Learning Multi-Granularity and Adaptive Representation for Knowledge Graph Reasoning

Ziyu Shang[1] , **Peng Wang**[1,2∗] , **Wenjun Ke**[1,2∗] , **Jiajun Liu**[1] , **Hailang Huang**[3] , **Guozheng Li**[1] , **Chenxiao Wu**[1] , **Jianghan Liu**[4] , **Xiye Chen**[5] and **Yining Li**[4]
[1]School of Computer Science and Engineering, Southeast University
[2]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education
[3]SKLSDE, School of Computer Science and Engineering, Beihang University
[4]College of Software Engineering, Southeast University
[5]College of Information Engineering, Nanjing University of Finance & Economics
{ziyus1999, pwang, kewenjun, jiajliu, gzli, liujianghan, liyining}@seu.edu.cn
lerogo@buaa.edu.cn, {yanzu0311, chemistrymaths2002}@gmail.com

Abstract

Knowledge graph reasoning (KGR) aims to infer new factual triples from existing knowledge graphs (KGs). Recently, a new category of methods, possessing both transductive and inductive reasoning capabilities, has been proposed to tackle this task via learning entity-independent representations from local neighboring structures. However, these methods are plagued by inefficiency issues and they exclusively capture evidence from well-designed local structures, ignoring the correlation between the query and different structures within KGs. In this work, we first propose a novel multi-granularity and adaptive representation framework, MulGA, exploiting the connectivity subgraph to uniformly and hierarchically model query-related triples, relation paths, and subgraphs without explicitly extracting any graph structure, hence mitigating inefficiency issues. Second, we introduce a message-passing mechanism across connectivity subgraphs, facilitating all entities to attain query-related structural representations of diverse granularity levels, *i.e.*, triple and relation paths of different lengths. Third, we design a self-attention-based merging mechanism that allocates weights to different granularities and then consolidates them into subgraph granularity representations for reasoning. The systematic experiments have been conducted on 15 benchmarks and MulGA achieves a significant improvement in MRR by an average of 1.5% on transductive and 2.7% on inductive tasks than existing state-of-the-art methods. Moreover, MulGA boasts faster convergence speed, competitive inference time, and alleviates the over-smoothing prevalent in graph neural networks.

∗Corresponding authors

1 Introduction

Knowledge graphs (KGs) [Dong *et al.*, 2014] such as Wikidata [Vrandečić, 2012] and DBPedia [Auer *et al.*, 2007] have been widely utilized in many knowledge-intensive applications including semantic search [Noy *et al.*, 2019; Dong, 2019], recommendation systems [Wang *et al.*, 2018; Wang *et al.*, 2021], and question answering [Huang *et al.*, 2019; Saxena *et al.*, 2020; Li *et al.*, 2022]. A KG is a multi-relational graph containing factual triples. However, real-world KGs constructed manually or automatically are inevitably incomplete, thus predicting unknown edges as analogous to uncovering new facts is challenging. Traditional knowledge graph reasoning (KGR) methods leverage the knowledge graph embedding (KGE) [Wang *et al.*, 2017], projecting entities and relations into low-dimensional dense vectors, to aid the inference of missing factual triples [Bordes *et al.*, 2013; Sun *et al.*, 2019; Liu *et al.*, 2023; Shang *et al.*, 2023]. Despite the better expressive capacity of KGE-based models, there remains an inherent deficiency in their explainability. For effective KGR, it is pivotal to acquire high-quality representations that not only embed the factual triples but also accurately reflect the underlying graph structure [Chen *et al.*, 2020b].

Recent KGR studies [Zhu *et al.*, 2021; Zhang and Yao, 2022; Teru *et al.*, 2020; Xu *et al.*, 2022; Mai *et al.*, 2021] have moved away from KGE-based methods and focus instead on learning entity-independent representation from existing structures in KGs, *e.g.*, relation paths and subgraphs. For methods based on relation paths, recent work [Zhu *et al.*, 2021; Zhang *et al.*, 2022; Zhang and Yao, 2022; Zhu *et al.*, 2023] emphasizes capturing features of fixed-length relation paths between two entities in KGs, overlooking the semantic correlation between the query and local evidence contained in paths of varying lengths. Intuitively, the local evidence reasoning required is dependent on the context of the query. For example, for query $(Messi, lives\_in, ?)$, we should capture evidence from the local structure of the `2-hop Path` between *Messi* and *Florida*. If the query changes to $(Messi, plays\_in, ?)$, `3-hop Path` deserves more atten-

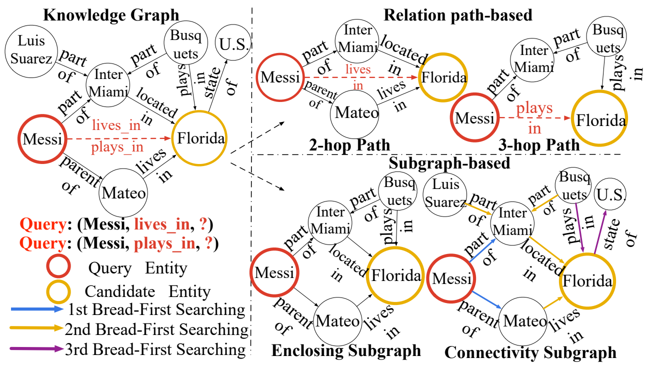

Figure 1: An example of two prevailing KGR solutions on a KG about *Messi*. Relation path-based methods only capture limited and local information, while enclosing subgraph-based methods may contain redundant extraction. Our proposed connectivity subgraph has better capability for efficiently modeling subgraphs in KGs.

tion. To mine the local evidence contained in variable-length relation paths, subgraph-based methods [Teru *et al.*, 2020; Xu *et al.*, 2022; Mai *et al.*, 2021] are proposed, exploiting the enclosing subgraph effectively. However, they overlook the fact that such subgraph modeling is inefficient and inevitably involves unrelated entities and relations, resulting in equal attention to different structures, increasing training difficulty, and introducing noise. Specifically, GraIL [Teru *et al.*, 2020] and the subsequent works [Xu *et al.*, 2022] first extract enclosing subgraphs between the query entity and other candidate entities. Then, they repeatedly apply graph neural networks (GNNs) to enclosing subgraphs. We identify the trend of repeated extraction of these enclosing subgraphs is unnecessary. As illustrated in Figure 1, while extracting *2-hop* enclosing subgraphs between *Messi* and *Florida*, and *Messi* and *Busquets* respectively, the resulting subgraph structures are identical, suggesting that these methods could expend excessive time on extracting identical subgraph. Moreover, they also ignore the correlation between queries and structures, *e.g.*, `2-hop Path` and `3-hop Path` between *Messi* and *Florida* are treated equally for different queries.

To address the above limitations, we propose a novel multi-granularity and adaptive representation framework for KGR, referred to as MulGA. To alleviate the inefficiencies associated with explicit enclosing subgraph extraction and to expand the scope of subgraph information, MulGA initially exploits the connectivity subgraph that can be implicitly extracted via the breadth-first search (BFS) algorithm. As depicted in Figure 1, the *2-hop* enclosing subgraph between *Messi* and *Florida*, which involves the longest relation path of length 3, can be replaced with a connectivity subgraph of size 3. This connectivity subgraph not only accurately models structures of the enclosing subgraph but also embodies the neighboring relations as considered in the SNRI [Xu *et al.*, 2022]. The iterative process of the BFS aligns with the extraction of enclosing subgraphs. Consequently, a connectivity subgraph can seamlessly represent multiple enclosing subgraphs, optimizing time and resources. Additionally, MulGA presents a GNN-based multi-granularity embedding propagation module across connectivity subgraphs to produce two levels of granularity embeddings hierarchically, *i.e.*, triple, and relation paths. Moreover, we propose a query-aware attention mechanism to select query-related edges during propagation to enrich the relation path granularity. Furthermore, to leverage query-relevant features contained in different structures, we design a self-attention-based merging mechanism that adaptively assigns query-related weights to different granularity embeddings, taking into account the correlation among them. Subsequently, subgraph granularity embeddings can be obtained by merging the above two granularity embeddings for reasoning. In experiments, we find that the proposed merging mechanism can alleviate over-smoothing in GNNs.

Our key contributions are summarized as follows:

- We adopt the connectivity subgraph to effectively model the various structures in KGs to achieve a hierarchical representation of triples, relational paths, and subgraphs, optimizing time and resources.
- To obtain multi-granularity embeddings, we design a GNN-based multi-granularity embedding propagation to learn query-related granularity embedding hierarchically.
- We introduce a self-attention-based merging mechanism that adaptively assigns query-related weight to different granularities and ultimately fuse them into robust subgraph granularity representations for reasoning.
- Extensive experiments have been conducted on 15 benchmarks involving both transductive and inductive reasoning tasks. MulGA achieves a significant improvement in mean reciprocal rank (MRR) by an average of 1.5% on transductive and 2.7% on inductive tasks.

2 Preliminary

2.1 Notations

Definition 1. *(Knowledge Graph). A knowledge graph (KG) $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ is defined by the set of entities $\mathcal{E}$, relations $\mathcal{R}$, and triples $\mathcal{T}$. A triple is usually denoted as $(h, r, t) \in \mathcal{T}$, where $h \in \mathcal{E}$, $t \in \mathcal{E}$ and $r \in \mathcal{R}$ represent the head entity, the tail entity and the relation between h and t, respectively.*

Definition 2. *(Transductive KGR). Given a KG $\mathcal{G}$, transductive KGR task refers to predicting either the tail entity $t \in \mathcal{E}$ given $h \in \mathcal{E}$ and $r \in \mathcal{R}$, i.e., $(h, r, ?)$, or the head entity $h \in \mathcal{E}$ given $r \in \mathcal{R}$ and $t \in \mathcal{E}$, i.e., $(?, r, t)$, where $(h, r, t) \notin \mathcal{G}$.*

Definition 3. *(Inductive KGR). Given a source KG $\mathcal{G}_S = (\mathcal{E}_S, \mathcal{R}_S, \mathcal{T}_S)$ and a target KG $\mathcal{G}_T = (\mathcal{E}_T, \mathcal{R}_T, \mathcal{T}_T)$, where $\mathcal{G}_T$ consists of unseen entities not in the $\mathcal{G}_S$: $\mathcal{E}_T \cap \mathcal{E}_S = \varnothing$, $\mathcal{R}_T \subseteq \mathcal{R}_S$. Inductive KGR task aims to train a model on the source KG $\mathcal{G}_S$ and then predict either the tail entity $t \in \mathcal{E}_T$ given $h \in \mathcal{E}_T$ and $r \in \mathcal{R}_T$, i.e., $(h, r, ?)$, or the head entity $h \in \mathcal{E}_T$ given $r \in \mathcal{R}_T$ and $t \in \mathcal{E}_T$, i.e., $(?, r, t)$, where $(h, r, t) \notin \mathcal{G}_T$.*

3 Method

Figure 2 illustrates the overall architecture of MulGA. Given a query $(h, r, ?)$ or $(?, r, e)$, MulGA initiates by implicitly extracting the connectivity subgraph utilizing BFS. Meanwhile, MulGA propagates structural messages from the query entity

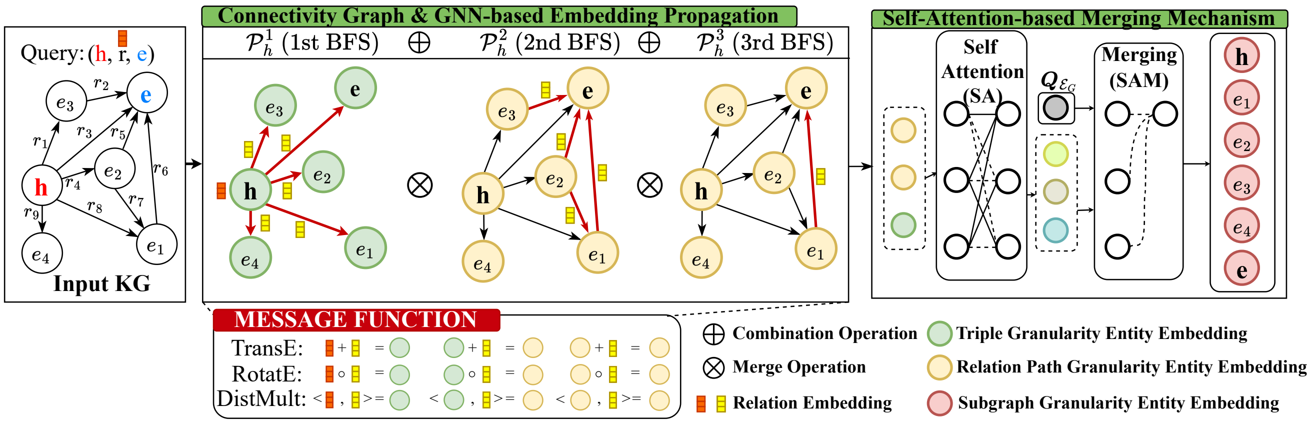
Query: (h, r, e)
Input KG
Connectivity Graph & GNN-based Embedding Propagation
$\mathcal{P}_h^1$ (1st BFS)
$\mathcal{P}_h^2$ (2nd BFS)
$\mathcal{P}_h^3$ (3rd BFS)
Self-Attention-based Merging Mechanism
Self Attention (SA)
$Q_{\mathcal{E}_G}$
Merging (SAM)
MESSAGE FUNCTION
TransE:
RotatE:
DistMult:
Combination Operation
Merge Operation
Relation Embedding
Triple Granularity Entity Embedding
Relation Path Granularity Entity Embedding
Subgraph Granularity Entity Embedding

Figure 2: An overview of our proposed MulGA framework.

to candidate entities, producing two types of granularity embeddings: triples and relation paths of varying lengths. Finally, the above two granularity embeddings are merged through the proposed self-attention-based merging mechanism, generating the final subgraph granularity embedding for reasoning.

3.1 Connectivity Subgraph

Given a query $(h, r, ?)$, intuitively, to perform reasoning, it is vital to utilize various structural information surrounding the query entity h. The connectivity subgraph can model adequate structures around query entities for this purpose.

Definition 4. *(Connectivity Subgraph). Given an entity h, a connectivity subgraph $\mathcal{G}_h^n$ can be modeled as follows:*

$$\mathcal{G}_h^n = \mathcal{P}_h^0 \oplus \mathcal{P}_h^1 \oplus \mathcal{P}_h^2 \oplus \cdots \oplus \mathcal{P}_h^n \tag{1}$$

where n is the size of the connectivity subgraph. $\mathcal{P}_h^i$, namely a connectivity subgraph component, is the set of relation paths of length i originating from h. Besides, $\oplus$ denotes the combination operator on the path set.

Let $\mathcal{E}(\mathcal{P}_h^{i,j})$ be the set of j_{th} entity on all i-length relation paths originating from h, and $\mathcal{E}(\mathcal{P}_h^{i,0}) = \{h\}$. Therefore, the entities in $\mathcal{P}_h^i$ are denoted as $\mathcal{E}(\mathcal{P}_h^i) = \mathcal{E}(\mathcal{P}_h^{i,0}) \cup \mathcal{E}(\mathcal{P}_h^{i,1}) \cup \cdots \cup \mathcal{E}(\mathcal{P}_h^{i,i})$. Analogously, the relations in $\mathcal{P}_h^i$ are denoted as $\mathcal{R}(\mathcal{P}_h^i) = \mathcal{R}(\mathcal{P}_h^{i,1}) \cup \mathcal{R}(\mathcal{P}_h^{i,2}) \cup \cdots \cup \mathcal{R}(\mathcal{P}_h^{i,i})$, where $\mathcal{R}(\mathcal{P}_h^{i,j})$ is the set of j_{th} relation on all i-length relation paths originating from h. Besides, $\mathcal{E}(\mathcal{P}_h^{i,j}) = \mathcal{R}(\mathcal{P}_h^{i,j}) = \varnothing,\ if\ j > i$.

From Definition 4, a connectivity subgraph of size n encompasses all entities located at a distance no more than n from the query entity, along with their corresponding structures. Following the previous work [Xu *et al.*, 2019], inverse edges of each factual triple and self-loop edge of each entity are added to original KGs. Specifically, if (h, r, t) exists in KGs, then (t, r^{-1}, h), $(h, self\_loop, h)$, and $(t, self\_loop, t)$ are added, where r^{-1} is the inverse of r, to original KGs. Therefore, the iterative formula for components can be obtained when $i \geq 1$:

$$\mathcal{P}_h^i = \mathcal{P}_h^{i-1} \oplus \{(e, r, t) | e \in \mathcal{E}(\mathcal{P}_h^{i-1,i-1}) \wedge (e, r, t) \in \mathcal{G}\} \tag{2}$$

In this way, the connectivity subgraph can be easily implicitly extracted using the BFS algorithm. For example, as shown in Figure 2, the red arrows indicate the new connectivity subgraph component found by BFS. Furthermore, a connectivity subgraph $\mathcal{G}_h^3$ can concurrently embody the enclosing subgraph formed between h and e, e_1, e_2, e_3, e_4, which reduces the time-consuming extraction of duplicated structures.

3.2 GNN-based Multi-Granularity Embedding Propagation

The GNN-based multi-granularity embedding propagation (GMEP) utilizes the GNN to produce two levels of granularity embeddings, *i.e.*, triple and relation path, of entities in $\mathcal{G}_h^n$.

Triple Granularity Embedding

According to Definition 4, $\mathcal{P}_h^1$ is the set of triples in the form of (h, r, e). Therefore, taking entity e as an example, we can perform representation learning directly on $\mathcal{P}_h^1$ and then obtain corresponding triple granularity embedding $\boldsymbol{e}_{\mathcal{T}_h} = \boldsymbol{e}_{\mathcal{P}_h^1}$.

Relation Path Granularity Embedding

Similarly, in the Definition 4, $\mathcal{P}_h^i$ $(2 \leq i \leq n)$ means the set of all i-length relation paths originating from h. We can perform representation learning on the $\mathcal{P}_h^i$ to obtain the corresponding embeddings. Therefore, taking entity e as an example, the relation path granularity embeddings $\boldsymbol{e}_{\mathcal{P}_h}$ of different lengths in $\mathcal{G}_h^n$ can be denoted as $\boldsymbol{e}_{\mathcal{P}_h} = [\boldsymbol{e}_{\mathcal{P}_h^2}, \boldsymbol{e}_{\mathcal{P}_h^3}, \cdots, \boldsymbol{e}_{\mathcal{P}_h^n}]$.

Propagation Mechanism

According to Eq. 2, the triple structure $\mathcal{P}_h^1$ is obtained before the relational path, thus enabling the hierarchical calculation of relational path granularity based on the triple granularity. Drawing inspiration from the hierarchical message-passing mechanism in GNN [Scarselli *et al.*, 2008], we design a propagation module that takes the query entity as initial subgraph $(\mathcal{P}_h^0)$. This module dynamically expands the current graph structure following BFS, facilitating layer-by-layer propagation of structural information. Through this sequential propagation, we obtain various granularities embedding, which ensures a thorough capture of structural information contained in KGs. Consequently, we can directly apply one layer of GNN in each connectivity subgraph component. Thus, the i-th layer of our GNN can be calculated as follows:

$$\boldsymbol{v}_{\mathcal{P}_h^i} = \mathrm{ReLU}((\sum_{u \in In\left(v|\mathcal{E}(\mathcal{P}_h^{i-1,i-1})\right)} \alpha^i_{ur_tv}\phi(\boldsymbol{u}_{\mathcal{P}_h^{i-1}}, \boldsymbol{r}_t^i))\boldsymbol{W}_3^{i\,T} + \boldsymbol{b}_3^i) \quad (3)$$

$$\alpha^i_{ur_tv} = \sigma\left(\boldsymbol{c}\boldsymbol{W}_2^{i\,T} + \boldsymbol{b}_2^i\right) \quad (4)$$

$$\boldsymbol{c} = \mathrm{ReLU}\left(\left[\boldsymbol{u}_{\mathcal{P}_h^{i-1}}; \boldsymbol{r}_t^i; \boldsymbol{r}^i; \boldsymbol{h}_{\mathcal{P}_h^{i-1}}\right]\boldsymbol{W}_1^{i\,T} + \boldsymbol{b}_1^i\right) \quad (5)$$

where i denotes the i-th connectivity subgraph component; $\phi(\cdot,\cdot)$ means MESSAGE function, which is used to pass known structural information, and can be used with most off-the-shelf KGE methods, such as TransE [Bordes *et al.*, 2013], RotatE [Sun *et al.*, 2019], and DistMult [Yang *et al.*, 2015]. $\boldsymbol{r}_t^i$ denotes the embedding of the relation between entity u in $\mathcal{P}_h^{i-1}$ and entity v in $\mathcal{P}_h^i$ at the i-th layer; $In\left(v|\mathcal{E}(\mathcal{P}_h^{i-1,i-1})\right)$ represents the entities on the $\mathcal{E}(\mathcal{P}_h^{i-1,i-1})$ pointing to entity v in $\mathcal{P}_h^i$. Since multiple queries share the same connectivity subgraph, we design the attention mechanism to adaptively assign weights to each edge. $\alpha^i_{ur_tv}$ is the edge attention weight at the i-th layer corresponding to the edge (u, r_t, v); $\sigma(\cdot)$ denotes the sigmoid function; $[\cdot;\cdot]$ denotes tensors concatenation; $\boldsymbol{W}_1^i \in \mathbb{R}^{d\times 4d}$, $\boldsymbol{W}_2^i \in \mathbb{R}^{1\times d}$, $\boldsymbol{W}_3^i \in \mathbb{R}^{d\times d}$, $\boldsymbol{b}_1^i \in \mathbb{R}^{1\times d}$, $\boldsymbol{b}_2^i \in \mathbb{R}^{1\times 1}$, and $\boldsymbol{b}_3^i \in \mathbb{R}^{1\times d}$ are trainable parameters at the i-th layer; d is embedding dimensions of the entity and relation.

Since GMEP propagates message in the connectivity subgraph from query entity h to the other entities, we can initialize the representation of the entity h as follows:

$$\boldsymbol{h}_{\mathcal{P}_h^0} = \boldsymbol{r}\boldsymbol{W}_4^T + \boldsymbol{b}_4 \quad (6)$$

where $\boldsymbol{W}_4 \in \mathbb{R}^{d\times d}$, $\boldsymbol{b}_4 \in \mathbb{R}^{1\times d}$ are trainable parameters and $\boldsymbol{r}$ denotes the embedding of query relation.

Finally, we stack n layers of GNN in Eq. 3 to form GMEP module. As a result, the triple embedding $\boldsymbol{e}_{\mathcal{T}_h}$ (1-th GNN layer) and relation path embeddings of different lengths $\boldsymbol{e}_{\mathcal{P}_h}$ (from 2-th to n-th GNN layers) can be produced.

Subgraph Granularity Embedding

Similar to triple and relation path granularity, subgraph granularity can be derived via representation learning across the entire connectivity subgraph. However, this may result in duplicated calculations. To avoid this, given representations of connectivity components, we can merge them to generate the subgraph granularity. Taking entity e as an example, subgraph embedding of the entity e in $\mathcal{G}_h^n$ can be expressed as $\boldsymbol{e}_{\mathcal{G}_h^n}$:

$$\boldsymbol{e}_{\mathcal{G}_h^n} = \otimes(\boldsymbol{e}_{\mathcal{T}_h}, \boldsymbol{e}_{\mathcal{P}_h}) = \otimes(\boldsymbol{e}_{\mathcal{P}_h^1}, \boldsymbol{e}_{\mathcal{P}_h^2}, \cdots, \boldsymbol{e}_{\mathcal{P}_h^n}) \quad (7)$$

where $\otimes$ signifies the merging operation, and $\boldsymbol{e}_{\mathcal{P}_h^i} = \boldsymbol{0}$, if $\mathcal{E}(\mathcal{P}_h^i) \cap \{e\} = \varnothing$. For merging operations, there are several common options, such as *max*, *sum*, *mean*, *concat*. However, the particularities of different granularity embeddings are ignored in such merging operations. To this end, we propose a self-attention-based merging mechanism, which can incorporate structural features among different granularities.

3.3 Self-Attention-based Merging Mechanism

Given n different granularity embeddings $\boldsymbol{H} = [\boldsymbol{e}_{\mathcal{P}_h^1}, \boldsymbol{e}_{\mathcal{P}_h^2}, \cdots, \boldsymbol{e}_{\mathcal{P}_h^n}] \in \mathbb{R}^{n\times d}$, the self-attention-based merging (SAM) mechanism, inspired by the Transformer [Vaswani *et al.*, 2017; Lee *et al.*, 2019], is to fuse n different granularity embeddings into a subgraph embedding, i.e., $\mathbb{R}^{n\times d} \to \mathbb{R}^{1\times d}$. Concretely, the SAM mechanism consists of two functions: the multi-head attention function (MA), and the self-attention function (SA).

The Multi-head Attention Function: To tackle the limitation of simple merging methods to distinguish important granularity, we use the multi-head attention (MA) [Vaswani *et al.*, 2017], which can be formulated as follows:

$$\mathrm{MA}(\boldsymbol{H}, \boldsymbol{H}, \boldsymbol{H}) = \mathrm{Concat}(head_1, \cdots, head_m)\boldsymbol{W}_O \quad (8)$$

$$head_i = \mathrm{Att}(\boldsymbol{H}\boldsymbol{W}_Q^i,\ \boldsymbol{H}\boldsymbol{W}_K^i,\ \boldsymbol{H}\boldsymbol{W}_V^i) \quad (9)$$

where $\mathrm{Att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = w(\boldsymbol{Q}\boldsymbol{K}^T)\boldsymbol{V}$, w is an activation function; $\boldsymbol{W}_Q^i \in \mathbb{R}^{d\times d}$, $\boldsymbol{W}_K^i \in \mathbb{R}^{d\times d}$, $\boldsymbol{W}_V^i \in \mathbb{R}^{d\times d}$ are trainable parameters and the output projection matrix is $\boldsymbol{W}_O \in \mathbb{R}^{md\times d_{final}}$, where d_{final} is the output dimension of the multi-head attention function.

Self-attention Function: Another evident disadvantage of simple merging operations is deemed as the neglect of relationships among different granularity embeddings. To this end, we introduce a self-attention (SA) function which can be formulated as follows:

$$\mathrm{SA}(\boldsymbol{H}) = \mathrm{LN}\left(\boldsymbol{T} + \mathrm{rFF}\left(\boldsymbol{T}\right)\right) \quad (10)$$

$$\boldsymbol{T} = \mathrm{LN}\left(\boldsymbol{H} + \mathrm{MA}\left(\boldsymbol{H},\ \boldsymbol{H},\ \boldsymbol{H}\right)\right) \quad (11)$$

where LN is the LayerNorm function, and rFF is any row-wise feedforward layer.

Overall Self-Attention-based Merging Mechanism: In order to fuse n different granularity embeddings into a more powerful subgraph embedding $\boldsymbol{e}_{\mathcal{G}_h^n}$, we introduce a trainable parameter $\boldsymbol{Q}_{\mathcal{E}_G} \in \mathbb{R}^{1\times d}$ that is directly optimized in an end-to-end fashion, as follows:

$$\boldsymbol{e}_{\mathcal{G}_h^n} = \mathrm{LN}\left(\boldsymbol{T} + \mathrm{rFF}\left(\boldsymbol{T}\right)\right) \quad (12)$$

$$\boldsymbol{T} = \mathrm{LN}\left(\boldsymbol{Q}_{\mathcal{E}_G} + \mathrm{MA}\left(\boldsymbol{Q}_{\mathcal{E}_G},\ \mathrm{SA}(\boldsymbol{H}),\ \mathrm{SA}(\boldsymbol{H})\right)\right) \quad (13)$$

Finally, for the query $(h, r, ?)$, we define a simple function to score the candidate entity e, as follows:

$$f(h,\ r,\ e) = \boldsymbol{e}_{\mathcal{G}_h^n}\boldsymbol{W}_5^T + \boldsymbol{b}_5 \quad (14)$$

where $\boldsymbol{W}_5 \in \mathbb{R}^{1\times d}$ and $\boldsymbol{b}_5 \in \mathbb{R}^{1\times 1}$ are trainable parameters.

3.4 Optimization

Following previous works [Dettmers *et al.*, 2018; Vashishth *et al.*, 2019], for each triple $(h,\ r,\ t)$ in the KG, we first convert it to queries $(h,\ r,\ ?)$ and $(t,\ r^{-1},\ ?)$. Taking $(h,\ r,\ ?)$ as an example, we minimize the binary cross-entropy loss to optimize the model parameters, as follows:

$$\mathcal{L} = -\log\sigma\left(f\left(h,\ r,\ t\right)\right) - \sum_{j=1}^{N}\log\left(1 - \sigma\left(f\left(h,\ r,\ t'_j\right)\right)\right) \quad (15)$$

where $(h,\ r,\ t'_j)$ denotes the j-th negative triple by corrupting tail entity in $(h,\ r,\ t)$. Since the MulGA can obtain multi-granularity embedding of all entities in the KG given the query $(h,\ r, ?)$ or $(t,\ r^{-1},\ ?)$, we set $N = |\mathcal{E}| - 1$ in Eq. 15.

Type	Methods	WN18RR			FB15K-237			NELL-995		
		MRR	Hits@1(%)	Hits@10(%)	MRR	Hits@1(%)	Hits@10(%)	MRR	Hits@1(%)	Hits@10(%)
Triple	TransE [2013]	0.226	N/A	53.2	0.294	N/A	46.5	0.424	34.0	55.5
	ConvE [2018]	0.427	39.2	49.8	0.325	23.7	50.1	0.511	44.6	61.9
	RotatE [2019]	0.477	42.8	57.1	0.337	24.1	53.3	0.508	44.8	60.8
	QuatE [2019]	0.480	44.0	55.1	0.350	25.6	53.8	0.533	46.6	64.3
	DistMult [2015]	0.430	39.0	49.0	0.241	15.5	41.9	0.485	40.1	61.0
Relation Path	MINERVA [2018]	0.448	41.3	51.3	0.293	21.7	45.6	0.513	41.3	63.7
	Neural LP [2017]	0.435	37.1	56.6	0.252	18.9	37.5	N/A	N/A	N/A
	DRUM [2019]	0.486	42.5	58.6	0.343	25.5	51.6	0.532	46.0	66.2
	RNNLogic [2020]	0.483	44.6	55.8	0.344	25.2	53.0	0.416	36.3	47.8
	NBFNet [2021]	0.551	49.7	66.6	0.415	32.1	59.9	0.525	45.1	63.9
	RED-GNN [2022]	0.535	48.9	62.4	0.369	27.7	55.2	0.547	48.3	65.5
	A*Net [2023]	0.549	49.5	65.9	0.411	32.1	58.6	N/A	N/A	N/A
Subgraph	R-GCN [2018]	0.402	34.5	49.4	0.273	18.2	45.6	0.12	8.20	18.8
	CompGCN [2019]	0.479	44.3	54.6	0.355	26.4	53.5	0.463	38.3	59.6
	DPMPN [2019]	0.482	44.4	55.8	0.369	28.6	53.0	0.513	45.2	61.5
	DisenKGAT [2021]	0.486	44.1	57.8	0.368	27.5	55.3	0.549	46.5	66.0
	MulGA	**0.573**	**52.3**	**67.1**	**0.422**	**33.2**	**60.0**	**0.564**	**50.4**	**66.9**

Table 1: Experimental results in the transductive setting. The bold scores indicate the best results and underlined ones indicate the second best results. N/A means unavailable results. Results of compared methods are from [Zhang and Yao, 2022; Zhu *et al.*, 2023].

4 Experiments

4.1 Experimental Setup

Datasets and Baselines

For transductive setting, WN18RR [Dettmers *et al.*, 2018], FB15K-237 [Toutanova *et al.*, 2015], and NELL-995 [Das *et al.*, 2018] are used to evaluate the performance of MulGA. For inductive setting, we use the inductive benchmark datasets provided in GraIL [Teru *et al.*, 2020]. To evaluate the performance, we chose three categories totaling 16 baselines for transductive setting and 8 baselines for inductive setting.

Evaluation Metrics

In transductive and inductive settings, we report mean reciprocal rank (MRR), Hits@1, and Hits@10 following the filtered setting as described in [Bordes *et al.*, 2013; Zhu *et al.*, 2023].

4.2 Main Results

Comparison for Performance

Table 1 reports results of MulGA and all baselines in the transductive setting. As expected, models that only exploit the single granularity structural feature in KGs do not show competitive performance. Conversely, MulGA consistently achieves the best performance on most benchmarks, demonstrating the effectiveness of the unified way to obtain different granularity embeddings and our proposed merging technique. Notably, on WN18RR, FB15K-237, and NELL-995, MulGA improves MRR by 9.3%, 7.2%, and 3.1% compared with the best triple-based methods, 2.2%, 0.7%, and 1.7% compared with the best relation path-based methods, and 8.7%, 5.3%, and 1.5% compared with the best subgraph-based methods.

In the inductive setting, the experimental results are presented in Table 2. MulGA achieves the best performances on most datasets. Overall, MulGA achieves remarkable improvements over the strongest baselines *w.r.t.* MRR by 0.1%-8.2% in the inductive setting. Such results indicate that the utilization of different granularity features in triples, relation paths, and subgraphs is helpful for MulGA to learn similar structural information to generalize unseen entities in target KGs. Similarly, MulGA has a significant improvement on Hits@1, which means MulGA can better capture semantic correlation between queries and various structures in KGs. Specifically, MulGA improves Hits@1 by 0.1%-8.3% compared with the second best results. Additionally, WN18RR (V1-V4) inductive dataset contains many empty subgraphs [Mai *et al.*, 2021], *i.e.*, the enclosing subgraphs of the head and candidate entities at a given hop only have the current triple and no other valid edges. This could be the reason why the results of MulGA on the WN18RR dataset are slightly lower than A*Net. Although the sparse KG can reduce different granularity information significantly, the performance of MulGA is competitive with the previous state-of-the-art methods in WN18RR (V1-V4). Moreover, methods explicitly extracting subgraphs and surrounding neighboring relation paths, such as SNRI, suffer from scalability drawbacks as the sizes of KGs increase(FB15K-237 (V3) (V4) and NELL-995 (V3) (V4)). Expansion of the graph scale results in a rapid rise in CPU memory usage, ultimately leading to an out-of-memory (OOM) breakdown.

Comparison for Efficiency

We further select seven competitive models to compare the efficiency with MulGA and results are shown in Table 3 and Figure 3. We can draw three conclusions: **(1)** MulGA achieves a trade-off between performance and efficiency across diverse graph scales, *i.e.*, from large-scale transductive to small-scale but challenging inductive reasoning tasks. **(2)** The inference time of MulGA is notably shorter than subgraph-based methods such as GraIL and SNRI, which require explicit enclosing subgraph extraction. This demonstrates the advantage of utilizing connectivity subgraphs in enhancing computational efficiency. **(3)** The wall time per epoch of MulGA is not particularly fast, mainly because it requires dynamically propagating on connectivity subgraphs with different sizes, which is harder to parallel on GPUs. However, MulGA can efficiently utilize

Metrics	Methods	WN18RR				FB15K-237				NELL-995			
		V1	V2	V3	V4	V1	V2	V3	V4	V1	V2	V3	V4
MRR	RuleN [2018]	0.717	0.686	0.410	0.638	0.373	0.432	0.425	0.398	0.728	0.443	0.310	0.360
	Neural LP [2017]	0.649	0.635	0.361	0.628	0.325	0.389	0.400	0.396	0.610	0.361	0.367	0.261
	DRUM [2019]	0.666	0.646	0.380	0.627	0.333	0.395	0.402	0.410	0.628	0.365	0.375	0.273
	GraIL [2020]	0.627	0.625	0.323	0.553	0.279	0.276	0.251	0.227	0.481	0.297	0.322	0.262
	SNRI [2022]	0.576	0.539	0.305	0.486	0.186	0.227	OOM	OOM	0.363	0.20	OOM	OOM
	NBFNet [2021]	0.741	0.704	0.452	0.641	0.422	0.514	0.476	0.453	0.584	0.410	0.425	0.287
	RED-GNN [2022]	0.701	0.690	0.427	0.651	0.369	0.469	0.445	0.442	0.637	0.419	0.436	0.363
	A*Net [2023]	0.727	0.704	0.441	**0.661**	0.457	0.510	0.476	0.466	0.789	0.439	0.489	0.311
	MulGA	**0.747**	**0.705**	**0.476**	0.655	**0.485**	**0.538**	**0.492**	**0.472**	**0.812**	**0.525**	**0.510**	**0.429**
Hits@1(%)	RuleN [2018]	67.0	63.0	37.6	59.8	30.9	33.1	32.1	30.1	66.0	34.2	23.1	26.1
	Neural LP [2017]	59.2	57.5	30.4	58.3	24.3	28.6	30.9	28.9	50.0	24.9	26.7	13.7
	DRUM [2019]	61.3	59.5	33.0	58.6	24.7	28.4	30.8	30.9	50.0	27.1	26.2	16.3
	GraIL [2020]	55.4	54.2	27.8	44.3	20.5	20.2	16.5	14.3	42.5	19.9	22.4	15.3
	SNRI [2022]	47.9	44.8	23.4	39.9	10.7	14.4	OOM	OOM	18.5	12.8	OOM	OOM
	NBFNet [2021]	69.5	65.1	39.2	60.8	33.5	42.1	38.4	36.0	50.0	27.1	26.2	23.3
	RED-GNN [2022]	65.3	63.3	36.8	60.6	30.2	38.1	35.1	34.0	52.5	31.9	34.5	25.9
	A*Net [2023]	68.2	64.9	38.6	**61.6**	38.1	41.9	38.9	36.5	72.9	34.0	39.4	26.7
	MulGA	**70.8**	**65.2**	**42.4**	61.1	**40.2**	**44.4**	**39.2**	**37.2**	**75.0**	**42.5**	**42.2**	**34.1**
Hits@10(%)	RuleN [2018]	79.8	69.4	40.7	68.1	48.0	63.0	60.6	58.5	87.0	60.5	46.7	52.3
	Neural LP [2017]	77.2	74.9	47.6	70.6	46.8	58.6	57.1	59.3	87.1	56.4	57.6	53.9
	DRUM [2019]	72.3	68.8	31.2	64.4	40.0	49.6	44.3	46.7	79.5	62.5	56.4	50.7
	GraIL [2020]	76.0	77.6	40.9	68.7	42.9	42.4	42.4	38.9	56.5	49.6	51.8	50.6
	SNRI [2022]	75.0	69.8	43.2	65.2	33.2	38.8	OOM	OOM	55.5	33.3	OOM	OOM
	NBFNet [2021]	82.6	79.8	**56.8**	69.4	57.4	68.5	63.7	62.7	79.5	63.5	60.6	**59.1**
	RED-GNN [2022]	79.9	78.0	52.4	72.1	48.3	62.9	60.3	62.1	86.6	60.1	59.4	55.6
	A*Net [2023]	81.0	**80.3**	54.4	**74.3**	58.9	67.2	62.9	64.5	90.3	61.2	67.3	37.5
	MulGA	**82.7**	80.1	56.0	73.6	**62.2**	**70.7**	**66.2**	**65.2**	**93.5**	**70.9**	**67.9**	57.1

Table 2: Experimental results in the inductive setting. OOM denotes no answer due to out-of-memory (1 TiB RAM). Except for SNRI and RuleN which we reproduce, the results of other compared methods are from [Zhang and Yao, 2022; Zhu *et al.*, 2023].

Methods	FB15K-237			Methods	FB15K-237 (V1)		
	WT(m)	IT(m)	P(M)		WT(s)	IT(s)	P(M)
RotatE	**0.2**	2.2	29.32	NBFNet	5.0	**0.02**	2.38
NBFNet	42.9	4.7	3.10	RED-GNN	**1.9**	1.4	0.05
RED-GNN	384.1	94.4	**0.12**	A*Net	3.8	**0.02**	2.31
A*Net	21.7	2.0	3.10	GraIL	58.4	4.63 h	**0.03**
DisenKGAT	0.8	**0.2**	18.21	SNRI	69.4	7.96 h	0.20
MulGA	50.5	6.3	3.05	MulGA	6.1	0.06	2.32

Table 3: Efficiency of competitive models in an A100 GPU (80GB), which involves three aspects: wall time per epoch (WT), inference time (IT), and the number of free parameters (P).

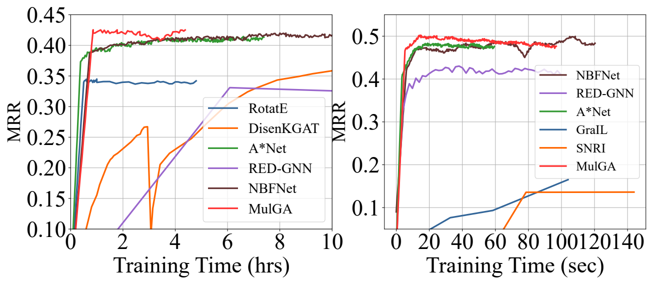

Figure 3: Validation MRR *w.r.t.* training time for competitive models during training on FB15K-237 (left) and FB15K-237 (V1) (right).

the existing structure of KGs and thus has superior convergence speeds in both transductive and inductive tasks.

4.3 Ablation Study

Effect of Granularity Embeddings

We investigate the effect of different granularity embeddings on MulGA performance and results are presented in Table 4.

We make the following observations: **(1)** Reasoning only with triple granularity embedding does not yield superior results. This can be attributed to the fact that the correct entities are not located in the one-hop neighborhood of the query entity. To make accurate predictions, it is necessary to leverage more structural information between the target entity and the query entity. **(2)** Compared with the best results using only a single granularity, MulGA improves MRR by 4.2%-26.4% on WN18RR, 1.3%-12.5% on WN18RR (V1), 3.9%-18.2% on FB15K-237 (V1), and 3.5%-19.0% on NELL-995 (V1), respectively. **(3)** For relation path granularity, we observe that the longer the relation path, the better the performance on the WN18RR and FB15K-237 (V1). Nevertheless, on WN18RR (V1) and NELL-995 (V1), the best performance does not always correspond to the longest relation path. The main reason is that stacking too many GNN layers could cause over-smoothing, *i.e.*, there is no obvious difference between the positive and negative entities.

Furthermore, we need to determine whether the over-smoothing leads to decreased performance on NELL-995 (V1) and WN18RR (V1). To quantitatively measure the smoothness of representation, we employ the Mean Average Distance

Methods	WN18RR			WN18RR (V1)			FB15K-237 (V1)			NELL-995 (V1)		
	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10
MulGA-t	0.309	20.3	38.1	0.622	60.9	63.3	0.303	27.8	33.4	0.716	60.0	91.5
MulGA-rp-2	0.483	44.2	58.7	0.684	65.7	72.9	0.349	30.7	37.6	0.719	62.0	91.0
MulGA-rp-3	0.499	45.4	59.1	0.705	66.5	78.6	0.419	35.9	51.2	0.736	62.0	$\underline{93.0}$
MulGA-rp-4	0.505	46.5	60.3	$\underline{0.734}$	$\underline{69.4}$	80.9	0.438	36.6	53.9	$\underline{0.777}$	$\underline{69.5}$	92.5
MulGA-rp-5	0.510	46.3	62.5	0.726	67.0	$\underline{81.9}$	0.434	$\underline{37.6}$	54.4	0.678	62.5	76.0
MulGA-rp-6	$\underline{0.531}$	$\underline{48.2}$	$\underline{63.4}$	0.725	67.6	$\underline{81.9}$	$\underline{0.446}$	36.8	$\underline{57.6}$	0.622	57.5	69.5
MulGA	**0.573**	**52.3**	**67.1**	**0.747**	**70.8**	**82.7**	**0.485**	**40.2**	**62.2**	**0.812**	**75.0**	**93.5**

Table 4: Experimental results of different granularity embedding. -t denotes the results of reasoning only with the triple granularity embedding (*i.e.*, $\boldsymbol{e}_{\mathcal{G}_h^n} = \boldsymbol{e}_{\mathcal{P}_h^1}$), -rp-$i$ denotes reasoning only with the relation path granularity embedding of length i (*i.e.*, $\boldsymbol{e}_{\mathcal{G}_h^n} = \boldsymbol{e}_{\mathcal{P}_h^i}$).

Methods	WN18RR (V1)		NELL-995 (V1)	
	Validation	Test	Validation	Test
MulGA-t	0.999	0.998	0.999	0.982
MulGA-rp-2	0.997	0.993	0.988	0.789
MulGA-rp-3	0.987	0.984	0.926	0.543
MulGA-rp-4	0.964	0.973	0.825	0.646
MulGA-rp-5	0.916	0.957	0.661	0.367
MulGA-rp-6	0.832	0.938	0.538	0.645
MulGA	0.901	1.081	0.924	1.110

Table 5: The results of $\overline{\mathrm{MAD}}$ on NELL-995 (V1), where the range of values is in $(0, 2]$. A smaller value represents a higher degree of smoothness and a greater tendency to cause over-smoothing.

Methods	WN18RR (V1)		FB15K-237 (V1)		NELL-995 (V1)	
	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10
MulGA-sum	0.737	81.9	0.435	55.1	0.761	84.6
MulGA-max	0.732	81.9	$\underline{0.457}$	56.8	0.730	85.3
MulGA-mean	0.735	82.2	0.449	56.8	$\underline{0.765}$	$\underline{90.6}$
MulGA-concat	$\underline{0.737}$	$\underline{82.2}$	0.442	$\underline{57.1}$	0.754	82.3
MulGA-SAM	**0.747**	**82.7**	**0.485**	**62.2**	**0.812**	**93.5**

Table 6: The experimental results of different merging operations.

(MAD) [Chen *et al.*, 2020a], which depicts the smoothness by calculating the mean of average distances between two nodes. For all queries in the validation and test set, we calculate the average of the sum of MAD, namely $\overline{\mathrm{MAD}}$, between the positive entity and other candidate entities, and results are presented in Table 5. Since training is performed on $\mathcal{G}_S$, the smoothing on validation set is more representative. Our findings reveal that the smoothness of MulGA-rp-i gradually increases as the relation path length i grows, raising the risk of over-smoothing. Moreover, Table 5 indicates that applying proposed self-attention mechanism can alleviate over-smoothing. Specifically, on NELL-995 (V1) test set, the $\overline{\mathrm{MAD}}$ of MulGA-rp-6 is 0.645, whereas, the $\overline{\mathrm{MAD}}$ of MulGA is 1.11.

Effect of Merging Operations

Regarding the merging process, we compare the common operations of *sum*, *max*, *mean*, and *concat* with our proposed SAM mechanism. The results, shown in Table 6, show that compared with the best results from the common merging operations, MRR is improved by 1.0%, 2.8%, and 4.7% on WN18RR (V1), FB15K-237 (V1) and NELL-995 (V1) datasets, respectively, when using our proposed SAM mechanism. The substantial improvement in performance with SAM mechanism is the result of considering the unique features in each granularity through adaptive weight assignment.

5 Related Work

We give a brief overview of different types of KGR methods. **(1) KGE-Based (Triple-based) Methods:** Such methods generally treat relations as operations between head and tail entities in a vector space to learn entity-specific embeddings, such as addition for TransE [Bordes *et al.*, 2013], rotation for RotatE [Sun *et al.*, 2019], and other complex transformations [Zhang *et al.*, 2019]. **(2) Relation Path-Based Methods:** Relation paths have richer local structural features than triples, and can provide more clues for KGR to infer missing triples. Relation path-based methods use multi-hop relation paths between head and tail entities directly. NBFNet [Zhu *et al.*, 2021], RED-GNN [Zhang and Yao, 2022], and A*Net [Zhu *et al.*, 2023] use the generalized Bellman-ford algorithm [Bellman, 1958] to compute relation path representations between two entities to learn entity-independent representation. Some relation path-based KGR models also learn logical rules as a form of generalized relation path between two entities [Meilicke *et al.*, 2018; Yang *et al.*, 2017; Sadeghian *et al.*, 2019; Qu *et al.*, 2020]. **(3) Subgraph-Based Methods:** Some subgraph-based methods extend traditional GNNs by further considering the relational information in the whole KGs. RGCN [Schlichtkrull *et al.*, 2018] aggregates relation-specific transformation matrices with neighborhood information. CompGCN [Vashishth *et al.*, 2019] instead utilizes composition operators to jointly embed entities and relations in KGs. Moreover, GraIL [Teru *et al.*, 2020], SNRI [Xu *et al.*, 2022], and CoMPILE [Mai *et al.*, 2021] strive to explicitly extract enclosing subgraphs, utilizing them to derive entity-independent relational semantics.

6 Conclusion

In this paper, we propose MulGA, which can efficiently produce high-quality multi-granularity embeddings of all entities for KGR. This is realized by adopting a hierarchical approach in modeling KGs structures via connectivity subgraphs, then proposing a multi-granularity message-passing module to efficiently obtain different granularity embeddings, and finally designing the SAM mechanism to facilitate adaptively assignment of weights to different granularity embeddings. Empirical results show that MulGA achieves new state-of-the-art KGR performance in both transductive and inductive settings.

Acknowledgments

We thank the reviewers for their insightful comments. This work was supported by National Science Foundation of China (Grant Nos.62376057) and the Start-up Research Fund of Southeast University (RF1028623234). All opinions are of the authors and do not reflect the view of sponsors.

References

[Auer *et al.*, 2007] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *ISWC*, 2007.

[Bellman, 1958] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16:87–90, 1958.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[Chen *et al.*, 2020a] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI*, 2020.

[Chen *et al.*, 2020b] Xiaojun Chen, Shengbin Jia, and Yang Xiang. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, 141:112948, 2020.

[Das *et al.*, 2018] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *ICLR*, 2018.

[Dettmers *et al.*, 2018] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, 2018.

[Dong *et al.*, 2014] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD*, 2014.

[Dong, 2019] Xin Luna Dong. Building a broad knowledge graph for products. In *ICDE*, 2019.

[Huang *et al.*, 2019] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *WSDM*, 2019.

[Lee *et al.*, 2019] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019.

[Li *et al.*, 2022] Guozheng Li, Xu Chen, Peng Wang, Jiafeng Xie, and Qiqing Luo. Fastre: Towards fast relation extraction with convolutional encoder and improved cascade binary tagging framework. In *IJCAI*, 2022.

[Liu *et al.*, 2023] Jiajun Liu, Peng Wang, Ziyu Shang, and Chenxiao Wu. Iterde: an iterative knowledge distillation framework for knowledge graph embeddings. In *AAAI*, 2023.

[Mai *et al.*, 2021] Sijie Mai, Shuangjia Zheng, Yuedong Yang, and Haifeng Hu. Communicative message passing for inductive relation reasoning. In *AAAI*, 2021.

[Meilicke *et al.*, 2018] Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion. In *ISWC*, 2018.

[Noy *et al.*, 2019] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: Lessons and challenges. *Commun. ACM*, 62(8):36–43, 2019.

[Qu *et al.*, 2020] Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. Rnnlogic: Learning logic rules for reasoning on knowledge graphs. In *ICLR*, 2020.

[Sadeghian *et al.*, 2019] Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. Drum: End-to-end differentiable rule mining on knowledge graphs. In *NeurIPS*, 2019.

[Saxena *et al.*, 2020] Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *ACL*, 2020.

[Scarselli *et al.*, 2008] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[Schlichtkrull *et al.*, 2018] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.

[Shang *et al.*, 2023] Ziyu Shang, Peng Wang, Yuzhang Liu, Jiajun Liu, and Wenjun Ke. Askrl: An aligned-spatial knowledge representation learning framework for open-world knowledge graph. In *ISWC*, 2023.

[Sun *et al.*, 2019] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2019.

[Teru *et al.*, 2020] Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In *ICML*, 2020.

[Toutanova *et al.*, 2015] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, 2015.

[Vashishth *et al.*, 2019] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *ICLR*, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[Vrandečić, 2012] Denny Vrandečić. Wikidata: A new platform for collaborative data collection. In *WWW*, 2012.

[Wang *et al.*, 2017] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[Wang *et al.*, 2018] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *CIKM*, 2018.

[Wang *et al.*, 2021] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. Learning intents behind interactions with knowledge graph for recommendation. In *WWW*, 2021.

[Wu *et al.*, 2021] Junkang Wu, Wentao Shi, Xuezhi Cao, Jiawei Chen, Wenqiang Lei, Fuzheng Zhang, Wei Wu, and Xiangnan He. Disenkgat: knowledge graph embedding with disentangled graph attention network. In *CIKM*, 2021.

[Xu *et al.*, 2019] Xiaoran Xu, Wei Feng, Yunsheng Jiang, Xiaohui Xie, Zhiqing Sun, and Zhi-Hong Deng. Dynamically pruned message passing networks for large-scale knowledge graph reasoning. In *ICLR*, 2019.

[Xu *et al.*, 2022] Xiaohan Xu, Peng Zhang, Yongquan He, Chengpeng Chao, and Chaoyang Yan. Subgraph neighboring relations infomax for inductive link prediction on knowledge graphs. In *IJCAI*, 2022.

[Yang *et al.*, 2015] Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*, 2015.

[Yang *et al.*, 2017] Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *NIPS*, 2017.

[Zhang and Yao, 2022] Yongqi Zhang and Quanming Yao. Knowledge graph reasoning with relational digraph. In *WWW*, 2022.

[Zhang *et al.*, 2019] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. Quaternion knowledge graph embeddings. In *NeurIPS*, 2019.

[Zhang *et al.*, 2022] Denghui Zhang, Zixuan Yuan, Hao Liu, Hui Xiong, et al. Learning to walk with dual agents for knowledge graph reasoning. In *AAAI*, 2022.

[Zhu *et al.*, 2021] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. In *NeurIPS*, 2021.

[Zhu *et al.*, 2023] Zhaocheng Zhu, Xinyu Yuan, Mikhail Galkin, Sophie Xhonneux, Ming Zhang, Maxime Gazeau, and Jian Tang. A* net: A scalable path-based reasoning approach for knowledge graphs. In *NeurIPS*, 2023.