf{v})}{\sum_{M_j \in \mathcal{M}} \exp(\mathbf{m}_j \cdot \mathbf{v})}. \tag{3}$$

Accounting for all core nodes, we minimize the following negative log-likelihood to achieve the first-order metagraph embedding.

$$L_1 = -\sum_{v \in C} \sum_{M_i \in \mathcal{M}} |\mathcal{S}^{(v)} \cap \mathcal{S}_i| \log P(M_i|v;\Theta). \tag{4}$$

4.3 Second-order Metagraph Embedding

Next, we guide the embedding of each pair of core nodes to express their latent relationships, as shown in Figure 3(c). More specifically, a pair of core nodes can co-appear in many subgraphs, and these subgraphs can be characterized by different metagraphs to capture latent relationships between the two nodes.

Let $\mathcal{S}^{(u,v)}$ denote the set of subgraphs containing both nodes u and v simultaneously. It follows that, in Figure 3(a), $\mathcal{S}^{(u,v)} = \{①\}$, $\mathcal{S}^{(p,q)} = \{②\}$, *etc.* Therefore, as illustrated in Figure 3(c), $\mathbf{m}_1$ will constrain $f(\mathbf{u}, \mathbf{v})$ and $f(\mathbf{p}, \mathbf{q})$, and $\mathbf{m}_3$ will constrain $f(\mathbf{u}, \mathbf{p})$ and $f(\mathbf{v}, \mathbf{q})$, given some vector-valued function f to aggregate the representations of two nodes. Our goal is to design $f : \mathbb{R}^{2D} \to \mathbb{R}^D$ to map the aggregation of two nodes to the same space as metagraphs, such that the containing metagraphs of the two nodes could constrain their representations. Specifically, we adopt the following formulation of f:

$$f(\mathbf{u}, \mathbf{v}) = \text{ReLU}([\mathbf{u} \| \mathbf{v}]W + \mathbf{b}), \tag{5}$$

where $\|$ is the concatenation operator, $W \in \mathbb{R}^{2D \times D}$ is a weight matrix, $\mathbf{b} \in \mathbb{R}^D$ is a bias vector, and ReLU is the activation function.

Similar to the first order, for a pair of core nodes u and v, we adopt the self-supervised goal of predicting their containing subgraphs $\mathcal{S}^{(u,v)}$, by maximizing $P(\mathcal{S}^{(u,v)}|u, v;\Theta)$. We also treat the containing subgraphs instantiated by the same metagraph as sharing the same underlying distribution. Specifically,

$$\begin{aligned} P(\mathcal{S}^{(u,v)}|u, v;\Theta) &= \prod_{M_i \in \mathcal{M}} P(\mathcal{S}^{(u,v)} \cap \mathcal{S}_i|u, v;\Theta) \\ &= \prod_{M_i \in \mathcal{M}} P(M_i|u, v;\Theta)^{|\mathcal{S}^{(u,v)} \cap \mathcal{S}_i|}, \end{aligned} \tag{6}$$

where $P(M_i|u, v;\Theta)$ is also defined with a softmax:

$$P(M_i|u, v;\Theta) = \frac{\exp(\mathbf{m}_i \cdot f(\mathbf{u}, \mathbf{v}))}{\sum_{M_j \in \mathcal{M}} \exp(\mathbf{m}_j \cdot f(\mathbf{u}, \mathbf{v}))}. \tag{7}$$

Accounting for all pairs of core nodes, we minimize the following negative log-likelihood to achieve the second-order metagraph embedding.

$$L_2 = -\sum_{u \in C} \sum_{v \in C \setminus \{u\}} \sum_{M_i \in \mathcal{M}} |\mathcal{S}^{(u,v)} \cap \mathcal{S}_i| \log P(M_i|u, v;\Theta). \tag{8}$$

Note that with the second-order embedding, the model parameters Θ are expanded to include the weight matrix W and bias vector $\mathbf{b}$ employed in the function f.

4.4 Overall Model

To integrate both orders of metagraph embedding, we optimize the overall probability of predicting the containing subgraphs given each core node, as well as given each pair of core nodes. Equivalently, we minimize the following overall loss:

$$L = (1 - \alpha)L_1 + \alpha L_2, \tag{9}$$

where $\alpha \in [0, 1]$ is a hyperparameter to control the balance between the two orders.

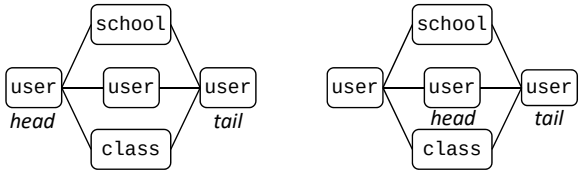
We adopt two common forms of sampling to efficiently solve the above optimization. First, we apply negative sampling [7] for the softmax functions in Eq. (3) and (7) to speed up their computation. Specifically, among all metagraphs $\mathcal{M}$, we randomly sample K of them as the negative samples, where K is typically a small constant. Second, for the second-order loss in Eq. (8), we apply random walks [5] to sample the node pairs under consideration. Specifically, starting from each core node, we perform η random walks on the HIN, such that each random walk traverses λ core nodes. We further extract node pairs with skip-grams using a window of size ω. On the one hand, node pairs not sampled by the skip-gram model are presumably far away from each other on the HIN, which are unlikely to be related in significant ways. Thus, discarding them would have an negligible impact on the performance. On the other hand, this sampling step drastically reduces the number of pairs from $O(|V|^2)$ to $O(\eta\lambda\omega|V|)$, since η, λ and ω are constants much smaller than $|V|$. Note that since the number of core nodes $|C|$ can be as large as all nodes $|V|$, we use the upper bound $|V|$ for a more general analysis.

We further conduct a complexity analysis. Updating the first-order loss incurs $O(|V||\mathcal{M}|KD)$ time, where K is the negative sampling size and D is the number of embedding dimensions. Moreover, updating the second-order loss incurs $O(\eta\lambda\omega|V||\mathcal{M}|KD)$ time, since there are $O(\eta\lambda\omega|V|)$ node pairs as analyzed earlier. Thus, the second-order loss dominates the overall complexity, which is linear in the graph size $|V|$ and the number of metagraphs $|\mathcal{M}|$.

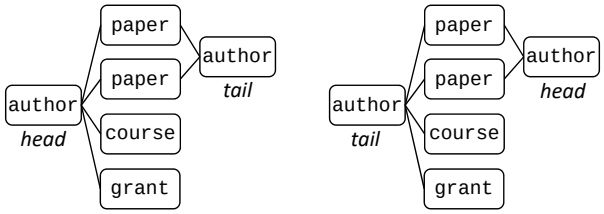
4.5 Extension to general metagraphs

So far we have assumed a symmetric metagraph with exactly two core nodes. However, a metagraph could contain more than two core nodes. For instance, given users as the core nodes on a social network, involving additional users could capture friends-of-friends interactions, as illustrated in Figure 4(a). The pair of users labeled head and tail on the left clearly entails different latent relationships from the pair labeled on the right, even though all three users appear in the same metagraph. To address this issue, we leverage the notion of *anchored metagraphs* [23], which augment the definition of metagraphs to include head and tail anchors. As such, the two anchored metagraphs in Figure 4(a) are considered different as they capture different semantics between the two anchors. Similarly, their subgraph instances would also be associated with similar anchors to be treated as distinct subgraphs. Subsequently, when calculating the first- and second-order losses in Eq. (4) and (8), the core nodes must correspond to the anchors in the subgraphs and metagraphs, instead of merely appearing in them.

Sometimes two core nodes may also play asymmetric roles, such as being an advisor or an advisee in academic graphs. As shown in Figure 4(b), the left author node could be the advisor and the right author node could be the advisee. To distinguish the direction of their relationship, we can also resort to head and tail anchors—the left anchored metagraph captures the advisor-advisee relationship, whereas the right one captures the advisee-advisor relationship.



(a) Metagraphs with more than two core nodes



(b) Metagraphs with asymmetric core nodes

Fig. 4: Special metagraphs with head and tail anchors.

5 Experiments

In this section, we empirically demonstrate the superior performance of mg2vec, and conduct an in-depth model analysis to understand its underlying mechanism.

5.1 Experimental Setup

5.1.1 Datasets

We conduct extensive experiments on three public datasets, namely, LinkedIn [55], AS [56] and DBLP [24]. We elaborate them below and summarize them in Table 2.

LinkedIn [55]: The dataset can be organized into a HIN with four types of nodes: user, employer, location and college. User nodes can link to the other types of node, and there also exist user—user edges. Through a user study, users labeled their primary relationship with their friends on LinkedIn, including *School*, *Work* and *Other* relationships. These relationships are latent as they cannot be directly observed on the HIN. For example, sharing the same employer does not automatically and necessarily make two users primarily related via *Work*. A common scenario is that they may view themselves as personal friends, who happen to be in different departments of the same large company without much work-related interaction. On this dataset, the core nodes consist of user nodes as we are interested in the relationships between users.

AS [56]: A HIN dataset for interconnecting Autonomous Systems (AS) on the Internet. On this graph, each node represents an AS and each edge represents the interaction between two ASes. There are three types of nodes: top AS nodes who are not customers of any other AS, bottom AS nodes who are not providers of any other AS, and middle AS nodes who are both customers and providers of some other ASes. Each interaction (*i.e.*, observable link) between two ASes belongs to one of the four relationships, namely, *Customer*, *Supplier*, *Peer* (ASes not involved in hierarchical relationships) and *Sibling* (ASes owned by the same organization). While *Customer* and *Supplier* are asymmetric relationships, *Peer* and *Sibling* are symmetric. These relationships are latent as they cannot be directly observed on the HIN during representation learning. We treat middle ISP

TABLE 2: Summary of datasets. $|V|$: number of nodes, $|E|$: number of edges, $|T|$: number of node types, $|R|$: number of edge types, $|\mathcal{M}|$: number of metagraphs.

	$\|V\|$	$\|E\|$	$\|T\|$	$\|R\|$	$\|\mathcal{M}\|$	# Labeled pairs
LinkedIn	65,925	220,812	4	4	318	5,959
AS	26,475	53,381	3	6	688	19,936
DBLP	172,136	968,822	5	4	88	5,388

nodes as the core nodes, as they can participate in all four relationships with each other.

DBLP [17]: This is an academic graph with five types of nodes: paper, author, year, venue and keyword. Paper nodes can link to nodes of the other types. Author nodes are the core nodes, as they form several latent relationships, including the symmetric *Colleague*, and the asymmetric *Advisor* and *Advisee*, which cannot be observed on the HIN during representation learning. Specifically, the relationships were labeled based on faculty homepages and the Mathematics Genealogy and AI Genealogy projects [24].

5.1.2 Enumeration of metagraphs and instances

Many algorithms exist to efficiently enumerate the metagraphs and their instances up to a given size from an input HIN. We first applied GRAMI [51] to find the set of metagraphs $\mathcal{M}$, and adopted SymISO [21] to compute the set of instances of each metagraph $M_i \in \mathcal{M}$.

Following previous work [21], [23], we applied several filters to prune less useful metagraphs. First, we only used metagraphs containing at least two core nodes in order to capture their interactions. Second, a metagraph must have at least two different node or edge types in order to capture the heterogeneity. Third, we removed metagraphs with "dangling" nodes, which are non-core nodes with degree one, as such nodes often do not explain the interactions between core nodes. Lastly, we restricted metagraphs to have at most 5 nodes on LinkedIn and AS and 6 nodes on DBLP, which are adequate in achieving good empirical performance on downstream tasks in line with previous studies [22], [23]. Note that DBLP has a relatively simple network schema, resulting in only 17 metagraphs of up to size 5. Thus, we chose size 6 on DBLP, which generates a reasonable variety of 88 metagraphs. A more detailed analysis on metagraph size will be conducted in Section 5.3.1. These rules are generally domain independent and do not require prior knowledge of the dataset. The number of metagraphs after pruning is reported in Table 2.

5.1.3 Baselines and parameters

We compare the proposed mg2vec model with the following state-of-the-art baselines, which belong to four broad categories: (i) homogeneous network methods DeepWalk and LINE; (ii) graph neural network or knowledge graph models GraphSAGE and TransR; (iii) non-metagraph-based HIN methods metapath2vec, hin2vec and HeGAN; and (iv) metagraph-based HIN methods metagraph2vec, MG+ and M-HIN.

- DeepWalk [5]: a pioneering skip-gram model based on random walk sampling for homogeneous graphs.
- LINE [6]: a homogeneous network embedding approach to preserve both first- and second-order proximity.
- GraphSAGE [41]: a graph neural network, using the one-hot encoding of node types as the features of the nodes. Thus, it accounts for the heterogeneity to some extent. We adopted its self-supervised variant to match our setting.
- TransR [49]: a knowledge graph embedding method, which is primarily designed to preserve explicit subject-predicate-object triples rather than latent composite relationships.
- metapath2vec [18]: a HIN embedding method, which employs pre-selected metapaths to sample random walks. For each dataset, we tuned the configuration of metapaths in order to achieve the optimal performance, for fair comparison with our method. Details of the configuration are reported in Appendix A.
- hin2vec [19]: a HIH embedding method, which samples random walks based on metapaths up to a given size, and feed them into a neural network.
- HeGAN [16]: a HIN embedding method, which employs the adversarial principle to generate better negative samples, and model the explicit heterogeneous relationships on HINs.
- metagraph2vec [32]: a HIN embedding method and a variant of metapath2vec, which employs metagraphs to sample random walks.
- MG+ [23]: a feature engineering method that derives node and edge representations based on the number of metagraph instances on a HIN, where each dimension corresponds to a metagraph.
- M-HIN [33]: a HIN embedding method that employs complex tensor factorization [43] to jointly embed metagraphs and nodes, which is tantamount to modeling the second-order interactions only.

For mg2vec, to sample the node pairs, we conducted 10 random walks per node with a walk length 100 and window size 5. We further set its negative sampling size to 10 and embedding size to 128. To achieve a balance between the first and second order, we set α to 0.5 and we will also study its impact on the model in Section 5.4.1. For the baselines, to ensure that they are well tuned, we performed grid search on the main hyper-parameters of each baseline. In most cases the optimal parameters found are consistent with existing literature. The details of the grid search process are reported in Appendix B.

5.1.4 Tasks and evaluation

To extensively evaluate the performance of learned representations, we conduct three relationship mining tasks, including relationship prediction, search and visualization. We elaborate each task in the following.

Prediction. Each dataset contains labeled relationship classes between core node pairs, as introduced in Section 5.1.1. These relationship labels are unknown to the representation learning process, and are only used in the downstream classification. Specifically, labeled pairs were divided into 50% training and 50% testing, and such splitting was repeated 10 times. For all methods, we used the

TABLE 3: Relationship prediction. Each reported figure is the average performance on ten splits, followed by their standard deviation. The best result (in bold) is further marked with * if it is significantly different from the runner-up (underlined) under the two-tail paired t-test at the 0.01 level.

	LinkedIn			AS			DBLP		
	Micro-F	Macro-F	Accuracy	Micro-F	Macro-F	Accuracy	Micro-F	Macro-F	Accuracy
DeepWalk	46.10 ± 1.22	54.89 ± 0.84	60.21 ± 0.53	79.18 ± 0.39	65.37 ± 0.51	77.99 ± 0.44	74.15 ± 1.24	70.74 ± 1.06	77.22 ± 0.87
LINE	21.15 ± 0.53	34.41 ± 0.86	48.27 ± 0.34	65.33 ± 0.59	55.97 ± 1.27	64.35 ± 0.41	55.76 ± 1.44	47.48 ± 0.94	59.65 ± 0.73
GraphSAGE	12.32 ± 0.25	29.25 ± 0.33	46.95 ± 0.21	80.57 ± 0.75	66.43 ± 0.23	81.74 ± 1.15	78.77 ± 0.92	73.62 ± 1.31	81.99 ± 0.45
TransR	42.95 ± 1.41	51.73 ± 1.15	57.36 ± 0.82	84.82 ± 0.86	64.11 ± 0.45	85.73 ± 0.34	84.93 ± 1.09	82.31 ± 1.21	86.94 ± 1.34
metapath2vec	39.36 ± 0.70	50.23 ± 0.63	57.82 ± 0.26	82.96 ± 0.35	66.24 ± 0.41	83.41 ± 0.46	78.62 ± 0.23	74.75 ± 0.74	81.39 ± 0.72
hin2vec	42.15 ± 0.15	50.37 ± 0.23	56.49 ± 0.17	79.04 ± 0.57	63.89 ± 0.24	79.42 ± 0.33	86.13 ± 0.47	85.22 ± 0.35	88.78 ± 0.66
HeGAN	44.76 ± 0.54	53.97 ± 0.43	58.91 ± 0.56	85.23 ± 0.33	66.72 ± 0.12	86.13 ± 0.22	83.90 ± 0.66	81.43 ± 0.19	86.52 ± 0.33
metagraph2vec	44.21 ± 0.53	52.62 ± 1.14	56.49 ± 0.59	84.56 ± 0.25	65.87 ± 0.26	82.57 ± 0.12	83.13 ± 0.41	82.07 ± 0.47	84.11 ± 0.34
MG+	40.19 ± 0.67	48.69 ± 0.91	54.73 ± 1.06	84.67 ± 0.47	64.52 ± 0.67	85.46 ± 0.53	86.06 ± 1.26	85.73 ± 0.97	87.39 ± 1.23
M-HIN	27.98 ± 0.52	39.20 ± 0.43	50.59 ± 0.68	84.45 ± 0.71	64.31 ± 0.86	84.32 ± 0.59	77.12 ± 0.92	73.41 ± 0.37	80.12 ± 1.26
mg2vec	**50.01*± 0.52**	**57.56*± 0.16**	**61.24*± 0.26**	**87.82*± 0.35**	**67.51*± 0.18**	**88.92*± 1.07**	**88.22*± 0.94**	**87.38*± 0.54**	**89.90*± 0.29**

TABLE 4: Relationship search. Each reported figure is the average performance on ten splits, followed by their standard deviation. The best result (in bold) is further marked with * if it is significantly different from the runner-up (underlined) under the two-tail paired t-test at the 0.01 level.

	LinkedIn			AS			DBLP		
	NDCG	MAP	MRR	NDCG	MAP	MRR	NDCG	MAP	MRR
DeepWalk	59.97 ± 1.42	44.79 ± 1.07	56.27 ± 1.06	89.67 ± 0.51	81.63 ± 0.99	87.05 ± 0.73	98.11 ± 0.52	91.99 ± 0.46	92.06 ± 0.59
LINE	55.57 ± 0.74	40.95 ± 1.95	55.61 ± 1.30	85.30 ± 0.27	79.26 ± 0.11	85.12 ± 0.57	95.31 ± 0.44	76.49 ± 1.29	76.53 ± 1.21
GraphSAGE	58.96 ± 0.23	42.56 ± 0.45	58.07 ± 0.30	86.18 ± 0.71	73.19 ± 0.24	78.23 ± 0.13	99.49 ± 0.08	95.11 ± 0.06	95.12 ± 0.13
TransR	58.14 ± 0.71	42.31 ± 0.47	54.27 ± 0.53	92.18 ± 0.23	85.70 ± 0.06	89.71 ± 0.09	98.82 ± 0.24	97.21 ± 0.25	97.22 ± 0.45
metapath2vec	60.41 ± 0.11	44.47 ± 0.16	57.52 ± 0.18	91.91 ± 0.19	84.33 ± 0.12	89.20 ± 0.24	98.51 ± 0.11	93.87 ± 0.13	93.69 ± 0.17
hin2vec	60.85 ± 0.33	44.64 ± 0.20	55.86 ± 0.67	88.88 ± 0.58	79.89 ± 0.81	86.63 ± 0.13	99.29 ± 0.11	95.02 ± 0.13	95.02 ± 0.10
HeGAN	59.84 ± 0.62	44.66 ± 0.33	56.34 ± 0.49	90.94 ± 1.21	82.20 ± 0.86	87.29 ± 0.67	99.79 ± 0.06	97.10 ± 0.36	97.13 ± 0.42
metagraph2vec	58.22 ± 0.44	43.71 ± 0.25	56.69 ± 0.21	90.35 ± 0.33	81.97 ± 0.19	86.53 ± 0.53	98.23 ± 0.41	97.03 ± 0.24	96.91 ± 0.35
MG+	67.88 ± 0.56	53.31 ± 0.69	65.92 ± 0.68	90.27 ± 0.39	80.72 ± 0.34	86.92 ± 0.31	98.42 ± 0.27	93.41 ± 0.73	93.46 ± 0.48
M-HIN	58.46 ± 1.51	42.72 ± 0.84	58.09 ± 0.80	89.77 ± 0.86	79.94 ± 1.28	85.07 ± 0.67	98.42 ± 0.33	91.86 ± 0.24	91.81 ± 0.49
mg2vec	**71.16*± 0.23**	**57.39*± 0.29**	**70.49*± 0.51**	**93.12*± 0.43**	**86.22*± 0.26**	**90.07*± 0.22**	**99.87 ± 0.11**	**97.90 ± 0.23**	**97.92 ± 0.17**

concatenation of the two nodes' embeddings as the final feature vector for each node pair, and trained an SVM classifier. The hyper-parameters of SVM were selected using a five-fold cross validation on the training data, with a grid search over $C \in \{0.01, 1, 100\}$ and $\gamma \in \{0.0001, 0.001, 0.01\}$. Note that the Hadamard product is a popular alternative to combine two nodes' embeddings, but it cannot distinguish asymmetric classes such as *Advisor* and *Advisee*. We averaged the performance on the 10 test sets in terms of Micro-F, Macro-F and accuracy.

Search. We conduct a ranking-based relationship search, where a query node is associated with both positive and negative candidates of the target relationship. We considered the target relationships *Work*, *Peer* and *Advisor* on LinkedIn, AS and DBLP, respectively. Positive candidates are related to the query via the target relationship, and the negative candidates consist of nodes related to the query via other relationships. We split all queries into 50% training and 50% testing, and repeated such splitting 10 times. For all methods, we adopted a learning to rank model [21], and averaged the performance on the 10 test sets in terms of NDCG, MAP and MRR at top 10 results.

Visualization. To intuitively showcase the embedding quality, we used the t-SNE algorithm for visualization. Specifically, we present the visualization in two ways: (i) visualization of the embeddings of node pairs that belong to different relationships, by concatenating the two nodes' embeddings as the input to the t-SNE algorithm; (ii) visualization of the metagraph embeddings to show their ability to preserve latent relationships.

5.2 Performance Evaluation

We empirically evaluate the proposed mg2vec and the baselines on the three relationship mining tasks.

5.2.1 Prediction

In Table 3, we report the results on relationship prediction. In summary, mg2vec consistently outperforms all the baselines with statistical significance. We discuss further observations in the following.

First, mg2vec is better than homogeneous network embedding algorithms such as DeepWalk and LINE. These baselines treat all nodes and edges as one uniform type, and

thus fall short of capturing and differentiating rich semantics on HINs. Note that on LinkedIn, DeepWalk performs surprisingly well in comparison to other HIN embedding methods such as metapath2vec. This observation can be attributed to a special characteristic of LinkedIn—being a professional network, each user is more exclusively connected to friends from the same college or employer. That means, if a user is currently in school (or working), it forms the *School* (or *Work*) relationship with majority of its direct neighbors. Specifically, in our ground truth, 67.7% or more than two thirds of the users have a very skewed distribution of neighbors, such that more than 80% of their neighbors belong to a one single relationship. This particularly benefits DeepWalk as it samples random walks starting from each user, which is guaranteed to form node pairs with its direct neighbors. The majority of these node pairs formed out of direct neighbors would already belong to the same relationship, even without considering the heterogeneity. Thus, similar embeddings would be learned for these node pairs, which makes the downstream task easier. On the other hand, while metapath2vec also applies random walk, it is forced to follow metapaths which skip over direct neighbor pairs. Nevertheless, our method mg2vec still performs the best on LinkedIn, showing its robustness and universality.

Second, mg2vec also outperforms the graph neural network model GraphSAGE and knowledge graph model TransR. GraphSAGE only utilizes the heterogeneity in a limited manner by encoding node types as input features, whereas TransR was originally proposed for knowledge graphs consisting of explicit subject-predicate-object triples, which might not be adaptive to capturing latent composite relationships on HINs. Thus, their performances are generally suboptimal.

Third, compared to baselines specifically designed for HIN embedding, mg2vec still remains advantageous. Some of these baselines (metapath2vec, hin2vec and HeGAN) do not utilize metagraph, which turn out to be inadequate for more complex relationships. More importantly, mg2vec is also superior to other metagraph-based methods including metagraph2vec, MG+ and M-HIN. While metagraph2vec achieves generally better performance than its cousin metapath2vec, it is still behind mg2vec by a large margin, since it only employs metagraphs to sample random walks in a precomputation step. Furthermore, MG+ is not an embedding method, which directly treats every metagraph as one dimension of the representations. Thus, its derived representations can still be high dimensional and sparse when there are a large number of metagraphs. On the other hand, M-HIN considers the joint embedding of metagraphs and nodes based on the framework of complex-valued embeddings, but it does not reconcile the first-order preferences of individual nodes in the same framework.

5.2.2 Search

We report the relationship search results in Table 4. The overall observation is similar to relationship prediction, with mg2vec being consistently better than all the baselines. More specifically, mg2vec outperforms the baselines in all four categories, for the same reasons discussed in relationship prediction.

5.2.3 Visualization

First, we visualize the embeddings of all the labeled node pairs on the AS dataset in Figure 5, to illustrate the relationship-preserving embedding of mg2vec. Particularly, for each approach, we concatenate the two embeddings of a node pair into a single vector, and project it into a 2D space using the t-SNE algorithm. The blue points represent node pairs of the *Customer* relationship, while the yellow points represent node pairs of the *Peer* relationship. The results show that DeepWalk and GraphSAGE do not reveal clear clusters and separate the two relationships; hin2vec produces many clusters, but most clusters are mixtures of both relationships. Moreover, M-HIN and MG+ generally separate the two relationships, but the boundaries are not well defined with quite extensive mixtures at the center. Compared to these methods, mg2vec separates the two relationships into much better defined clusters in its embedding space. Note that both *Peer* and *Customer* pairs further develop into many smaller clusters, which may naturally correspond to different groups of closely related pairs in the autonomous systems (*e.g.*, groups of ASes from different geographic locations).

Second, we visualize the metagraph embeddings, similarly projected to 2D planes as shown in Figure 6. We labeled metagraphs as preserving the *School* (green) or *Work* (blue) relationships on the LinkedIn dataseet, and ignored ambiguous metagraphs without a clear label (see Appendix C for the labeling methodology). Among the baselines, only M-HIN also learns metagraph embeddings. However, its metagraph embeddings do not preserve the latent relationships well. In contrast, mg2vec's metagraph embeddings are better separated by the two relationships.

The visualizations for the representations of both node pairs and metagraphs demonstrate that mg2vec is able to fully utilize metagraphs towards relationship-preserving embeddings on HINs.

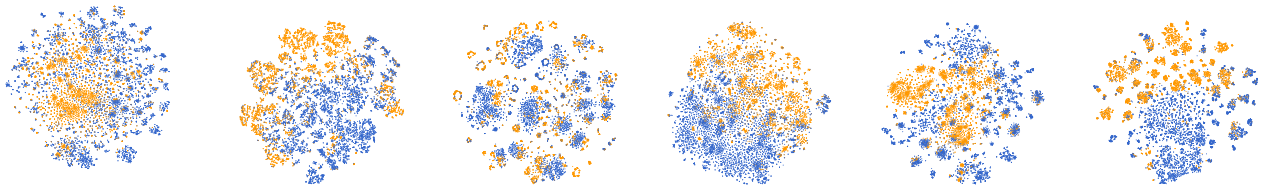
5.3 Metagraph Analysis

Next, we analyze the utility of metagraphs, the core instrument in our model mg2vec.

5.3.1 Metagraph size

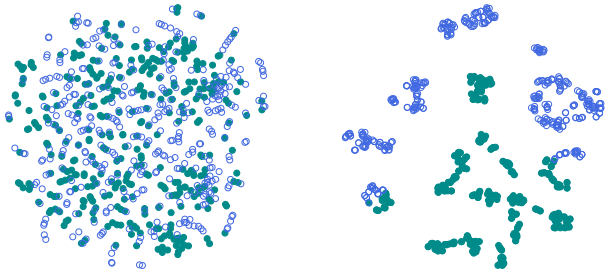
Generally, larger metagraphs are able to express more intricate relationships at the cost of time for the enumeration of metagraphs and their instances, as well as model training. The basic principle is to achieve a reasonable trade-off between effectiveness and efficiency.

To investigate the impact of metagraph size, we filter metagraphs up to a given size, and report the results in Figure 7 for relationship prediction and search. In general, the performance gradually increases with size, since larger metagraphs are more expressive. However, the gains taper off quickly and the performance tends to stabilize at size five or six. Thus, it is sufficient to constrain metagraphs to such sizes as similarly concluded by previous work [22], [23], since further increasing the size would generate limited returns but incur exponentially higher processing and training time. Moreover, it has been argued [22] that metagraphs of very large size may potentially introduce more noises, since the core nodes can reside too far apart in a very large metagraph.



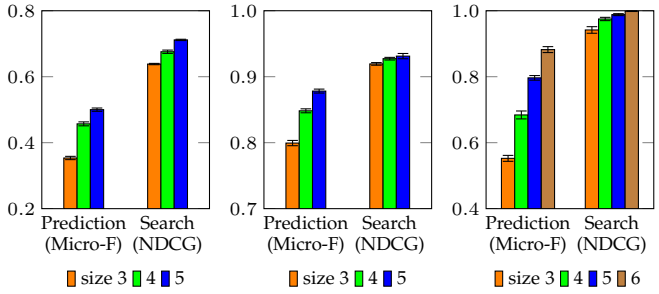
(a) DeepWalk (b) GraphSAGE (c) hin2vec (d) M-HIN (e) MG+ (f) mg2vec

Fig. 5: Visualization of all labeled node pairs on the AS dataset. Blue points denote *Customer* and yellow points denote *Peer*.



(a) M-HIN (b) mg2vec

Fig. 6: Visualization of metagraph embeddings for the LinkedIn dataset. Green points denote *School*-inclined metagraphs and blue points denote *Work*-inclined metagraphs.



(a) LinkedIn (b) AS (c) DBLP

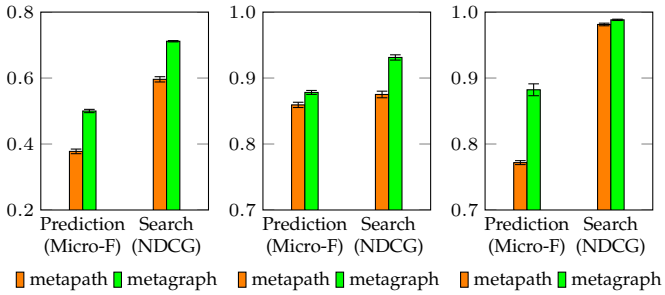
Fig. 7: Impact of metagraph size on mg2vec.

5.3.2 Complexity of metagraphs

We further investigate the impact of structural complexity of metagraphs. Note that metapath is a special subclass of metagraph with a simple path structure. In Figure 8, we compare the performance of using only metapaths as the input to mg2vec, and that of using all metagraphs as the input to mg2vec. We observe that, the structurally more complex metagraphs consistently achieve better performance than metapaths, demonstrating that metagraphs are more expressive and capture more informative semantics.

5.3.3 Case study

In this part, we showcase several example metagraphs on LinkedIn, which illustrate a variety of semantics relating two users nodes. For instance, the metagraph in Figure 9(a) is helpful to identify the the relationship between the two users as colleagues, whereas (b) implies the two users are potentially closer colleagues due to their common location, which is an important factor in large companies with multiple locations. Other relationships such as two users being



(a) LinkedIn (b) AS (c) DBLP

Fig. 8: Impact of structural complexity on mg2vec.

fellow alumni can be captured by the metagraph in (c). Lastly, there also exist many metagraphs that entail a fusion of semantics like the example in (d), which may infer a more intricate latent relationship such as being close personal friends whose friendship has been groomed over many years of shared hometown, education and employment.

5.4 Model analysis

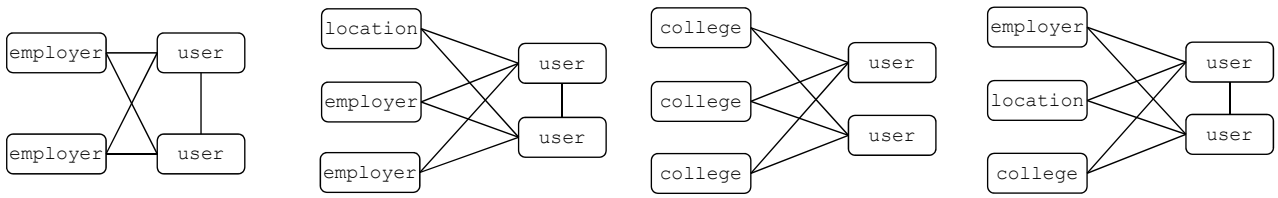
We further analyze our model by studying the impact of the hyper-parameter α and the embedding dimensions, as well as the rate of convergence during model training.

5.4.1 Impact of α

The ratio between first- and second-order metagraph embedding is a key factor in mg2vec, since it controls the trade-off between the latent preferences of individual nodes and interactions between a pair of nodes. As Figure 10 and 11 show, the best performance is generally achieved in the range of $[0.4, 0.6]$ for both relationship prediction and search, indicating that a balance between both the first- and second order is the most desirable configuration. In two extreme cases, when $\alpha = 0$, only the first order is accounted for; when $\alpha = 1$, only the second order is accounted for. In either case, the performance is generally worse than a balanced strategy, implying that it is not sufficient to consider the first or second order alone. Furthermore, $\alpha = 1$ is typically much better than $\alpha = 0$, which is intuitive as the second-order metagraph embedding directly deals with the latent interactions between nodes.

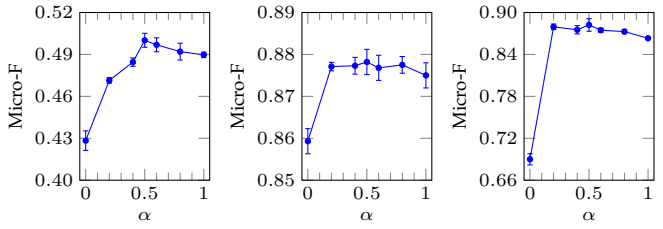
5.4.2 Impact of embedding dimensions

We further evaluate the impact of the embedding dimensions on relationship prediction and search, as shown in Figure 12 and 13, respectively. In both tasks, we vary the



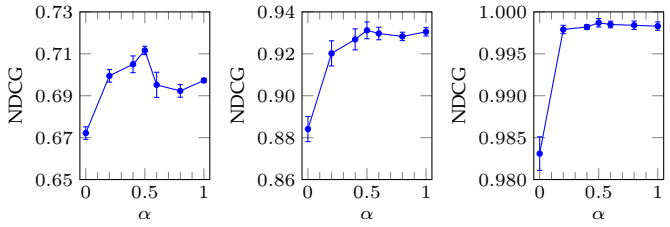
(a) Colleagues (b) Closer colleagues (c) Fellow alumni (d) Fusion

Fig. 9: Example metagraphs on LinkedIn.



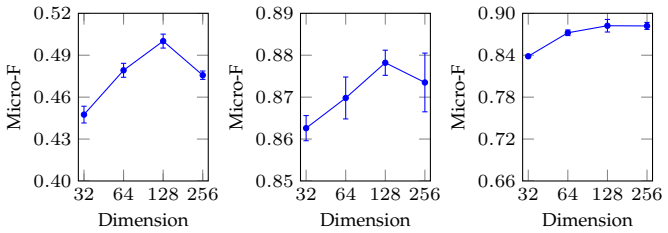
(a) LinkedIn (b) AS (c) DBLP

Fig. 10: Impact of α on relationship prediction.



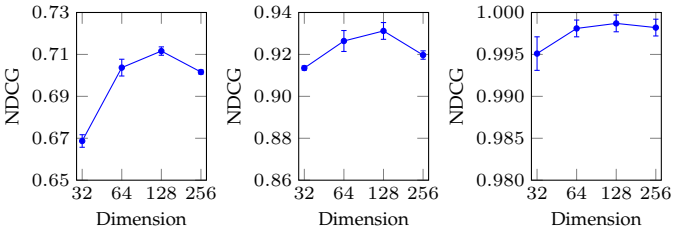
(a) LinkedIn (b) AS (c) DBLP

Fig. 11: Impact of α on relationship search.



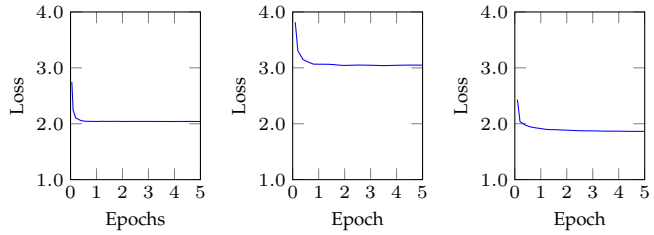
(a) LinkedIn (b) AS (c) DBLP

Fig. 12: Impact of dimensions on relationship prediction.



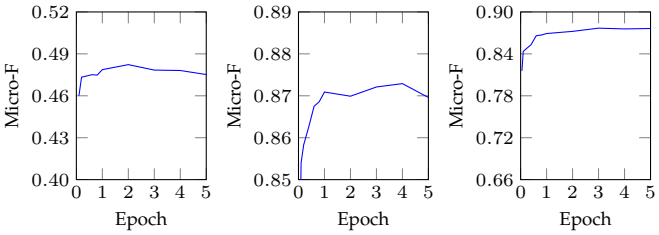
(a) LinkedIn (b) AS (c) DBLP

Fig. 13: Impact of dimensions on relationship search.



(a) LinkedIn (b) AS (c) DBLP

Fig. 14: Convergence of loss during training.



(a) LinkedIn (b) AS (c) DBLP

Fig. 15: Convergence of results on relationship prediction.

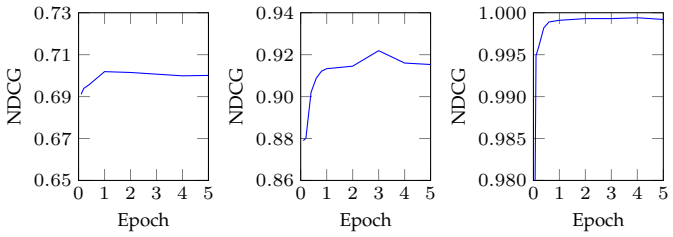
dimension between 32 and 256. Large dimensions often lead to over-parameterization, whereas small dimensions may result in insufficient modeling capacity. In our experiments, we observe that the performance of our model mg2vec is generally robust when the dimensions are set to around 128, which is also a typical choice established by previous work on network embedding.

5.4.3 Convergence

Finally, we investigate the rate of convergence in model training. We first examine the loss during training in Figure 14. On each dataset, the loss decreases rapidly in the initial epoch, and converges after about two or three epochs. The apparent reduction in loss and fast convergence imply that metagraphs are the right tool for modeling HINs. We further present the convergence of testing performance on each dataset in Figure 15 and 16 for relationship prediction and search, respectively. Consistent with the changes in training loss, the testing performance increases significantly in the first epoch, and gradually attain the best results after two or three epochs. There are some performance fluctuations after three epochs due to overfitting.

6 Conclusion and Future Work

In this paper, we studied relationship-preserving embedding on heterogeneous information networks. While metagraphs are a valuable tool to model rich semantics on HINs,



(a) LinkedIn (b) AS (c) DBLP

Fig. 16: Convergence of results on relationship search.

current approaches often under-utilize metagraphs, applying them only in a precomputation step. Thus, we proposed a novel self-supervised framework called mg2vec, employing metagraphs as first-class citizens to jointly embed both metagraphs and nodes into a common low-dimensional space. Specifically, in the joint embedding model, metagraphs actively guide the learning of node representations via both first- and second-order embeddings, to capture not only the latent preferences of individual nodes, but also the latent relationships between nodes. Finally, we conducted extensive experiments on three real-world public datasets. The results on various relationship mining tasks demonstrated the effectiveness of our model mg2vec, which consistently and significantly outperforms a wide range of state-of-the-art approaches.

There remain a few open questions for future work. First, to better capture the semantics between nodes that cannot be directly connected by smaller metagraphs, we may resort to focusing on a chain of key metagraphs, instead of only increasing the metagraph size. Simply increasing the size may not be suitable due to the diminishing return and higher processing time. Second, our proposed approach mg2vec is self-supervised by predicting the metagraphs associated with the nodes. It is also possible to bring in supervision from the downstream task so that we can automatically learn and differentiate the importance of metagraphs, such as through the neural attention mechanism.

ACKNOWLEDGEMENT

This research / project is supported by the Ministry of Education, Singapore under its Academic Research Funding Tier 1 (18-C220-SMU-006).

REFERENCES

[1] Z.-K. Zhang, T. Zhou, and Y.-C. Zhang, "Personalized recommendation via integrated diffusion on user-item-tag tripartite graphs," *Physica A: Stat. Mech. and Appl.*, vol. 389, no. 1, pp. 179–186, 2010.
[2] S. K. Ata, Y. Fang, M. Wu, X.-L. Li, and X. Xiao, "Disease gene classification with metagraph representations," *Methods*, vol. 131, pp. 83–92, 2017.
[3] C. Xiong, R. Power, and J. Callan, "Explicit semantic ranking for academic search via knowledge graph embedding," in *WWW*, 2017, pp. 1271–1279.
[4] H. Cai, V. W. Zheng, and K. Chang, "A comprehensive survey of graph embedding: problems, techniques and applications," *TKDE*, vol. 30, no. 9, pp. 1616–1637, 2018.
[5] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *KDD*, 2014, pp. 701–710.
[6] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *WWW*, 2015, pp. 1067–1077.
[7] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*, 2016, pp. 855–864.
[8] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *TKDE*, vol. 29, no. 1, pp. 17–37, 2017.
[9] M. Qu, J. Tang, J. Shang, X. Ren, M. Zhang, and J. Han, "An attention-based collaboration framework for multi-view network representation learning," in *CIKM*, 2017, pp. 1767–1776.
[10] H. Zhang, L. Qiu, L. Yi, and Y. Song, "Scalable multiplex network embedding," in *IJCAI*, 2018, pp. 3082–3088.
[11] Y. Ma, Z. Ren, Z. Jiang, J. Tang, and D. Yin, "Multi-dimensional network embedding with hierarchical structure," in *WSDM*, 2018, pp. 387–395.
[12] J. Li, C. Chen, H. Tong, and H. Liu, "Multi-layered network embedding," in *SDM*, 2018, pp. 684–692.
[13] V. W. Zheng, M. Sha, Y. Li, H. Yang, Y. Fang, Z. Zhang, K. Tan, and K. C. Chang, "Heterogeneous embedding propagation for large-scale e-commerce user alignment," in *ICDM*, 2018, pp. 1434–1439.
[14] H. Chen, H. Yin, W. Wang, H. Wang, Q. V. H. Nguyen, and X. Li, "Pme: projected metric embedding on heterogeneous networks for link prediction," in *KDD*, 2018, pp. 1177–1186.
[15] Y. Shi, Q. Zhu, F. Guo, C. Zhang, and J. Han, "Easing embedding learning by comprehensive transcription of heterogeneous information networks," in *KDD*, 2018, pp. 2190–2199.
[16] b. Hu, Y. Fang, and C. Shi, "Adversarial learning on heterogeneous information networks," in *KDD*, 2019, pp. 120–129.
[17] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "PathSim: Meta path-based top-k similarity search in heterogeneous information networks," *PVLDB*, vol. 4, no. 11, pp. 992–1003, 2011.
[18] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *KDD*, 2017, pp. 135–144.
[19] T.-y. Fu, W.-C. Lee, and Z. Lei, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *CIKM*, 2017, pp. 1797–1806.
[20] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *TKDE*, vol. 31, no. 2, pp. 357–370, 2019.
[21] Y. Fang, W. Lin, V. W. Zheng, M. Wu, K. C.-C. Chang, and X.-L. Li, "Semantic proximity search on graphs with metagraph-based learning," in *ICDE*, 2016, pp. 277–288.
[22] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis, and X. Li, "Meta structure: Computing relevance in large heterogeneous information networks," in *KDD*, 2016, pp. 1595–1604.
[23] Y. Fang, W. Lin, V. W. Zheng, M. Wu, J. Shi, K. Chang, and X. Li, "Metagraph-based learning on heterogeneous graphs," *TKDE*, vol. Early Access, 2019.
[24] C. Wang, J. Han, Y. Jia, J. Tang, D. Zhang, Y. Yu, and J. Guo, "Mining advisor-advisee relationships from research publication networks," in *KDD*, 2010, pp. 203–212.
[25] Z. Liu, V. W. Zheng, Z. Zhao, Z. Li, H. Yang, M. Wu, and J. Ying, "Interactive paths embedding for semantic proximity search on heterogeneous graphs," in *KDD*, 2018, pp. 1860–1869.
[26] T.-Y. Liu, *Learning to rank for information retrieval*. Springer, 2011.
[27] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *KDD*, 2017, pp. 635–644.
[28] L. Sun, L. He, Z. Huang, B. Cao, C. Xia, X. Wei, and S. Y. Philip, "Joint embedding of meta-path and meta-graph for heterogeneous information networks," in *ICBK*, 2018, pp. 131–138.
[29] A. Sankar, X. Zhang, and K. C.-C. Chang, "Meta-GNN: Metagraph neural network for semi-supervised learning in attributed heterogeneous information networks," in *ASONAM*, 2019, pp. 137–144.
[30] C. Yang, Y. Feng, P. Li, Y. Shi, and J. Han, "Meta-graph based HIN spectral embedding: Methods, analyses, and insights," in *ICDM*, 2018, pp. 657–666.
[31] H. Jiang, Y. Song, C. Wang, M. Zhang, and Y. Sun, "Semi-supervised learning over heterogeneous information networks by ensemble of meta-graph guided random walks." in *IJCAI*, 2017, pp. 1944–1950.
[32] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Metagraph2vec: Complex semantic path augmented heterogeneous network embedding," in *PAKDD*, 2018, pp. 196–208.
[33] Y. Fang, X. Zhao, P. Huang, W. Xiao, and M. de Rijke, "M-HIN: Complex embeddings for heterogeneous information networks via metagraphs," in *SIGIR*, 2019, pp. 913–916.

[34] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *TKDE*, vol. 31, no. 5, pp. 833–852, 2018.
[35] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *KDD*, 2016, pp. 1225–1234.
[36] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *KDD*, 2016, pp. 1105–1114.
[37] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "GraphGan: Graph representation learning with generative adversarial nets," in *AAAI*, 2018, pp. 2508–2515.
[38] Q. Dai, Q. Li, J. Tang, and D. Wang, "Adversarial network embedding," in *AAAI*, 2018, pp. 2167–2174.
[39] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[40] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.
[41] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017, pp. 1024–1034.
[42] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *KDD*, 2015, pp. 1165–1174.
[43] T. Trouillon, C. R. Dance, É. Gaussier, J. Welbl, S. Riedel, and G. Bouchard, "Knowledge graph completion via complex tensor factorization," *JMLR*, vol. 18, no. 1, pp. 4735–4772, 2017.
[44] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *ACL*, 1992, pp. 539–545.
[45] Y. Fang and K. C. Chang, "Searching patterns for relation extraction over the web: rediscovering the pattern-relation duality," in *WSDM*, 2011, pp. 825–834.
[46] D. Zeng, K. Liu, Y. Chen, and J. Zhao, "Distant supervision for relation extraction via piecewise convolutional neural networks," in *EMNLP*, 2015, pp. 1753–1762.
[47] Y. Lin, S. Shen, Z. Liu, H. Luan, and M. Sun, "Neural relation extraction with selective attention over instances," in *ACL*, 2016, pp. 2124–2133.
[48] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, 2013, pp. 2787–2795.
[49] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *AAAI*, 2015, pp. 2181–2187.
[50] Z. Liu, V. W. Zheng, Z. Zhao, F. Zhu, K. C.-C. Chang, M. Wu, and J. Ying, "Semantic proximity search on heterogeneous graph by proximity embedding," in *AAAI*, 2017, pp. 154–160.
[51] M. Elseidy, E. Abdelhamid, S. Skiadopoulos, and P. Kalnis, "GRAMI: Frequent subgraph and pattern mining in a single large graph," *PVLDB*, vol. 7, no. 7, pp. 517–528, 2014.
[52] W.-S. Han, J. Lee, and J.-H. Lee, "Turbo$_{ISO}$: towards ultrafast and robust subgraph isomorphism search in large graph databases," in *SIGMOD*, 2013, pp. 337–348.
[53] F. Bi, L. Chang, X. Lin, L. Qin, and W. Zhang, "Efficient subgraph matching by postponing Cartesian products," in *SIGMOD*, 2016, pp. 1199–1214.
[54] S. Sun, Y. Che, L. Wang, and Q. Luo, "Efficient parallel subgraph enumeration on a single machine," in *ICDE*, 2019, pp. 232–243.
[55] R. Li, C. Wang, and K. C.-C. Chang, "User profiling in an ego network: co-profiling attributes and relationships," in *WWW*, 2014, pp. 819–830.
[56] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *KDD*, 2005, pp. 177–187.

APPENDIX A
METAPATHS FOR METAPATH2VEC

For a fair comparison of metapath2vec against mg2vec, we examined all constituent metapaths of the metagraphs used in mg2vec. For each dataset, we identified the list of metapaths such that *any* metagraph used in mg2vec can be assembled from these metapaths, as reported in Table 5. We evaluated the performance of metapath2vec using each of the listed metapaths, and additionally, using *all* of the metapaths simultaneously. Results show that using all metapaths would indeed give a better performance on both relationship prediction and search, although it still falls short of the performance of mg2vec. Note that we also tried some combinations of the more promising metapaths (as judged by their performance on the tasks), and obtained similar results to using all metapaths. Thus, in our empirical results in Section 5.2, we reported only the optimal performance on each dataset, when all metapaths in Table 5 were used.

TABLE 5: Constituent metapaths of all metagraphs used by mg2vec on each dataset. LinkedIn: user (u), college (c), employer (e), location (l); AS: middle ISP (m), bottom ISP (b), top ISP (t); DBLP: author (a), paper (p), venue (v), keyword (k), year (y).

LinkedIn	AS			DBLP
u-u	m-m	m-b-m	m-t-m	a-p-a
u-c-u	m-b-b-m	m-b-t-m	m-t-t-m	a-p-v-p-a
u-e-u	m-b-b-b-m	m-b-b-t-m	m-b-t-b-m	a-p-k-p-a
u-l-u	m-b-t-t-m	m-t-b-t-m	m-t-t-t-m	a-p-y-p-a

APPENDIX B
GRID SEARCH FOR BASELINES

We performed a grid search to select the hyper-parameters of each baseline method. Specifically, we reserved 20% of the training data used in the task of relationship prediction as the validation set. After choosing the optimal parameters and learning the embeddings, we still used all training data to build a model for the downstream tasks.

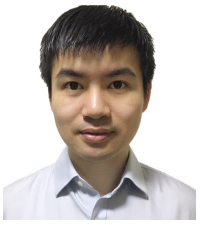
For each baseline, we list the main parameters and their values tuned in the grid search. Other parameters not mentioned adopt their recommended values reported in the literature. For DeepWalk [5], hin2vec [19] and metapath2vec [18], we tuned the number of walks over {5, 10, 20} and walk length over {50, 100, 150}. We further tuned the metapaths used in metapath2vec as detailed in Appendix A. For LINE [6], we tuned the order of proximity over {1st, 2nd, 1st+2nd} and the learning rate over {0.01, 0.025, 0.05}. For GraphSAGE [41], we tuned the inner dim over {32, 64, 128} and chosen the aggregator from {mean, meanpool, maxpool}. For TransR [49], we chose the training method from {unif, bern} and tuned the learning rate over {0.001, 0.01, 0.05}. For HeGAN [16], we tuned the Guassian variance over {0.1, 1, 10} and the number of inner epochs over {1g/5d, 5g/15d, 10g/30d} (g for generator and d for discriminator). For metagraph2vec [32], we tuned the window size over {3, 5} and the learning rate over {0.01, 0.025, 0.05}. For M-HIN [33] we tuned the learning rate over {0.01, 0.025, 0.05}. For MG+, there is no hyper-parameter to tune since it uses the frequency of the metagraph instances as graph representations directly without any training required.

APPENDIX C
LABELING METAGRAPHS

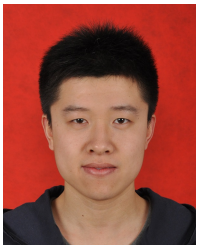
In Section 5.2.3, we visualized the metagraph embeddings to show that different metagraphs can preserve different relationships on the LinkedIn dataset. We manually designed the rules to label the metagraphs as preserving *Work* or *School* relationships. The general guideline is to look for college and employer nodes. Typically, if a metagraph contains mostly employer nodes rather than college nodes,

such as the examples in Figure 9(a) and (b), we labeled the metagraph as *Work*-inclined; if it contains mostly college nodes like Figure 9(c), we labeled it as *School*-inclined. Sometimes a metagraph, as shown in Figure 9(d), can be a fusion of semantics, *e.g.*, close personal friends whose friendship has been groomed over years of shared hometown, education and employment. We regarded such metagraphs as ambiguous and omitted them from the visualization.

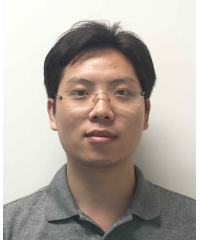
Wentao Zhang received his Bachelor's degree in Computer Science from the University of Science and Technology of China, Hefei, China in 2018. He is currently a Master's student at the University of Science and Technology of China, and was a Visiting Research Student at the School of Information Systems, Singapore Management University, Singapore between August 2019 and April 2020. His research interests include graph representation learning.



Yuan Fang received his Ph.D. degree in Computer Science from the University of Illinois at Urbana-Champaign, United States in 2014, and Bachelor's degree in Computer Science from National University of Singapore, Singapore in 2009. He is currently an Assistant Professor at the School of Information Systems, Singapore Management University, Singapore. Previously, he was a scientist at the Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore and a data scientist at DBS Bank, Singapore. His work has been featured in the Best Papers collection of VLDB 2013. His current research focuses on graph-based machine learning, Web and social media mining, recommendation systems and bioinformatics.



Zemin Liu received his Ph.D. degree in Computer Science from Zhejiang University, Hangzhou, China in 2018, and B.S. Degree in Software Engineering in Shandong University, Jinan, China in 2012. He is currently a Research Scientist in the School of Information Systems, Singapore Management University, Singapore. His research interests include graph representation learning.



Min Wu is currently a Senior Scientist in Data Analytics Department, Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore. He received his Ph.D. degree in Computer Science from Nanyang Technological University, Singapore in 2011 and B.S. degree in Computer Science from the University of Science and Technology of China, Hefei, China in 2006. He received two best paper awards in InCoB 2016 and DASFAA 2015. He also won the IJCAI competition on repeated buyers prediction in 2015. His current research interests include machine learning, graph mining and bioinformatics.

Xinming Zhang received the B.E. and M.E. degrees in Electrical Engineering from China University of Mining and Technology, Xuzhou, China in 1985 and 1988, respectively, and the Ph.D. degree in Computer Science and Technology from the University of Science and Technology of China, Hefei, China in 2001. Since 2002, he has been with the faculty of the University of Science and Technology of China, where he is currently a Professor with the School of Computer Science and Technology. From September 2005 to August 2006, he was a Visiting Professor with the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, Korea. His research interest includes wireless networks, big data and deep learning. He has published more than 100 papers. He won the second prize of Science and Technology Award of Anhui Province of China in Natural Sciences in 2017. He won the awards of Top Reviewers (1%) in Computer Science & Cross Field by Publons in 2019. He is a senior member of IEEE.