Message Function Search for Knowledge Graph Embedding

Shimin DI
Computer Science and Engineering, HKUST
Hong Kong SAR, China
sdiaa@connect.ust.hk

Lei CHEN
Data Science and Analytics, HKUST(GZ)
Guangzhou, China
leichen@ust.hk

ABSTRACT

Recently, many promising embedding models have been proposed to embed knowledge graphs (KGs) and their more general forms, such as n-ary relational data (NRD) and hyper-relational KG (HKG). To promote the data adaptability and performance of embedding models, KG searching methods propose to search for suitable models for a given KG data set. But they are restricted to a single KG form, and the searched models are restricted to a single type of embedding model. To tackle such issues, we propose to build a search space for the message function in graph neural networks (GNNs). However, it is a non-trivial task. Existing message function designs fix the structures and operators, which makes them difficult to handle different KG forms and data sets. Therefore, we first design a novel message function space, which enables both structures and operators to be searched for the given KG form (including KG, NRD, and HKG) and data. The proposed space can flexibly take different KG forms as inputs and is expressive to search for different types of embedding models. Especially, some existing message function designs and some classic KG embedding models can be instantiated as special cases of our space. We empirically show that the searched message functions are data-dependent, and can achieve leading performance on benchmark KGs, NRD, and HKGs.

CCS CONCEPTS

• **Information systems** → *Web searching and information discovery*; • **Computing methodologies** → **Knowledge representation and reasoning**; **Machine learning algorithms**.

KEYWORDS

graph neural networks, knowledge graph embedding

ACM Reference Format:
Shimin DI and Lei CHEN. 2023. Message Function Search for Knowledge Graph Embedding. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3543507.3583546

1 INTRODUCTION

Web-scale knowledge graphs (KGs) [2] have attracted much attention due to their widespread existence, which have promoted a series of web applications such as recommendation system [8] and question answering [33]. Generally, KGs store and organize human knowledge with the form of binary fact $r(e_1, e_2)$ that represents the relation r between entities e_i, e.g., `playIn(Leonard,StarTrek1)`. More recently, researchers observe that binary facts are only a part of knowledge bases. For example, more than 30% of entities in Freebase [6] involve facts that contain more than 2 entities [58]. Thus, research communities start to learn more general forms of KGs, such as n-ary relational data (NRD) $S = \{r(e_1, \ldots, e_n)\}$ [19, 31] (e.g., `playInCharacter(Leonard, StarTrek1, Spock)`) and hyper-relational KG (HKG) $S = \{r(e_1, e_2, \{(r_{o_j}, e_j)\}_{j=3}^{n})\}$ [15, 40] (e.g., `playInCharacter(Leonard,StarTrek1,(character:Spock))`).

To manipulate web-scale KG/NRD/HKG, various promising embedding models [35, 39, 53] have been proposed to encode sets of relations R and entities E into d-dimensional vector space $\boldsymbol{R} \in \mathbb{R}^{|R| \times d}, \boldsymbol{E} \in \mathbb{R}^{|E| \times d}$, such as geometric models [7, 43, 58], neural network models [19, 40, 49], bilinear models [25, 47, 61], and more general tensor decomposition models [3, 28, 31]. However, these methods follow the classical way to design a universal model for different data sets. But due to the diversity of data sets [53], an embedding model that performs well on one data set may not adapt well to another one [39, 68]. To tackle this data-aware issue, KG searching models [11, 42, 68] promote the adaptability of embedding models by searching appropriate models for the given data.

Despite the success of KG searching models [11, 42, 67, 68], there are two major limitations of them. First, existing searching models are strictly restricted to one KG form. They cannot be applied or extended to diverse KG forms. This obviously limit the data adaptability of searching models. Second, the search space of current searching models is only based on tensor decomposition models, i.e., only tensor decomposition models can be searched. This design may limit the performance of searched models since there are many other types of promising embedding models. Therefore, a more flexible and expressive search space is needed to search for embedding models on diverse KG forms.

Recently, some pioneer embedding models [15, 41, 60] have designed domain-specific message functions in powerful graph neural networks (GNNs) [20, 27, 29] by capturing the interaction between entities and relations (see Fig. 1). Inspired by their success, we may be able to build a flexible and expressive search space on the message functions. Unfortunately, it is a non-trivial task because existing message function designs are rigid. First, in GNNs for KG embedding works, their message functions manually design and fix the structures and operators, which are inflexible to handle diverse KG forms and not conducive to handling complex relational patterns in different KG data sets. As shown in Fig. 1, their message functions fix the input forms, such as CompGCN [49] for KG, G-MPNN [60] for NRD, StarE [15] for HKG. Moreover, under a KG form, relations usually have distinct patterns in different KG

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583546

Table 1: Overview of the existing GNN-based works. NC, GC, LP, RP denote node classification, graph classification, link prediction, relation prediction, respectively. "Data-aware" measures whether the structures and operators of message function change for different inputs. $DP(\cdot)$ is dropout; $BN(\cdot)$ is batch normalization; $\bar{o}(\cdot)$ summarizes mixed operations [70]; $\boldsymbol{h}_i^K$ denotes the output from i-th operator of O in K-th layer of the message function (see Sec. 3.1); other notations can be checked in Sec. 2.

Type	Model	Scenarios				Message Function	
		Task	# Edge Types	# Edge Length	KG Form	Data-aware	Function
GNNs Searching	You et al. [63]	NC/GC/LP	= 1	= 2	N/A	×	$DP(BN(\boldsymbol{W}\boldsymbol{e}_j + \boldsymbol{b}))$
	GraphNAS [16]	NC				×	$a_{ij} concat(\boldsymbol{e}_i, \boldsymbol{e}_j)$
	AGNN [71]					×	$a_{ij}\boldsymbol{W}\boldsymbol{e}_j$
	SANE [70]		≥ 1	= 2		×	$\boldsymbol{W}\bar{o}(\{\boldsymbol{e}_j\}_{e_j \in N(e_i)})$
	NAS-GCN [24]	GC				×	$a_{ij} MLP(\boldsymbol{h}_{ij})\boldsymbol{e}_j$
	AutoGEL [54]	NC/GC/LP			KG	×	$\boldsymbol{W}_{(r)}\phi(\boldsymbol{e}_j, \boldsymbol{r})$
GNNs for KGs	R-GCN [41]	LP/RP	≥ 1	= 2	KG	×	$\boldsymbol{W}_r \boldsymbol{e}_j$
	CompGCN [49]					×	$\boldsymbol{W}_{\lambda(r)}\phi(\boldsymbol{e}_j, \boldsymbol{r})$
	G-MPNN [60]		≥ 1	≥ 2	NRD	×	$\boldsymbol{r}_{P(e)} * \prod_i \boldsymbol{p}_{e_i} * \boldsymbol{e}_i$
	StarE [15]				HKG	×	$\boldsymbol{W}_{\lambda(r)}\phi_r(\boldsymbol{e}_o, \gamma(\boldsymbol{r}, \boldsymbol{h}_r))$
Ours	MSeaKG	LP/RP	≥ 1	≥ 2	KG/NRD/HKG	√	$MLP\&concat(\{\boldsymbol{h}_i^K\}_{i=1}^{\lvert O \rvert})$

data sets, which brings difficulties for the fixed message function to adapt to different KG data sets. For example, the message function of G-MPNN [60] adopts the inner product way like DistMult [61] to compute the correlation between entities and relations, which has been proven to only cover symmetric relations [25]. Its performance may not be good if there are many non-symmetric relations. Second, existing GNN searching methods [69] more focus on searching connections between GNN layers and other GNN functions. Their message functions are also fixed, thereby incapable of handling different KG data sets. And most of them usually ignore edge embeddings to represent relations, which cannot capture complex correlation between entities and relations. We summarized existing GNNs for KG embedding and GNN searching models in Tab. 1 in terms of allowed KG forms and message functions.

In summary, the search space of KG searching methods is inflexible to handle diverse KG forms and is limited to tensor decomposition models, while rigid message functions in existing GNN works are not conducive to handling different KG forms and data sets. Therefore, to improve the adaptability and performance of KG embedding models, we propose ***M***essage function ***SEA***rch for the given ***KG*** form (including KG, NRD and HKG) and data, named as MSeaKG. In this paper, we propose to build a flexible and expressive search space based on the message function. More concretely, we first propose a flexible space that allows diverse KG forms as inputs. Then, we identify the necessary computation operators that are domian-specific designs for KG and search the structures that interact these operators in the message function. Not only various types of embedding models can be instantiated by message functions with different structures and operators, but also the searched message function can capture the relational patterns in the given data. Besides, we also search other GNN components (e.g., aggregation function) for pursuing more performance improvements. Finally, we formulate the discrete GNN models with probabilistic modelings to enable an efficient search algorithm working on our scenario. The main contributions are listed as:

- MSeaKG proposes a novel search space for KG embedding models. As shown in Fig. 3 (a), the space allows different KG forms (including KG/NRD/HKG) as inputs and covers multiple types of KG embedding models (including tensor/GNN/geometric models), while previous KG searching methods are specifically designed for one KG form and only cover one type of embedding models.
- The message function in existing GNN works is rigid, which is incapable of handling different KG forms and data sets. MSeaKG proposes a novel message function space, which enables the structures and operators of message functions being optimized for different KG forms (including KG/NRD/HKG) and data sets.
- We compare MSeaKG with baselines on the link prediction and relation prediction tasks. Experimental results show that MSeaKG can consistently achieve state-of-the-art performance on benchmark KGs, NRD, and HKGs by designing data-aware message functions, which verifies the improvements in adaptability and performance of the KG embedding model.

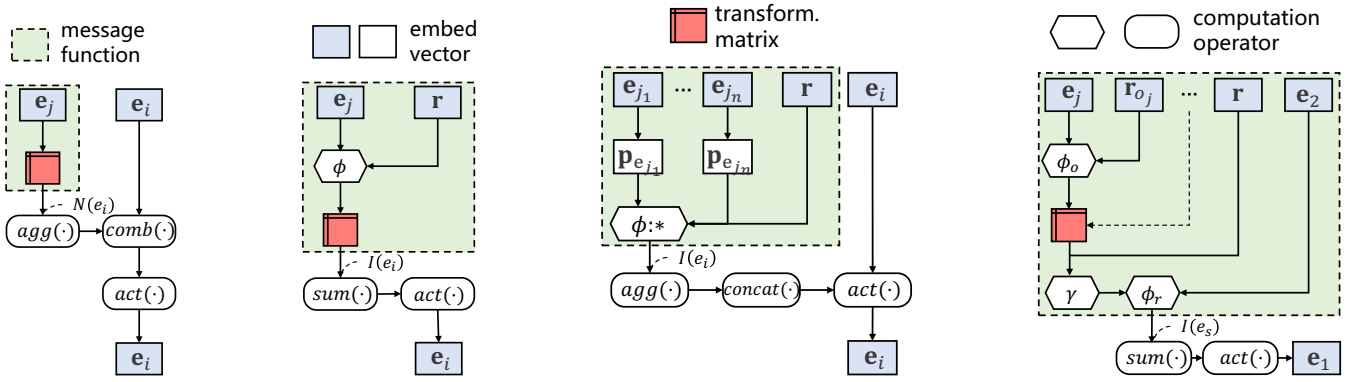
Notations. We denote scalars by lowercase letters (r), vectors by bold lowercase letters ($\boldsymbol{r}$), sets by uppercase letters (R), matrices by bold uppercase letters ($\boldsymbol{R}$). Note that the superscript and subscript are utilized to identify notations instead of indexing elements, such as e_i is i-th entity and $\mathbf{e}_i$ is the vector representation of i-th entity.

2 RELATED WORK

2.1 Graph Neural Network Searching Methods

To avoid manual efforts on neural architecture designs, Neural Architecure Search (NAS) [22, 62] aims to automatically search suitable neural architectures for the given data and task. Generally, search space, search algorithm, and evaluation measurement are three important components of NAS [12]. Search space defines what network architectures in principle should be searched. The search algorithm performs an efficient search over the search space and finds architectures that achieve good performance. Evaluation measurement decides how to evaluate the searched architectures during the search. Classical NAS methods are computationally consuming because candidate architectures are evaluated in a standalone way, i.e., evaluating the performance of architecture after training it to convergence. To reduce the search cost, one-shot NAS [38] proposes weight sharing to share network weights across candidate architectures and evaluate them on the shared weights.

Some pioneer works have explored NAS for GNNs [16, 24, 63, 71]. And the one-shot NAS also has been introduced to search



(a) MPNN for (e_i, e_j) (b) CompGCN for $r(e_i, e_j)$ (c) G-MPNN for $r(e_1, \cdots, e_n)$ (d) StarE for $r(e_1, e_2, \{(r_{o_j}, e_j)\}_{j=3}^{n})$

Figure 1: The framework of several GNN-based works. The green box refers to the message function.

GNN architectures recently [54, 56, 57, 70]. As shown in Fig. 1(a), most GNN searching methods follow the message passing neural networks (MPNNs) [17] to unify two steps in one GNN layer:

$$step1: \ \boldsymbol{m}_i \leftarrow agg(\{mg_c(\boldsymbol{e}_i, \boldsymbol{e}_j)\}_{e_j \in N(e_i)}), \tag{1}$$

$$step2: \ \boldsymbol{e}_i \leftarrow act(comb(\boldsymbol{e}_i, \boldsymbol{m}_i)), \tag{2}$$

where $\boldsymbol{e}_i \in \mathbb{R}^d$ represents the embedding of node e_i, $\boldsymbol{m}_i$ is the intermediate embeddings of e_i gathered from its neighborhood $N(e_i)$. The search space of operators are summarized into:

- **Message Function** $mg_c(\cdot)$: The message function decides the way to gather information from a neighborhood e_j of the center node e_i. The typical message functions in existing GNN searching methods are summarized as $mg_c(\boldsymbol{e}_i, \boldsymbol{e}_j) = a_{ij}\boldsymbol{W}\boldsymbol{e}_j$ [69], where a_{ij} denotes the attention scores between nodes e_i with e_j.
- **Aggregation Function** $agg(\cdot)$: It controls the way to aggregate message from nodes' neighborhood. Usually $agg \in \{sum, max, mean\}$, where $sum(\cdot) = \sum_{e_j \in N(e_i)} mg_c(\boldsymbol{e}_i, \boldsymbol{e}_j)$, $max(\cdot)$ denotes channel-wise maximum, and $mean(\cdot) = \sum_{e_j \in N(e_i)} mg_c(\boldsymbol{e}_i, \boldsymbol{e}_j)/|N(v)|$.
- **Combination Function** $comb(\cdot)$: It determines the way to merge messages between node and its neighbors. $comb$ is usually selected from $\{concat, add, mlp\}$, where $concat(\cdot) = [\boldsymbol{e}_i, \mathbf{m}_i]$, $add(\cdot) = \mathbf{e}_i + \mathbf{m}_i$, and Multi-layer Perceptron $mlp(\cdot) = MLP(\boldsymbol{e}_i + \boldsymbol{m}_i)$.
- **Activation Function** $act(\cdot)$: $[identity, sigmoid, tanh, relu, elu]$ are some of the most commonly used activation functions [16].

Overall, above message functions are still fixed. No matter what the input data is, their structures and operators remain unchanged. Besides, most instantiations of $mg_c(\cdot)$ (see Tab. 1) only learn node embeddings, which cannot encode relations (i.e., edge types). Note that NAS-GCN [24] takes the edge feature $\boldsymbol{h}_{ij}$ between e_i and e_j as input without learning edge embeddings.

2.2 Graph Neural Networks for KG Embedding

KG embedding models [35, 39, 53, 66] have demonstrated their effectiveness in the past decades. Compared with classic models, it may be a better way to adopt different graphs to model several KG forms. For instance, tensor decomposition models [3, 28] represent a KG into a 3-order tensor and decompose tensors into $\boldsymbol{R}$ and $\boldsymbol{E}$. But it is hard to extend them from the case of fixed arity (e.g., KG) to the of mixed arities where facts may have different n in the given data (e.g., $\{r(e_1, e_2), r(e_1, e_2, e_3)\}$). That is because a tensor can only model a set of facts under the same arity [11].

As presented in Tab. 1 and Eq. 1, message functions in classic GNNs simply aggregate messages from adjacent nodes. But in scenarios of KGs, NRD and HKGs, it is important to know the type of edge (relation) that connects several nodes (entities). To capture relations, R-GCN [41] takes the binary fact $r(e_i, e_j)$ as inputs and proposes to model $r \in R$ with $\mathbf{W}_r \in \mathbb{R}^{d \times d}$, which is instantiated as:

$$\boldsymbol{e}_i = act(sum(\{\boldsymbol{W}_r \boldsymbol{e}_j\}_{r(e_i, e_j) \in I(e_i)})), \tag{3}$$

where $I(e_i) = \{r(e_i, e_j) \in S : r \in R, e_j \in E\}$ is the set of facts incident on e_i. But such relation modeling may lead to the over-parameterization issue because $|R|$ could be large. Thus, CompGCN uses the vector $\boldsymbol{r}$ to represent r instead of matrix $\boldsymbol{W}$ [49]:

$$\boldsymbol{e}_i = act(sum(\{\boldsymbol{W}_{\lambda(r)} \phi(\boldsymbol{e}_j, \boldsymbol{r})\}_{r(e_i, e_j) \in I(e_i)})), \ \boldsymbol{r} = \boldsymbol{W}\boldsymbol{r} \tag{4}$$

where $\lambda(r)$ records the directional information of edges. The entity-relation composition operator set $\{sub, mult, corr\}$ is inspired by classical scoring function design in existing KG embedding models, such as element-wise subtraction $sub(\cdot) = \boldsymbol{e}_j - \boldsymbol{r}$ [7], inner product $mult(\cdot) = \boldsymbol{e}_j * \boldsymbol{r}$ [61], circular correlation $corr(\cdot) = \boldsymbol{e}_j \circ \boldsymbol{r}$ [36].

Subsequently, G-MPNN [60] extends GNNs from KGs to NRD. It models NRD $\{r(e_1, \ldots, e_n) : 2 \le n \le N\}$ under the mixed arity case as multi-relational hypergraph. The message function is formed as:

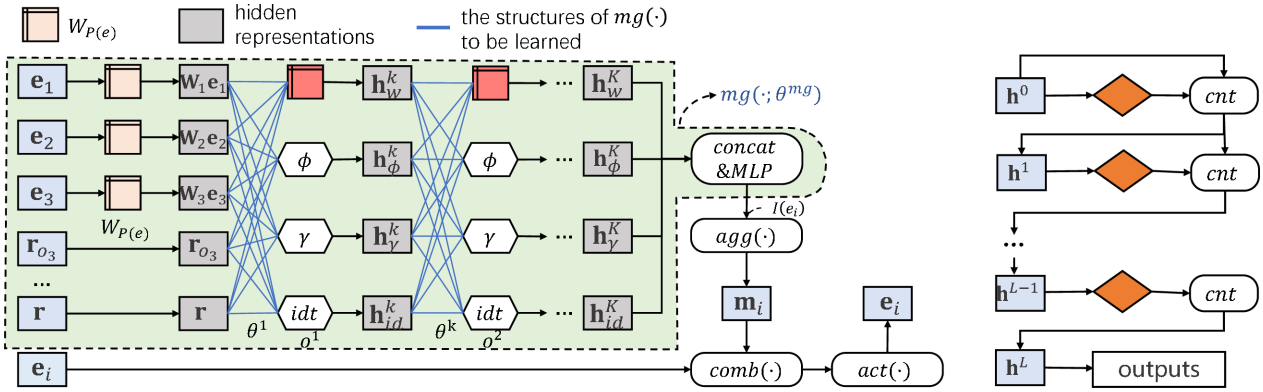
$$\boldsymbol{e}_i = act([\boldsymbol{e}_i, agg(\{\boldsymbol{r}_{s,P(s)} * \prod_{j \in \{1,\ldots,n\}} \boldsymbol{p}_{e_j,s} * \boldsymbol{e}_j\}_{r(e_1,\ldots,e_n) \in I(e_i)})]),$$

where s represents $r(e_1, \ldots, e_n) \in S$, $P(s) : s \to \{1, \ldots, n_p\}$ is a positional mapping ($n_p \le |E|$), and $\boldsymbol{p}_{e_j,s}$ is the positional embedding vector of e_j on fact s to model the positional information.

StarE [15] takes the hyper-relational fact $r(e_1, e_2, \{(r_{o_j}, e_j)\})$ as inputs, where $r(e_1, e_2)$ is the base triplet and r_{o_j} records the role information of entity e_j that plays in this fact. Note that $r(e_1, e_2, \{(r_{o_j}, e_j)\})$ is same as $r(e_1, \ldots, e_n)$ if r_{o_j} is not available. StarE first uses $\boldsymbol{h}_r = \boldsymbol{W} sum(\{\phi_o(\boldsymbol{r}_{o_j}, \boldsymbol{e}_j)\})$ to aggregate information from the role-value pairs, then uses a vector concatenate operator $\gamma(\cdot)$ to form a hyper-relation based on $\boldsymbol{r}$ with $\boldsymbol{h}_r$ as:

$$\boldsymbol{e}_1 = act(sum(\{\boldsymbol{W}_{\lambda(r)} \phi_r(\boldsymbol{e}_2, \gamma(\boldsymbol{r}, \boldsymbol{h}_r))\}_{r(e_1, e_2, \{(r_{o_j}, e_j)\}) \in I(e_1 r)})),$$

where the composition operator $\phi_r(\cdot)$ performs on the hyper-relation with base entity and the update of $\boldsymbol{r}$ in StarE is similar



(a) One GNN layer of MSeaKG's space. The green box refers to the message function space.

(b) The overall space of MSeaKG.

Figure 2: The framework of MSeaHKG. Fig. (a) shows a layer of Fig. (b), including the search for message functions $mg(\cdot;\theta)$ and other functions (e.g., $agg(\cdot), comb(\cdot), act(\cdot)$). The operator cnt enables the connectivity of different layers [24].

to CompGCN. Fig. 1 plots several message functions for different KG forms.

3 MSEAKG

In this section, we first propose a search space that can flexibly take KG/NRD/HKG as input forms and is expressive to search different types of models. Then, we formulate the search problem and leverage an efficient algorithm to solve it.

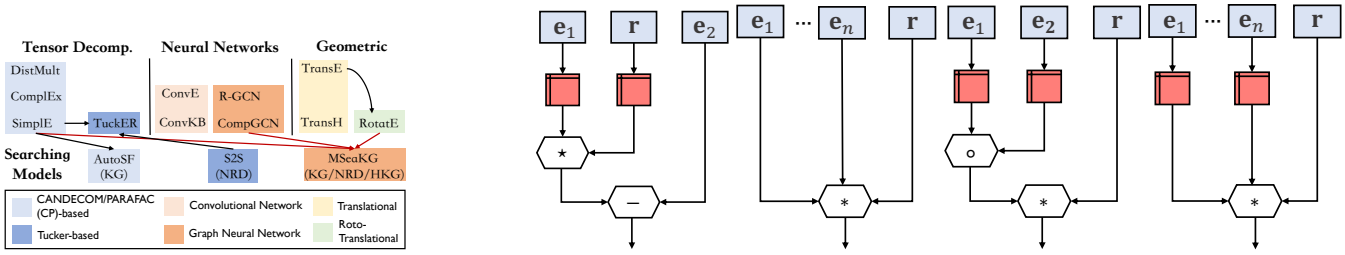
3.1 Search Space Design

As discussed in Sec. 1, dynamicly designing the message function is more conducive to pursuing high empirical performance, because the searched message function can adapt to various KG forms and complex relational patterns in the given KG data. However, the space of existing GNN searching methods is neither data-aware nor applicable to various KG scenarios (see Eq. (1) and Fig. 1(a)). Thus, we focus on the search space design of message functions, which is important to KGs but is neglected by the current works.

From Fig. 1, we can observe that those message functions are mainly different in these two aspects: 1) the operators (e.g., $\boldsymbol{W}, \phi, \gamma$) for computing hidden representations, 2) the structure of message functions that decides how computational operators are connected. For the operator selection, existing works manually tune them on different data sets, such as ϕ in CompGCN [49], ϕ_o, ϕ_r, γ in StarE [15], $agg(\cdot)$ in G-MPNN [60]. Moreover, the structure of message functions for NRD (Fig. 1(c)) and HKG (Fig. 1(d)) tends to be deeper and more complex than those for KGs (Fig. 1(b)). This is because the message function needs to process more entities/roles when facing the facts with higher arity (n is arity of $r(e_1, \cdots, e_n)$). These observations motivate us to build spaces of operators and structures for message function search.

Operator Space. We investigate more about the relationship between operators and relational patterns. Generally, the relational pattern [39] can be represented as a certain correlation among $r(pemu(e_1, \ldots, e_n))$, where $pemu$ denotes the permutation. For example, r is symmetric if $r(e_j, e_i)$ must be true when $r(e_i, e_j)$ is true. r_1 and r_2 are inverse relations if $r_1(e_i, e_j)$ must be true when $r_2(e_j, e_i)$ is true. Therefore, the message function in the search space must be able to handle such correlation in the form of $r(pemu(e_1, \ldots, e_n))$. In the next, we introduce the space of operators and discuss how they deal with $r(pemu(e_1, \ldots, e_n))$.

- Positional Transformation Matrix $\boldsymbol{W}_{P(e)}$: The position of entity in a fact may determine the plausibility of the fact. For example, `isCaptialOf(Beijing,China)` is true while `isCaptialOf (China,Beijing)` is false. G-MPNN [60] utilizes the positional embedding $\boldsymbol{p}_{e,s}$ and $\boldsymbol{r}_{e,P(e)}$ to encode the position of entity e and relation r in different facts, which requires the model complexity $O(d|S|(|E|+N))$. However, the training data set is very sparse in KGs. This may suffer from over-parameterization and make the training insufficient. Instead, we adopt the way to transform one entity e to N possible positions. Let the positional mapping be $P(e) : e \rightarrow \{1, \ldots, N\}$, then the positional matrix is able to transform $\boldsymbol{e}$ to the permutation position in $pemu(\cdot)$ by $\boldsymbol{W}_{P(e)}\boldsymbol{e}$, where $\boldsymbol{W}_{P(e)}$ consumes $O(Nd^2)$ ($|S|, |E| \gg d$ in practice).
- Concatenate Operator $\gamma(\cdot)$: It mainly determines the concatenation way between embedding vectors. We set $O_\gamma = \{concat, mult, wsum\}$ [15], where $wsum$ is the weighted sum. In general, $\gamma(\cdot)$ can concatenate embeddings after the positional transform matrix $\boldsymbol{W}_{P(e)}$, i.e., encoding $pemu(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n)$.
- Role Embedding $\boldsymbol{r}_o$: It is utilized to model the semantic information of entities [32]. For example, the roles in 2nd position of facts `playInCharacter(Zachary,StarTrek,(character:Spock))` and `playInAward(Zachary,StarTrek,(award:BC-BSFC))` are different though other entities are same. Thus, the model should be able to capture the role of candidate entities, e.g., entity `BC-BSFC` is unlikely to be the correlated with relation `playInCharacer` since `BC-BSFC` is semantically similar to `award` instead of `character`. Note that role embedding is an optional choice depending on whether the inputs have role information.
- Composition Operator $\phi(\cdot)$: Following CompGCN [49], we utilize composition operator $\phi(\cdot)$ to capture messages between the node and edge embeddings. Note that ϕ actually encodes the interaction between $\boldsymbol{r}$ and $pemu(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n)$. While CompGCN and StarE empirically selects the most proper $\phi(\cdot)$, we include it into the operator space to search $\phi(\cdot)$. We combine the settings of CompGCN and StarE as $O_\phi = \{sub, mult, corr, rotat\}$ ($rotat$ [43], see others in Sec 2.2). Besides, ϕ not only occurs between base



(a) Taxonomy of KG embedding models. (b) RotatE [43]. (c) n-DistMult [61]. (d) HolE [36]. (e) n-CP [28].

Figure 3: Fig. (a) is motivated by [39], where arrows indicate that the target model can be specialized into the source method. Intuitively, the space of MSeaKG is based on GNNs (orange color), but it can cover tensor/neural network/geometric models and allow KG/NRD/HKG as inputs. Fig. (b)-(d): Several instantiation cases of MSeaHKG message function space. $\circ$ denotes circular correlation $corr(\cdot)$ [36], $*$ denotes inner product $mult(\cdot)$ [61], $-$ denotes the substraction $sub(\cdot)$ [7], $\star$ is from RotatE [44].

relation r with entities, but also captures the correlation between roles r_{o_j} with entities in HKGs.
- Others: (1) The $idt(\boldsymbol{h}) = \boldsymbol{h}$ operation allows inputs to skip one layer in the message function; (2) Unlike $\boldsymbol{W}_{P(e)}$, the transform matrix $\boldsymbol{W}$ processes the hidden representations.

In summary, $\boldsymbol{W}_{P(e)}$ encodes the positional information, which transforms entity embeddings into corresponding positions. Then, the operator $\gamma(\cdot)$ concatenates the entity embeddings after encoding positional information. $\boldsymbol{W}_{P(e)}$ and $\gamma(\cdot)$ are employed to represent $pemu(\cdot)$. The operator $\phi(\cdot)$ computes the interaction between $\boldsymbol{r}$ with $\boldsymbol{e}_i$ to capture the correlation between r and $pemu(\cdot)$.

Structure Space for Message Function Search. Among above components, we fix the role embedding and positional transform matrix $\boldsymbol{W}_{P(e)}$ in the message function (see Fig. 2(a)) at the initial layer. And we include others into the space of message function $O = \{\boldsymbol{W}, \phi, \gamma, idt\}$ for searching. As shown in Fig. 2(a), we denote the node o_i^k as i-th operator of O in k-th layer and $\boldsymbol{h}_i^k$ be the hidden representation outputted by o_i^k. Then we have:

$$\{\boldsymbol{h}_i^0\} = \begin{cases} \{\boldsymbol{W}_{P(e_1)}\boldsymbol{e}_1, \cdots, \boldsymbol{W}_{P(e_n)}\boldsymbol{e}_n, \boldsymbol{r}_{o_3}, \cdots, \boldsymbol{r}_{o_n}, \boldsymbol{r}\} & \text{if } r(e_1, e_2, \{r_{o_j}, e_j\}_{j=3}^n) \\ \{\boldsymbol{W}_{P(e_1)}\boldsymbol{e}_1, \cdots, \boldsymbol{W}_{P(e_n)}\boldsymbol{e}_n, \boldsymbol{r}\} & \text{if } r(e_1, \cdots, e_n) \end{cases}$$

$$\boldsymbol{h}_i^k = o_i^k\big(\{\theta_{ij}^k \boldsymbol{h}_j^{k-1}\}_{j=1}^{|O|}\big), k \in \{1, \ldots, K\} \text{ and } i \in \{1, \ldots, |O|\}, \quad (5)$$

where $\theta_{ij}^k \in \{0, 1\}$ controls the connection between o_i^k with o_j^{k-1}, i.e., $\{\theta_{ij}^k\}$ controls the structure of message functions (see Fig. 2 (c)). Compared with works in Sec. 2.2, it is more flexible to take any data form as inputs. In practice, to handle the facts with mixed arities, we use the maximum arity N as the maximum allowed inputs, i.e., $|\{\boldsymbol{h}_i^0\}| = 2N - 1$ for HKG or $N + 1$ for NRD. For those facts $n < N$, we pad zero embeddings like $\{\boldsymbol{W}_{P(e_1)}\boldsymbol{e}_1, \boldsymbol{W}_{P(e_2)}\boldsymbol{e}_2, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{r}\}$ when $n = 2, N = 4$.

To avoid manual operation selection, we also search for concrete operations of two operators ϕ and γ. Given the operator set O_ϕ and O_γ, let $\theta_i^{\phi_k}, \theta_i^{\gamma_k} \in \{0, 1\}$ records the selection i-th operation $o_i \in O_\phi, O_\gamma$ at k-layer respectively. Then, ϕ and γ perform the computation in Eq. (5) could be $\phi^k(\boldsymbol{h}) = \sum \theta_i^{\phi_k} o_i(\boldsymbol{h})$ and $\gamma^k(\boldsymbol{h}) = \sum \theta_i^{\gamma_k} o_i(\boldsymbol{h})$. Note that $\sum_i \theta_i^{\phi_k} = 1$ and $\sum_i \theta_i^{\gamma_k} = 1$. Let $\boldsymbol{\theta}^{mg} = \{\theta_{ij}^k\} \cup \{\theta_i^{\phi_k}\} \cup \{\theta_i^{\gamma_k}\}$. The message function parameterized by $\boldsymbol{\theta}^{mg}$ is defined as:

$$mg\big(\boldsymbol{r}(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n); \boldsymbol{\theta}^{mg}\big) = MLP\&concat(\{\boldsymbol{h}_i^K\}_{i=1}^{|O|}), \quad (6)$$

Algorithm 1: MSeaKG

Input: Facts $\{\mathcal{S}^{tra}, \mathcal{S}^{val}\}$ from KG/NRD/HKG $\mathcal{G}(E, R, S)$
1: Initialize embedding $\boldsymbol{\omega}$ and GNN architecture parameter $\bar{\boldsymbol{\Theta}}$
2: **while** not converged **do**
3: Sample a GNN architecture X as $X \sim p_{\bar{\boldsymbol{\Theta}}}(X)$;
4: Feed $\mathcal{S}^{tra}$ into the sampled X to compute $\mathcal{L}(X, \boldsymbol{\omega}; \mathcal{S}^{tra})$;
5: Compute $\nabla_{\bar{\boldsymbol{\Theta}}} E[\mathcal{L}(\cdot)]$ (Eq. 9) and $\nabla_{\boldsymbol{\omega}} E[\mathcal{L}(\cdot)]$ (Eq. 10) to update $\boldsymbol{\omega}, \bar{\boldsymbol{\Theta}}$;
6: **end while**
7: Sample 10 architectures $\{X^i\}$ and select the one $X^* = \arg\min_{X^i} \mathcal{L}(X^i, \boldsymbol{\omega}; \mathcal{S}^{val})$
8: Retrain the derived X^* from scratch to obtain the final $\boldsymbol{\omega}^*$
Output: Searched GNN architecture X^* with embedding $\boldsymbol{\omega}^*$

where we discard $\boldsymbol{r}_o$ for simplicity, and $\boldsymbol{h}_i^K$ is outputted by the last layer of Eq. 5. Intuitively, existing message functions for KG/NRD/HKG (Fig. 1) are contained in the MSeaKG space (Fig. 2(a)). Moreover, some classic KG embeddings can be instantiated as special cases of our space (see Fig. 3), including geometric model RotatE [43] and tensor models n-DistMult [61], HolE [36], n-CP [28].

Overall GNN Space. Except for searching $mg(\cdot; \boldsymbol{\theta}^{mg})$, we also search for other operators (e.g., $agg, comb, act$) like existing GNN searching methods (see Sec. 2.1) as shown in Fig. 2(a). For example, let $O^{agg} = \{sum, mean, max\}$ be the set of candidate $agg(\cdot)$. Step 1 of MPNN (Eq. (1)) can be built on the top of Eq. (6):

$$\begin{aligned} \boldsymbol{m}_i &= agg(\{mg(\boldsymbol{r}(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n); \boldsymbol{\theta}^{mg})\}_{r(e_1,\ldots,e_n) \in I(e_i)}; \boldsymbol{\theta}^{agg}) \\ &= \sum_{o_j \in O^{agg}} \theta_j^{agg} \cdot o_j(\{mg(\boldsymbol{r}(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n); \boldsymbol{\theta}^{mg})\}_{r(e_1,\ldots,e_n) \in I(e_i)}). \end{aligned}$$

Then, step 2 of MPNN (Eq. (2)) for updating $\boldsymbol{e}_i$ are similar to above equation. And the update of $\boldsymbol{r}$ follows the way of [15, 49].

Let $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^{mg}, \boldsymbol{\theta}^{agg}, \ldots\}$ be parameter set for all operators selection in our MPNN framework. Then, an architecture can be represented as $X_{\boldsymbol{\Theta}} = \{mg(\cdot; \boldsymbol{\theta}^{mg}), agg(\cdot; \boldsymbol{\theta}^{agg}), \cdots\}$. Existing GNNs for KG embedding actually can be represented by different $\boldsymbol{\Theta}$. Overall, a GNN model $X_{\boldsymbol{\Theta}}$ encodes the given KG $\mathcal{G}$ into embedding space $\boldsymbol{\omega} = \{\boldsymbol{E}, \boldsymbol{R}\}$, i.e., $\boldsymbol{\omega} = X_{\boldsymbol{\Theta}}(\mathcal{G})$.

3.2 Search Algorithm Design

In this subsection, we introduce how to select the GNN $X_{\boldsymbol{\Theta}}$ that can achieve high performance on the given $\mathcal{G}$. First, we need to

Table 2: The model comparison of LP task on the case of mixed arity. The results of Geometric, NNs and multi-linear baselines are copied from [32]. GNN baselines are re-implemented due to the task variance. S2S is copied from the original paper.

Type	Model	**WikiPeople** (HKG)				**JF17K** (NRD)			
		MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
Geometric	RAE [65]	0.253	0.118	0.343	0.463	0.396	0.312	0.433	0.561
NNs	NaLP [19]	0.338	0.272	0.364	0.466	0.310	0.239	0.334	0.450
	HINGE [40]	0.333	0.259	0.361	0.477	0.473	0.397	0.490	0.618
	NeuInfer [18]	0.350	0.282	0.381	0.467	0.451	0.373	0.484	0.604
Multi-linear	HypE [14]	0.292	0.162	0.375	0.502	0.507	0.421	0.550	0.669
	RAM [32]	0.380	0.279	0.445	0.539	0.539	0.463	0.573	0.690
GNNs	StarE [15]	0.378	0.265	0.452	0.542	0.542	0.454	0.580	0.685
	G-MPNN [60]	0.367	0.258	0.439	0.526	0.530	0.459	0.572	0.688
Searching	S2S [11]	0.372	0.277	0.439	0.533	0.528	0.457	0.570	0.690
	MSeaKG	**0.392**	**0.290**	**0.468**	**0.553**	**0.561**	**0.475**	**0.591**	**0.705**

evaluate the performance of a given GNN X_{Θ}. Generally, the scoring function $f(s;\boldsymbol{\omega})$ verifies the plausibility of fact $s = r(e_1, \ldots, e_n)$. A good embedding $\boldsymbol{\omega}$ can make $f(s;\boldsymbol{\omega})$ to distinguish true or false for a given fact s. Since the GNN X_{Θ} encodes $\mathcal{G}(E, R, S)$ into embedding $\boldsymbol{\omega}$ as $\boldsymbol{\omega} = X_{\Theta}(\mathcal{G})$, we build evaluation of X_{Θ} on $f(s;\boldsymbol{\omega})$. Formally, the GNN search problem for a given HKG $\mathcal{G}$ is formulated as:

$$\min_{\Theta,\boldsymbol{\omega}} \mathcal{L}(X_{\Theta}, \boldsymbol{\omega}; \mathcal{G}), \tag{7}$$

where $\mathcal{L}(X_{\Theta}, \boldsymbol{\omega}; \mathcal{G}) = \sum_{s\in S} \ell(f(s;\boldsymbol{\omega}))$. We follow [10] to instantiate $\ell(\cdot)$ as cross entropy loss with label smoothing ratio.

Solving Eq. 7 is a non-trivial task because X_{Θ} is from a large space. For example, just the structure space of $\{\theta_{ij}^k\}$ reaches to $O(2^{K|O|^2+(2N+1)|O|})$. And X_{Θ} is discrete, indicating the gradient-based optimization cannot be employed since $\nabla_{X_{\Theta}}\mathcal{L}(\cdot)$ does not exist. To enable an efficient search, we first relax the parameters of GNN model Θ from a discrete space into a continuous and probabilistic space $\bar{\Theta}$. More specifically, $\theta_{ij}^k \in \{0, 1\}$ restrictively controls the connectivity between o_i^k with o_j^{k-1}, while $\bar{\theta}_{ij}^k \in [0, 1]$ is the probability that o_i^k is connected with o_j^{k-1}. Then, let $X \sim p_{\bar{\Theta}}(X)$ represent a GNN model X being sampled from the distribution $p_{\bar{\Theta}}(X)$. We reform the problem in Eq. 7 into:

$$\min_{\bar{\Theta},\boldsymbol{\omega}} E_{X\sim p_{\bar{\Theta}}(X)}[\mathcal{L}(X, \boldsymbol{\omega}; \mathcal{G})], \tag{8}$$

where $E[\cdot]$ is the expectation. Appx. A presents more details of Eq. 8's optimization. The overall search procedure of MSeaKG has been summarized in Alg. 1.

4 EXPERIMENTS

4.1 Experimental Setup

The experiments are implemented on top of PyTorch [37] and performed on one single RTX 2080 Ti GPU. Appx. B.1.1 introduces the details of hyper-parameters.

Data Sets. The details of data sets are summarized into Tab. 8 in Appx. B.1.2. For experiments on facts with mixed arities (i.e., the arity n of facts in a data set may be different), we employ: 1) Wiki-People [19] is a HKG $\{r(e_1, e_2, \{(r_{o_j}, e_j)\}_{j=3}^n) : n \in \{2, \cdots, N\}\}$ extracted from wiki-data; 2) JF17K [65] is the n-ary relational data $\{r(e_1, \cdots, e_n) : n \in \{2, \cdots, N\}\}$ extracted from Freebase [6]. Because Alg. 1 needs the validation data, we follow RAM [32] to split the training set of JF17K into training and validation sets, which differs from the original setting in [19, 40, 65]. For experiments on facts with fixed arities (i.e., the arities of facts in a data set are same), we utilize following data sets: 1) KGs: WN18RR [10], FB15k237 [46], and YAGO3-10 [10] ($n = 2$); 2) NRD: WikiPeople-3 and JF17k-3 ($n = 3$), WikiPeople-4 and JF17k-4 ($n = 4$) [31]. Note that GETD [31] removes roles and filters out 3-ary and 4-ary facts from WikiPeople and JF17K to construct WikiPeople-n and JF17k-n.

Tasks and Evaluation Metrics. We compare KG/NRD/HKG embedding models on the link and relation prediction task in the transductive setting. The link prediction (LP) task is to predict the missing entity in the given fact at n possible positions, e.g., predicting the 3rd missing entity $r(e_1, e_2, (r_{o_3}, ?))$ or $r(e_1, e_2, ?)$. The relation classification (RC) task needs to predict the missing relation in a fact when all entities are known, i.e., $?(e_1, \cdots, e_n)$. We employ Mean Reciprocal Ranking (MRR) [51] and Hit@$\{1, 3, 10\}$ (see Appx. B.1.3). Higher MRR and Hit@k values mean better embedding quality.

Baselines. Although the space of MSeaKG is based on GNNs, MSeaKG could cover other types of embedding models. Thus, we include a set of non-GNN models into comparison.

- **Geometric:** Classic TransE [7] and RotatE [43] are for KG. RAE [65] is an upgrade version of m-TransH [58].
- **GNNs:** We adopt R-GCN [41], CompGCN [49], G-MPNN [60] and StarE [15] (Sec. 2.2). Note that we re-implement and tune G-MPNN and StarE since their original tasks are different from ours. G-MPNN follows the inductive setting and StarE only tests the performance of main triplets in hyper-relational facts.
- **Other NNs:** NaLP [19], HINGE [40], and NeuInfer [18].
- **Multi-linear:** The final score of HypE [14] and RAM [32] is computed by multi-way inner product which is extended from a bilinear model DistMult [61].
- **Tensor decomposition:** TuckER [3] is based on Tucker decomposition [48]. Then, we follow GETD [31] to include the extensions n-CP [28] and n-TuckER [3] since they perform well on the case of high arity. n-CP [28] leverages CP decomposition [21].
- **Searching:** Most of GNN searching models in Tab. 1 cannot be applied to KGs since they cannot model relations. We employ one recent GNN searching model AutoGEL [54] that can handle KG. But AutoGEL [54] simply extends $mg_c(\cdot)$ to embed edge in KGs (see Tab. 1), thereby failing to handle more general cases

Table 3: The model comparison of LP task on facts with fixed arity $n = 2$ (i.e., KGs). The results of TransE, RotatE, DistMult, and TuckER are copied from [39]. The reuslts of R-GCN is copied from [49]. Others are copied from original papers.

Type	Model	FB15k237			WN18RR			YAGO3-10		
		MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10
Geometric	TransE [7]	0.310	0.217	0.497	0.206	0.028	0.495	0.501	0.406	0.674
	RotatE [43]	0.336	0.238	0.531	0.475	0.426	**0.574**	0.498	0.405	0.671
Bilinear Tensor Decomp.	DistMult [61]	0.313	0.244	0.490	0.433	0.397	0.502	0.501	0.413	0.661
	TuckER [3]	0.352	0.259	0.536	0.459	0.430	0.514	0.544	0.466	0.681
GNNs	R-GCN [41]	0.248	0.151	0.417	-	-	-	-	-	-
	CompGCN [49]	0.355	0.264	0.535	0.479	0.443	0.546	-	-	-
Searching	AutoGEL [54]	0.357	0.266	0.538	0.479	0.444	0.549	-	-	-
	MSeaKG	**0.360**	**0.267**	**0.545**	**0.482**	**0.445**	0.554	**0.580**	**0.505**	**0.708**

Table 4: The model comparison of LP task on facts with fixed arity $n = 3, 4$ (i.e., NRD). The results of tensor decomposition models are copied from [31], others are copied from [11].

Type	Model	WikiPeople-3		JF17K-3		WikiPeople-4		JF17K-4	
		MRR	Hit@10	MRR	Hit@10	MRR	Hit@10	MRR	Hit@10
Geometric	RAE [65]	0.239	0.379	0.505	0.644	0.150	0.273	0.707	0.835
NNs	NaLP [19]	0.301	0.445	0.515	0.679	0.342	0.540	0.719	0.805
	HINGE [40]	0.338	0.508	0.587	0.738	0.352	0.557	0.745	0.842
	NeuInfer [18]	0.355	0.521	0.622	0.770	0.361	0.566	0.765	0.871
Tensor Decomp.	n-CP [28]	0.330	0.496	0.700	0.827	0.265	0.445	0.787	0.890
	n-TuckER [3]	0.365	0.548	0.727	0.852	0.362	0.570	0.804	0.902
	GETD [31]	0.373	0.558	0.732	0.856	0.386	0.596	0.810	0.913
Searching	S2S [11]	0.386	0.559	0.740	0.860	0.391	0.600	0.822	0.924
	MSeaKG	**0.403**	**0.579**	**0.754**	**0.889**	**0.409**	**0.624**	**0.833**	**0.938**

NRD and HKG. Besides, another searching method S2S [11] is also included, which cannot be extended to handle HKG.

Note that we have included the references to baseline performance in the captions of Tab. 2, 3, and 4. And some baselines are not applicable to different KG forms, thus they cannot be consistently compared on all tables.

Additional Experiments in Appendix. Due to the space limitation, we include more experimental results in Appx. B.2 to provide more insights, including relation prediction in Appx. B.2.1, ablation study of the search algorithm in Appx. B.2.2, and sensitiveness analysis in Appx. B.2.3. Besides, Appx. B.3 presents searched message functions that are data-dependent and can adapt to the given data.

4.2 Main Report for Effectiveness Comparison

The link prediction results on WikiPeople and JF17K have been summarized into Tab. 2. And the relation prediction results are in Tab. 9. Compared with Geometric and NNs methods, GNNs methods achieve outstanding performance, which demonstrates the power of GNNs on the graph tasks. And StarE generally is better than G-MPNN in GNNs methods because the inner product way in G-MPNN cannot handle several relational patterns as mentioned in Sec. 1. Besides, although the multi-linear method RAM utilizes the simple inner product as its scoring function, it carefully models the role semantic information and interaction patterns, thus achieving good performance. Overall, all existing methods cannot consistently achieve the leading performance on different tasks and data sets. In this paper, MSeaKG pursues the high model performance by dynamically designing the most suitable message function for the given data and task. The searched message functions can capture data-level properties (see Fig. 5), thereby showing the leading performance. Especially, the search space of another search method S2S is based on the tensor modeling. Although S2S alleviates the extension issue of tensor modeling, its performance still slightly inferior. Following the graph modeling, MSeaKG benefits from building a message function search space in GNNs.

We also show link prediction results on data sets with fixed arity in Tab. 3 ($n = 2$) and Tab. 4 ($n = 3, 4$). In Tab. 3, MSeaKG achieves comparable results on common KGs because the message functions do not need to be too complex to model facts with low arity. For experiments on high arity in Tab. 4, we first observe that classic tensor decomposition models (n-CP, n-TuckER, GETD) perform better than Geometric and NN-based methods. Then, S2S proposes to dynamically sparsify the core tensor of tensor decomposition models for the given data and further improve the performance of tensor decomposition models. MSeaHKG still significantly improves the performance of S2S even in the scenario of fixed arity. That is because S2S simply assumes 3 relationships between entities and relations in the search space: positive, irrelevant, and negative. But the message function space in Sec. 3.1 could characterize more complex interactions between entities and relations.

4.3 Ablation Study

Except for the main experimental results, here we report the performance of several variants of MSeaKG (see Tab. 5) to investigate some key designs in this paper, including MSeaKG^{W_r}, MSeaKG^{op}, MSeaKGst for the search space, MSeaKG^{darts} and MSeaKG^{rl} for the search algorithm. Due to the space limitation, Appx. B.2.2 presents the experimental settings of MSeaKG^{darts} and MSeaKG^{rl} with an analysis of effectiveness and efficiency.

Table 5: The comparison of variants of MSeaKG in the link prediction task on the case of mixed arity.

Type	Model	WikiPeople (HKG)				JF17K (NRD)			
		MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
GNNs	StarE	0.378	0.265	0.452	0.542	0.542	0.454	0.580	0.685
	G-MPNN	0.367	0.258	0.439	0.526	0.530	0.459	0.572	0.688
Search	S2S	0.372	0.277	0.439	**0.533**	0.528	0.457	0.570	0.690
	MSeaKG	**0.392**	**0.291**	**0.468**	**0.553**	**0.561**	**0.475**	**0.591**	**0.705**
Variants of space	MSeaKG^{W_r}	0.354	0.233	0.431	0.520	0.512	0.445	0.553	0.671
	MSeaKG^{op}	0.385	0.274	0.460	0.548	0.554	0.468	0.579	0.699
	MSeaKGst	0.391	0.278	0.465	0.552	0.559	**0.475**	0.585	0.702
Variants of algorithm	MSeaKG^{darts}	0.373	0.275	0.445	0.535	0.554	0.460	0.588	0.697
	MSeaKG^{rl}	0.380	0.281	0.457	0.542	0.558	0.472	0.590	0.701

Search Space. We first present the configuration of variants:

- MSeaKG^{W_r} basically enables current GNN searching methods working on HKGs. Inspired by R-GCN (see Eq. 3), we first replace the transform matrix $\boldsymbol{W}$ in $mg_c(\cdot)$ (see Eq. 1) to $\boldsymbol{W}_r$. Then, we concatenate the entity embeddings as $\boldsymbol{h} = concat(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n, \boldsymbol{0}, \ldots, \boldsymbol{0})$. Note that the number of zero embeddings $\boldsymbol{0}$ is equal to $N - n$. We utilize the message function $mg_c(\boldsymbol{r}(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n)) = \boldsymbol{W}_r\boldsymbol{h}$ to replace Eq. 6. Other steps are same with original version.
- MSeaKG^{op} only searches operations of operators ϕ and γ in $mg(\cdot;\boldsymbol{\theta})$ (i.e., $\boldsymbol{\theta} = \{\theta_i^{\phi_k}\} \cup \{\theta_i^{\gamma_k}\}$), while keeping the structure of StarE's message functions. Other steps are same with original version.
- MSeaKGst only searches structures of the proposed message function $mg(\cdot;\boldsymbol{\theta})$, and sets ϕ, γ to $corr, wsum$ respectively (i.e., $\boldsymbol{\theta} = \{\theta_{ij}^k\}$). The fixed operations are selected based on better empirical performance. Other steps are same with original version.

From Tab. 5, we observe that the simple extension version MSeaKG^{W_r} even cannot achieve as good performance as existing GNNs (e.g., StarE and G-MPNN). This verifies the claim that the simple message function in the existing GNN searching method (e.g., AutoGEL [54] discussed in Sec. 2.2) may not be able to handle the complex correlations between relations and entities on HKGs (see Tab. 3 for more comparison on KGs). Moreover, MSeaKG^{op} keeps the same message function structure with StarE but searches suitable operations. Differ from manually tuning operations in StarE, the automatic way is more powerful so that MSeaKG^{op} achieves a minor improvement compared with StarE. As for MSeaKGst, it can search for more flexible structures of message functions for the given data and achieve the best performance among several variants. It can illustrate that the message function design is important to KG embedding. However, MSeaHKGst is still slightly inferior compared with the original version of MSeaKG. This shows that the best structure and operations are dependent. Simply fixing operations to search the structure may lead to the sub-optimal results.

Scoring Functions. There are many scoring functions that can be utilized to decode embeddings into score, such as DistMult [61] and Transformer [50]. In principle, MSeaKG can implement most existing scoring functions as its decoder. In this paper, we simply concatenate the embeddings of known entities and relations in a fact and feed it into a two-layer MLP. That is mainly because the message function space has the strong capability to capture the interactions between entities and relations. And some classic scoring functions are covered by MSeaKG (see Fig. 3).

Table 6: The model comparison of variants of MSeaKG in terms of adopted scoring functions.

Scoring Function	MRR Performance		Searching Time (in hours)	
	WikiPeople	JF17K	WikiPeople	JF17K
MLP	0.392	0.561	30.9	7.7
DistMult	0.377	0.522	28.1	7.1
Transformer	0.381	0.548	92.8	18.5

As shown in Tab. 6, we investigate the influence of scoring functions on the searching effectiveness and efficiency. We can observe that the MLP version achieves best effectiveness and comparable efficiency. During search, MSeaKG samples different architectures, uses the sampled architecture to learn embeddings, then forward embeddings to scoring functions for calculating final scores. It is intuitive that the transformer version consumes more time than the versions of MLP and DistMult. Hence, the transformer version is hard to achieve the feedback of sampled architectures, which lead to insufficient training compared with the MLP version. As for the DistMult version, its capability is inferior since it cannot cover some relational patterns. Thus, we use a simple scoring function for the sake of effectiveness and efficiency.

5 CONCLUSION

In this paper, we propose a new searching method for KG embedding, named MSeaKG. First, we present a novel search space of message functions, which allows KG/NRD/HKG forms as inputs. By enabling the structure search and operation selection, some classic KG embedding models and existing message functions for KG/NRD/HKG could be instantiated as special cases of our space. Then, we leverage an efficient algorithm to search the message function and other GNN components for the given data. Experimental results show that MSeaKG can consistently achieve leading performance on benchmark KGs, NRD, and HKGs by designing data-aware message functions.

One limitation of MSeaKG is not covering the path-based GNNs [72], which aggregate messages from paths instead of neighbors. This is a worthwhile direction to try, although in the case of NRD and HKG, the path concept in graph is more complex. Moreover, MSeaKG tends to fit those relations with a large proportion of KGs for high performance, while ignoring those rare relations. Thus, another future direction is expected to search multiple message functions to alleviate this issue.

ACKNOWLEDGMENTS

Lei Chen is partially supported by National Science Foundation of China (NSFC) under Grant No. U22B2060 and 61729201, National Key Research and Development Program of China Grant No. 2021YFE020339, the Hong Kong RGC GRF Project 16213620, RIF Project R6020-19, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants MHX/078/21 and PRP/004/22FX, Microsoft Research Asia Collaborative Research Grant and HKUST-Webank joint research lab grants. Shimin Di is supported by the JC STEM Lab of Data Science Foundations funded by The Hong Kong Jockey Club Charities Trust.

REFERENCES

[1] Youhei Akimoto, Shinichi Shirakawa, Nozomu Yoshinari, Kento Uchida, Shota Saito, and Kouhei Nishida. 2019. Adaptive stochastic natural gradient method for one-shot neural architecture search. In *ICML*.

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The Semantic Web*. Springer, 722–735.

[3] I. Balazevic, C. Allen, and T. Hospedales. 2019. TuckER: Tensor Factorization for Knowledge Graph Completion. In *EMNLP*. 5188–5197.

[4] J. Bergstra, D. Yamins, and D. D. Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. (2013).

[5] Max Berrendorf, Evgeniy Faerman, Laurent Vermue, and Volker Tresp. 2020. On the ambiguity of rank-based evaluation of entity alignment or link prediction methods. *arXiv preprint arXiv:2002.06914* (2020).

[6] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*. 1247–1250.

[7] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*. 2787–2795.

[8] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *WWW*. 151–161.

[9] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. 2020. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *ECCV*. Springer, 465–480.

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

[11] Shimin Di, Quanming Yao, and Lei Chen. 2021. Searching to Sparsify Tensor Decomposition for N-ary Relational Data. In *Webconf*. 4043–4054.

[12] Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. 2019. Neural architecture search: A survey. *J. Mach. Learn. Res.* 20, 55 (2019), 1–21.

[13] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and efficient hyperparameter optimization at scale. In *ICML*. PMLR, 1437–1446.

[14] Bahare Fatemi, Perouz Taslakian, David Vazquez, and David Poole. 2020. Knowledge hypergraphs: Prediction beyond binary relations. (2020).

[15] Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. 2020. Message Passing for Hyper-Relational Knowledge Graphs. In *EMNLP*.

[16] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2020. Graph neural architecture search. In *IJCAI*, Vol. 20. 1403–1409.

[17] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*. PMLR, 1263–1272.

[18] Saiping Guan, Xiaolong Jin, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. 2020. Neuinfer: Knowledge inference on n-ary facts. In *ACL*. 6141–6151.

[19] Saiping Guan, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2019. Link prediction on n-ary relational data. In *WWW*. 583–593.

[20] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1025–1035.

[21] Frank L Hitchcock. 1927. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics* 6, 1-4 (1927), 164–189.

[22] F. Hutter, L. Kotthoff, and J. Vanschoren. 2018. *Automated Machine Learning: Methods, Systems, Challenges*. Springer.

[23] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[24] Shengli Jiang and Prasanna Balaprakash. 2020. Graph Neural Network Architecture Search for Molecular Property Prediction. In *IEEE Big Data*. IEEE, 1346–1353.

[25] S. Kazemi and D. Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *NeurIPS*. 4284–4295.

[26] D.P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. In *ICLR*.

[27] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[28] T. Lacroix, N. Usunier, and G. Obozinski. 2018. Canonical Tensor Decomposition for Knowledge Base Completion. In *ICML*. 2863–2872.

[29] Haoyang Li, Shimin Di, Zijian Li, Lei Chen, and Jiannong Cao. 2022. Black-box Adversarial Attack and Defense on Graph Neural Networks. In *ICDE*. IEEE, 1017–1030.

[30] H. Liu, K. Simonyan, and Y. Yang. 2018. DARTS: Differentiable architecture search. In *ICLR*.

[31] Yu Liu, Quanming Yao, and Yong Li. 2020. Generalizing Tensor Decomposition for N-ary Relational Knowledge Bases. In *WebConf*. 1104–1114.

[32] Yu Liu, Quanming Yao, and Yong Li. 2021. Role-Aware Modeling for N-ary Relational Knowledge Bases. In *WebConf 2021*. 2660–2671.

[33] Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. 2017. Neural network-based question answering over knowledge graphs on word and character level. In *WWW*. 1211–1220.

[34] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. *ICLR* (2017).

[35] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *Proc. IEEE* 104, 1 (2015), 11–33.

[36] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *AAAI*.

[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 8024–8035.

[38] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. 2018. Efficient Neural Architecture Search via Parameter Sharing. In *ICML*. 4092–4101.

[39] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. 2021. Knowledge graph embedding for link prediction: A comparative analysis. *TKDD* 15, 2 (2021), 1–49.

[40] Paolo Rosso, Dingqi Yang, and Philippe Cudré-Mauroux. 2020. Beyond triplets: hyper-relational knowledge graph embedding for link prediction. In *WebConf*. 1885–1896.

[41] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*. Springer, 593–607.

[42] DI Shimin, YAO Quanming, Yongqi ZHANG, and CHEN Lei. 2021. Efficient Relation-aware Scoring Function Search for Knowledge Graph Embedding. In *ICDE*. IEEE, 1104–1115.

[43] Z. Sun, Z. Deng, J. Nie, and J. Tang. 2019. RotatE: Knowledge graph embedding by relational rotation in complex space. In *ICLR*.

[44] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2018. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*.

[45] Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. 2020. A Re-evaluation of Knowledge Graph Completion Methods. In *ACL*. 5516–5522.

[46] K. Toutanova and D. Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Workshop on CVSMC*. 57–66.

[47] T. Trouillon, Christopher R., É. Gaussier, J. Welbl, S. Riedel, and G. Bouchard. 2017. Knowledge graph completion via complex tensor factorization. *JMLR* 18, 1 (2017), 4735–4772.

[48] Ledyard R Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika* 31, 3 (1966), 279–311.

[49] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based Multi-Relational Graph Convolutional Networks. In *ICLR*.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*. 5998–6008.

[51] Ellen Voorhees. 1999. The TREC-8 question answering track report. In *TREC*, Vol. 99. 77–82.

[52] Hongwei Wang, Hongyu Ren, and Jure Leskovec. 2020. Entity context and relational paths for knowledge graph completion. *arXiv:2002.06757* (2020), 47.

[53] Q. Wang, Z. Mao, B. Wang, and L. Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *TKDE* 29, 12 (2017), 2724–2743.

[54] Zhili Wang, Shimin Di, and Lei Chen. 2021. AutoGEL: An Automated Graph Neural Network with Explicit Link Information. *Advances in Neural Information Processing Systems* 34 (2021), 24509–24522.

[55] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*.

[56] Lanning Wei, Huan Zhao, and Zhiqiang He. 2022. Designing the topology of graph neural networks: A novel feature fusion perspective. In *Proceedings of the ACM Web Conference 2022*. 1381–1391.

[57] Lanning Wei, Huan Zhao, Quanming Yao, and Zhiqiang He. 2021. Pooling architecture search for graph classification. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2091–2100.

[58] Jianfeng Wen, Jianxin Li, Yongyi Mao, Shini Chen, and Richong Zhang. 2016. On the representation and embedding of knowledge bases beyond binary relations. In *IJCAI*.

[59] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. 2018. SNAS: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926* (2018).

[60] Naganand Yadati. 2020. Neural Message Passing for Multi-Relational Ordered and Recursive Hypergraphs. *NeurIPS* 33 (2020).

[61] B. Yang, W. Yih, X. He, J. Gao, and L. Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

[62] Quanming Yao and Mengshuo Wang. 2018. Taking human out of learning applications: A survey on automated machine learning. (2018).

[63] Jiaxuan You, Zhitao Ying, and Jure Leskovec. 2020. Design space for graph neural networks. *NeurIPS* 33 (2020).

[64] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. 2019. Understanding and robustifying differentiable architecture search. *arXiv preprint arXiv:1909.09656* (2019).

[65] Richong Zhang, Junpeng Li, Jiajie Mei, and Yongyi Mao. 2018. Scalable instance reconstruction in knowledge bases via relatedness affiliated embedding. In *WWW*. 1185–1194.

[66] Yongqi Zhang and Quanming Yao. 2022. Knowledge graph reasoning with relational digraph. In *Proceedings of the ACM Web Conference 2022*. 912–924.

[67] Yongqi Zhang, Quanming Yao, and Lei Chen. 2020. Interstellar: Searching recurrent architecture for knowledge graph embedding. *Advances in Neural Information Processing Systems* 33 (2020), 10030–10040.

[68] Yongqi Zhang, Quanming Yao, Wenyuan Dai, and Lei Chen. 2020. AutoSF: Searching Scoring Functions for Knowledge Graph Embedding. In *ICDE*. 433–444.

[69] Ziwei Zhang, Xin Wang, and Wenwu Zhu. 2021. Automated Machine Learning on Graphs: A Survey. *arXiv preprint arXiv:2103.00742* (2021).

[70] Huan Zhao, Quanming Yao, and Weiwei Tu. 2021. Search to aggregate neighborhood for graph neural network. *arXiv preprint arXiv:2104.06608* (2021).

[71] Kaixiong Zhou, Qingquan Song, Xiao Huang, and Xia Hu. 2019. Auto-gnn: Neural architecture search of graph neural networks. *arXiv preprint arXiv:1909.03184* (2019).

[72] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. *NeurIPS* 34 (2021), 29476–29490.

A FULL DERIVATION INVOLVED IN SEC. 3.2

To compute the gradient w.r.t. $\bar{\Theta}$, we first utilize the reparameterization trick $X = g_{\bar{\Theta}}(U)$ [59], where U is sampled from a uniform distribution $p(U)$. Then the gradient w.r.t. $\bar{\Theta}$ and $\boldsymbol{\omega}$ is computed as:

$$\nabla_{\bar{\Theta}} E_{X\sim p_{\bar{\Theta}}(X)}[\mathcal{L}(X,\boldsymbol{\omega};\mathcal{G})] = \nabla_{\bar{\Theta}} E_{U\sim p(U)}[\mathcal{L}(g_{\bar{\Theta}}(U),\boldsymbol{\omega};\mathcal{G})] \quad (9)$$
$$= \nabla_{\bar{\Theta}} \int p(U)\mathcal{L}(g_{\bar{\Theta}}(U),\boldsymbol{\omega};\mathcal{G})dU = \int p(U)\nabla_{\bar{\Theta}}\mathcal{L}(g_{\bar{\Theta}}(U),\boldsymbol{\omega};\mathcal{G})dU$$
$$= E_{U\sim p(U)}[\nabla_{\bar{\Theta}}\mathcal{L}(g_{\bar{\Theta}}(U),\boldsymbol{\omega};\mathcal{G})] = E_{U\sim p(U)}[\mathcal{L}'(g_{\bar{\Theta}}(U),\boldsymbol{\omega};\mathcal{G})\nabla_{\bar{\Theta}} g_{\bar{\Theta}}(U)],$$

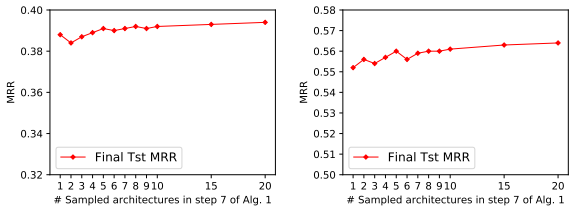
$$\nabla_{\boldsymbol{\omega}} E_{X\sim p_{\bar{\Theta}}(X)}[\mathcal{L}(X,\boldsymbol{\omega};\mathcal{G})] = \nabla_{\boldsymbol{\omega}} \int p_{\bar{\Theta}}(X)\mathcal{L}(X,\boldsymbol{\omega};\mathcal{G})dX$$
$$= \int p_{\bar{\Theta}}(X)\nabla_{\boldsymbol{\omega}}\mathcal{L}(X,\boldsymbol{\omega};\mathcal{G})dX = E_{X\sim p_{\bar{\Theta}}(X)}[\nabla_{\boldsymbol{\omega}}\mathcal{L}(X,\boldsymbol{\omega};\mathcal{G})]. \quad (10)$$

Table 7: List of hyper-parameters in main experiments.

	LP						RP	
	WP	**JK**	**W-3**	**J-3**	**W-4**	**J-4**	**WP**	**JK**
learning rate	0.0001	0.001	0.0001	0.001	0.0001	0.0001	0.0001	0.001
MPNN layers	2	2	1	1	1	1	2	2
K	4	4	3	2	2	3	4	4
batch size	256	128	128	128	256	128	256	128
dim. d	256	256	128	128	256	128	128	128
dropout ratio	0.15	0.2	0.1	0.1	0.05	0.15	0.2	0.15
label smooth. τ	0.3	0.8	0.7	0.5	0.8	0.7	0.1	0.1

Table 8: The statistical summary on data sets.

data set	**role**	**# all / $n>2$ facts**	n	**# ent**	**# rel**	**train**	**valid**	**test**
JK	×	100,947 / 46,320	2/6	28,645	322	61,104	15,275	24,568
WP	√	382,229 / 44,315	2/9	47,765	707	305,725	38,223	38,281
W-18	×	93,003 / 0	2/2	40,943	11	86,835	3,034	3,134
F-237	×	310,116 / 0	2/2	14,541	237	272,115	17,535	20,466
YG-3	×	1,089,040 / 0	2/2	123,188	37	1,079,040	5,000	5,000
J-3	×	34,544 / 34,544	3/3	11,541	104	27,635	3,454	3,455
J-4	×	9,509 / 9,509	4/4	6,536	23	7,607	951	951
W-3	×	25,820 / 25,820	3/3	12,270	66	20,656	2,582	2,582
W-4	×	15,188 / 15,188	4/4	9,528	50	12,150	1,519	1,519



(a) WikiPeople. (b) JF17K.

Figure 4: Testing MRR v.s. the number of sampled architectures in step 7 of Alg. 1.

Note that $\nabla_{\bar{\Theta}} g_{\bar{\Theta}}(U)$ can be computed if $g_{\bar{\Theta}}(U)$ is differentiable. we build the reparameterization trick $X = g_{\bar{\Theta}}(U)$ based on Gumbel-Softmax [23] or Concrete distribution [34]. For clarity, we simplify $\bar{\Theta}$ to the parameter $\bar{\theta}$ for a specific operator space O:

$$X_o = g_{\bar{\theta}}(U) = \frac{\exp((\log\bar{\theta}_o - \log(-\log(U_o)))/tem)}{\sum_{o'\in O}\exp((\log\bar{\theta}_{o'} - \log(-\log(U_{o'})))/tem)}, \quad (11)$$

where tem is the temperature of softmax, and $U_o \sim Uniform(0,1)$. It has been proven that $p(\lim_{tem\to 0} X_o = 1) = \bar{\theta}_o / \sum_{o'\in O}\bar{\theta}_{o'}$ making the stochastic differentiable relaxation unbiased once converged [59]. And the details of $\nabla_{\bar{\Theta}} g_{\bar{\Theta}}(U)$ can refer to [59].

B MORE EXPERIMENTS

B.1 Experimental Setup

B.1.1 Hyper-parameter Settings and Related Discussion. We have summarized the hyper-parameters of this paper in Tab. 7. The hyper-parameter set includes Adam optimizer [26], learning rate $\in \{0.1, 0.01, 0.001, 0.0001, 0.00001\}$, # MPNN layers $\in \{1,2,3,4\}$ (see left part of Fig. 2), # layers in message functions $K \in \{1,2,3,4,5\}$ (see right part of Fig. 2), batch size $\in \{64, 128, 256, 512\}$, embedding dimension $d \in \{64, 128, 256, 512\}$, dropout ratio $\in \{0, 0.05, 0.1, 0.15, \ldots, 0.5\}$, label smoothing ratio $\tau \in \{0, 0.1, 0.2, \ldots, 0.9\}$. Note that the label smoothing ratio τ is employed to relax the one-hot label vector $\boldsymbol{y}$. In practical, we set $\boldsymbol{y}_i = 1-\tau$ for the ground truth entity/relation, while $\boldsymbol{y}_i = \tau/|E|-1$ for link prediction and $\boldsymbol{y}_i = \tau/|R|-1$ for relation prediction. That is because there are usually a large candidate space for relations and entities, while Using one-hot vector is quite restrictive. The hyper-parameters of MSeaKG/G-MPNN/StarE are tuned with the help of `optuna.samplers.TPESampler` [4, 13][1].

[1] https://optuna.readthedocs.io/en/stable/reference/samplers/index.html

Table 9: The model comparison of the relation prediction task on the arity of mixed case. The results of NNs are copied from [18], others are based on our implementations.

Type	Model	WikiPeople (HKG)				JF17K (NRD)			
		MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
NNs	NaLP [19]	0.735	0.595	-	-	0.825	0.762	-	-
	NeuInfer [18]	0.765	0.686	-	-	0.861	0.832	-	-
GNNs	StarE [15]	0.800	0.753	0.936	0.951	0.901	0.884	0.929	0.963
	G-MPNN [60]	0.777	0.694	0.905	0.912	0.864	0.842	0.883	0.917
Searching	S2S [11]	0.813	0.744	0.928	0.960	0.912	0.877	0.932	0.951
	MSeaKG	**0.828**	**0.784**	**0.951**	**0.970**	**0.932**	**0.893**	**0.948**	**0.971**

B.1.2 Data Statistics. As shown in Tab. 8, we present the statistic summary of the benchmark data sets adopted in this paper.

B.1.3 Evaluation Measurement. As in [7, 55], we report performance under the "filtered" setting, i.e., evaluating the rank of test fact after removing all corrupted facts that exist in the train, valid, and test data set. That is because true facts in data set should be not considered as faults when evaluating a test fact. Let $\text{rank}_{i,n}$ denote the rank of ground truth entity at position n of i-th fact $s = (r, e_1, \ldots, e_N)$ (see more discussions about policy in [5, 39, 45]):

$$\text{rank}^{+}_{i,n} = |\{e' \in E \setminus \{e_n\} : f(s'; \boldsymbol{\omega}) > f(s; \boldsymbol{\omega}) \cap s' \notin S\}| + 1,$$

$$\text{rank}^{-}_{i,n} = |\{e' \in E \setminus \{e_n\} : f(s'; \boldsymbol{\omega}) \geq f(s; \boldsymbol{\omega}) \cap s' \notin S\}|,$$

where $s' = (r, e_1, \ldots, e_{n-1}, e', \ldots, e_N)$ and $\text{rank}_{i,n} = \frac{1}{2}(\text{rank}^{+}_{i,n} + \text{rank}^{-}_{i,n})$ [5]. Then we adopt the classical metrics [7, 55]:

- Mean Reciprocal Ranking (MRR): $1/N|S| \sum_{i=1}^{|S|} \sum_{n=1}^{N} 1/\text{rank}_{i,n}$;
- Hit@1, Hit@3, and Hit@10, where Hit@k is given by $1/|S| \sum_{i=1}^{|S|} \sum_{n=1}^{N} \mathbb{I}(\text{rank}_{i,n} \leq k)$ and $\mathbb{I}(\cdot)$ is the indicator function.

B.2 More Experimental Results

B.2.1 Relation Prediction. Due to the space limit, we put the results of the relation prediction here. Tab. 9 reports the model performance of the relation prediction task on the mixed case of KGs (HKG and NRD), which also demonstrates the effectiveness of MSeaKG compared with baselines.

B.2.2 Ablation Study of Search Algorithm. In this paper, we adopt one-shot NAS search algorithms due to the searching efficiency (see more discussion in Sec. 2.1). Existing one-shot NAS algorithms can be roughly categorized into: stochastic differentiable method (e.g., SNAS [59]), deterministic differentiable method (e.g., DARTS [30]), and policy gradient-based method (e.g., ENAS [38], ASNG [1]). Here, we implement two more variants, MSeaKG^{darts} based on DARTS and MSeaKG^{rl} based on ASNG, to investigate the performance of other NAS search algorithms in our application scenario.

- MSeaKG^{darts} follows DARTS to directly relax $\boldsymbol{\Theta}$ to learnable parameters. Then, the computation in DAG (see Eq. 5) will be reformed to: $\boldsymbol{h}_i^k = \sum_{j=1}^{|O|} \alpha_{ij}^k \cdot o_i^k(\boldsymbol{h}_j^{k-1})$, $\alpha_{ij}^k = \theta_{ij}^k / \sum_{j'=1}^{|O|} \theta_{ij'}^k$. Then, we can minimize the loss $\mathcal{L}(\boldsymbol{\Theta}, \boldsymbol{\omega}; \mathcal{G})$ through the gradient $\nabla_{\boldsymbol{\Theta}} \mathcal{L}(\cdot)$.
- MSeaKG^{rl} follows the policy gradient-based NAS search algorithm to derive: $\nabla_{\bar{\boldsymbol{\Theta}}} E_{X \sim p_{\bar{\boldsymbol{\Theta}}}(X)}[\mathcal{L}(X, \boldsymbol{\omega}; \mathcal{G})] = \nabla_{\bar{\boldsymbol{\Theta}}} E_{X \sim p_{\bar{\boldsymbol{\Theta}}}(X)}[\mathcal{L}(X, \boldsymbol{\omega}; \mathcal{G}) \nabla_{\bar{\boldsymbol{\Theta}}} \log p_{\bar{\boldsymbol{\Theta}}}(X)]$. We utilize ASNG [1], which implements the fisher information matrix in $\nabla_{\bar{\boldsymbol{\Theta}}} \log p_{\bar{\boldsymbol{\Theta}}}(X)$ for fast convergence.

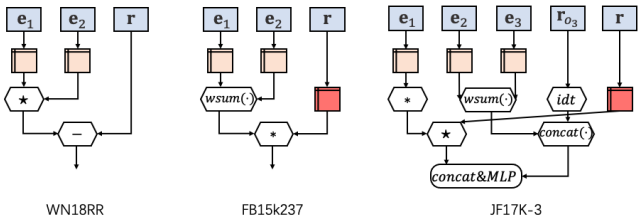


Figure 5: The searched message functions by MSeaKG.

We here analyze the model comparison between original MSeaKG with its variants MSeaKG^{rl} and MSeaKG^{darts}. From Tab. 5 and Tab. 10, we observe that MSeaKG and MSeaKG^{rl} have better performance than MSeaKG^{darts} in both effectiveness and efficiency comparisons. First, DARTS aims to train a supernet by mixing all candidate operations during the searching phase, then it will derive a discrete architecture after finishing the search. But the weights $[\boldsymbol{\alpha}_i^k]$ cannot cannot converge to a one-hot vector, which lead to performance collapse after removing $|O-1|$ operations in O [9, 64]. Second, it will consume more computational resources when maintaining all operations during the search. Instead, MSeaKG and MSeaKG^{rl} train discrete architectures in searching (the bounded discreteness of MSeaKG is discussed in Appx. A), which avoids performance collapse and large computational overhead. As for the comparison between MSeaKG with MSeaKG^{rl}, MSeaKG achieves slight improvements. That is mainly because MSeaKG directly calculates the gradient w.r.t. $\bar{\boldsymbol{\Theta}}$ from the loss $\mathcal{L}(\cdot)$, while MSeaKG^{rl} takes the loss $\mathcal{L}(\cdot)$ as a reward to feed it to a reinforcement learning controller.

B.2.3 Sensitive analysis on the number of sampled architectures. Before deriving the final architecture, MSeaKG samples several architectures (step 7 in Alg. 1) since the stochastic algorithm may output different models. We investigate the influence of the different numbers of sampled architectures. As shown in Fig. 4, the model performance may not be good enough if the number of samples is too less, But overall, MSeaKG is not sensitive to this parameter.

B.3 Data-aware Model for Relational Patterns

As shown in Fig. 5, we demonstrate several message functions searched by MSeaKG on WN18RR, FB15k237 and JF17K-3. We can observe that the message functions are data-dependent, which can capture the relational patterns in different data sets. For example, the message function searched on WN18RR is the exact RotatE

Table 10: The training time comparison (in hours) of GNN-based models in the link prediction task on HKGs.

Type	Model	Mixed Arity		Arity=3		Arity=4	
		WikiPeople	JF17K	W-3	J-3	W-4	J-4
Search	MSeaKG	30.9 ± 4.7	7.7 ± 1.8	1.8 ± 0.4	1.4 ± 0.2	0.9 ± 0.1	1.1 ± 0.2
Variants of algorithm	MSeaKG^{darts}	40.2 ± 2.5	11.5 ± 2.2	2.5 ± 0.5	2.1 ± 0.3	1.5 ± 0.2	1.3 ± 0.1
	MSeaKG^{rl}	35.1 ± 3.0	10.3 ± 1.7	2.3 ± 0.3	1.5 ± 0.2	1.4 ± 0.2	1.1 ± 0.1

Table 11: The MRR performance of models tested on relation-level of DDB14 [52].

	Symmetry	Asymmetry		Overall
relation name	may be allelic with	belong(s) to	may cause	
ratio in $\mathcal{S}_{tra}$	0.7%	20.4%	61.9%	100.0%
TransE	0.00	0.19	0.20	0.18
DistMult	0.19	0.21	0.25	0.20
MSeaKG	0.08	0.25	0.30	0.26

model [43], which has been proven to cover symmetric relations (37% in WN18RR).

One motivation behind this paper is to pursue a message function that can adapt to the relational patterns in the given KG/NRD/HKG. To verify it, we conduct experiments on a medical KG DDB14 [52]. It contains the terminologies such as drugs, symptoms, diseases and their relationships. As shown in Tab. 11, MSeaKG focuses on those relations with high ratio (e.g., `may cause`) in the given KG to pursue the high performance. It will adapt to those relations that are mostly present in the data. In other words, the message function optimized by MSeaKG is very flexible to handle relation patterns since the optimized one can consistently evolve for the given data.