Modeling N-ary Relational Knowledge Bases with Tensor Decomposition

YU LIU, Beijing National Research Center for Information Science and Technology (BNRist), Department of Electronic Engineering, Tsinghua University, Beijing, China
QUANMING YAO, Department of Electronic Engineering, Tsinghua University, Beijing, China
YONG LI, Beijing National Research Center for Information Science and Technology (BNRist), Department of Electronic Engineering, Tsinghua University, Beijing, China

The binary relational knowledge base (KB, a.k.a. knowledge graph), representing real-world knowledge with binary relations and entities, has been an important research topic in artificial intelligence, while, considerable knowledge also involves beyond-binary relations. Recently, the area proposes to model n-ary relational KBs with both binary and beyond-binary relations included. However, most current models are extended from translational distance and neural network models in binary relational KBs, which suffer from weak expressiveness and high complexity, respectively. To overcome such issues, in this work, we propose a novel two-step modeling framework, GETD, generalizing the powerful tensor decomposition technique from binary relational KBs to the n-ary case. For n-ary relational KBs with single-arity relations, the GETD framework introduces Tucker decomposition and Tensor Ring decomposition for expressive and efficient modeling. Furthermore, the framework is technically extended for the representation of n-ary relational KBs with mixed-arity relations. The existing negative sampling technique is also generalized to the n-ary case for GETD. In addition, we theoretically prove that the GETD framework is fully expressive to completely represent any KBs. Empirical results on two representative datasets show that the proposed framework significantly outperforms the state-of-the-art methods, achieving 11–26% and 4–7% improvements on Hits@10 for the single-arity and the mixed-arity cases, respectively.

CCS Concepts: • **Computing methodologies** → **Semantic networks**; **Machine learning algorithms**; **Artificial intelligence**;

Additional Key Words and Phrases: Knowledge base modeling, n-ary relation, scoring function, Tucker decomposition, Tensor Ring decomposition

ACM Reference format:
Yu Liu, Quanming Yao, and Yong Li. 2025. Modeling N-ary Relational Knowledge Bases with Tensor Decomposition. *ACM Trans. Intell. Syst. Technol.* 16, 3, Article 61 (May 2025), 29 pages.
https://doi.org/10.1145/3709002

This work was supported in part by the National Nature Science Foundation of China under U23B2030, 62272260, and 62171260.

Authors' Contact Information: Yu Liu, Department of Electronic Engineering, Tsinghua University, Beijing, China; e-mail: liuyu2419@126.com; Quanming Yao, Department of Electronic Engineering, Tsinghua University, Beijing, China; e-mail: qyaoaa@tsinghua.edu.cn; Yong Li (corresponding author), Department of Electronic Engineering, Tsinghua University, Beijing, China; e-mail: liyong07@tsinghua.edu.cn.

ACM 2157-6912/2025/5-ART61
https://doi.org/10.1145/3709002

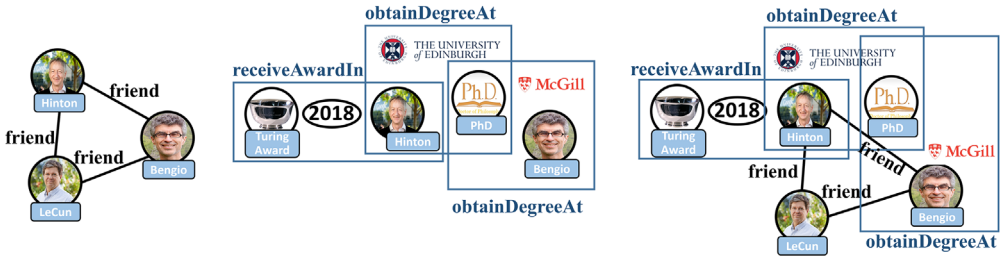
1 Introduction

In the past decade, the emerging of numerous web-scale **knowledge bases (KBs)** such as Freebase [3], Wikidata [46], YAGO [41], and Google's Knowledge Graph [38], has inspired various applications, e.g., question answering [32], recommender systems [53], and **natural language processing (NLP)** [31]. Most of these KBs are constructed based on binary relations with triplet facts represented as (*head entity, relation, tail entity*), which are termed as binary relational KBs. Specifically, extensive studies have been proposed to model such KBs, including translational distance models [6, 26, 50], neural network models [8, 37, 39], and tensor decomposition models [2, 22, 43, 52]. These models first embed relations and entities into low-dimensional space, and then design a *scoring function* based on such embeddings for fact plausibility measure [20, 33, 47].

Despite the great attention in binary relational KBs, n-ary relational KBs with both binary and beyond-binary relations are less studied. In fact, n-ary (a.k.a. multi-fold) relations play an important role in KBs. For instance, *Purchase* is a common ternary (3-ary) relation, involved a *Person*, a *Product*, and a *Seller*. *Sports_award* is a 4-ary relation, involved a *Player*, a *Team*, an *Award,* and a *Season*, giving an example of *Michael Jordan from Chicago Bulls was awarded the MVP award in 1991–1992 NBA season*. Also, as observed in [11] and [51], 61% of the relations in Freebase are beyond-binary and more than 1/3 of the entities therein participate in these relations. Besides, since higher-arity relations with more knowledge are closer to natural language, modeling n-ary relational KBs provides an excellent potential for question answering-related NLP applications [10]. Figure 1 provides examples of n-ary relational KBs. Especially, Figure 1(a) and (b) is n-ary relational KBs with single-arity relations, while Figure 1(c) is an example of n-ary relational KB with mixed-arity relations (binary and ternary relations). As shown in Figure 1, mixed-arity relational KBs can capture more complex semantics and encode more comprehensive knowledge than single-arity relational KBs. This capability makes them a key focus of current research [9, 30].

Specifically, to model n-ary relational KBs, existing works can be categorized into two classes of translational distance models and neural network models in terms of scoring function. The first category models such as m-TransH [51] and RAE [54] are extended from TransH [50] in binary relational KBs, translating entities onto relation-specific hyperplanes for plausibility scores. However, these models also face the weak expressiveness of TransH [22], and fail to represent some n-ary relations, which impairs the performance. Neural network models such as NaLP [14] and HINGE [36] explore neural network-based scoring functions for n-ary relational fact modeling and entity relatedness evaluation, which obtains state-of-the-art results. However, these models achieve good performance at the cost of time-consuming pairwise operations and a large number of parameters. Additionally, higher-arity relations further complicate the issue, which conflicts with the linear time and space requirements for KB modeling [5]. Thus, existing works do not provide an expressive and efficient solution for n-ary relational KB modeling, and it is still an open problem to be addressed.

On the other hand, although tensor decomposition has been proved to be very powerful in binary relational KBs by both the state-of-art results [2] and theoretical guarantees on full expressiveness [22, 49], few works is adopting such a technique for n-ary relational KB modeling. A possible way is to extend current tensor decomposition models from the binary case to the n-ary case, while direct extensions yield serious issues. First, several existing models [22, 43] leverage some customized operations for scoring functions design, while these operations are constrained on binary relations, which are not able to be applied in n-ary relations. Second, powerful tensor decomposition models [2] introduce exponential model complexity with the increase of arity, which cannot be applied in large-scale KBs.



(a) Binary Relational KB (b) Ternary Relational KB (c) Mixed-arity Relational KB

Fig. 1. The illustration of n-ary relational KBs.

To solve n-ary relational KB modeling problem as well as tackle the above challenges, we *Ge*neralize *T*ensor *D*ecomposition in a two-step modeling framework, termed as GETD. In the first step, the GETD framework focuses on modeling n-ary relational KBs with single-arity relations (only n-ary relations, as shown in Figure 1(a) and (b)), referred to as GETD-S. More specifically, GETD-S follows TuckER [2], the state-of-the-art model in binary relational KBs, and applies Tucker decomposition [44] for the scoring function design in the n-ary case. However, the core tensor in Tucker decomposition grows exponentially with the arity, and excessively complex models usually overfit. Thus, motivated by the model compression benefits of **tensor ring (TR)** decomposition [55] in computer vision [35, 48], the core tensor is further decomposed by TR for linear complexity. In the second step, the GETD framework extends GETD-S for modeling the KBs with mixed-arity relations (not exceeding n-ary relations, as shown in Figure 1(c)),[1] referred to as GETD-M. A group of TR tensors are introduced as the base space for TR decomposition with different arity cases. Specifically, based on the arity of relation, GETD-M chooses different tensors from the group to decompose corresponding core tensors, while, the final scoring function is unified with the entity and relation embeddings shared across different arities. Furthermore, since most KBs only provide positive observations, we also generalize the existing negative sampling technique for efficiently training on the GETD framework. Theoretically, we prove that our GETD framework is *fully expressive* to represent various true and false facts in n-ary relational KBs. Through extensive experiments on both synthetic and real-world datasets, we demonstrate the expressiveness and effectiveness of our proposed framework on n-ary relational KB modeling.

The main contributions of this article are summarized as follows:

—We investigate tensor decomposition for n-ary relational KB modeling, and identify the bottleneck of directly extending existing binary relational models to the n-ary case, including the binary relation constrained scoring function and exponential model complexity.
—We propose GETD, a generalized tensor decomposition framework for n-ary relational KBs. The framework integrates Tucker decomposition with TR decomposition, where linear complexity is guaranteed for modeling the KBs with single-arity relations. Furthermore, we extend our GETD framework to address the problem of modeling the KBs with mixed-arity relations. To the best of our knowledge, GETD is the first framework that leverages tensor decomposition techniques for n-ary relational KB modeling.
—We give theoretical analysis on our proposed framework and prove that our framework is fully expressive, which is able to represent all types of relations, and can completely separate true facts from false ones. We also generalize the negative sampling technique from the binary to the n-ary case.

[1]In the following, n-ary relational KBs refer to KBs with mixed-arity relations if not specified.

—We conduct extensive experiments on both synthetic and real-world datasets, and the results show that GETD improves state-of-the-art solutions by 11–26% and 4–7% for n-ary relational KBs with single-arity and mixed-arity relations, respectively. Moreover, GETD achieves close and even better performance on two binary relational benchmarks compared with state-of-the-art solutions.

The rest of this article is organized as follows. Section 2 introduces the background of tensor decomposition and notations. Section 3 gives a systematic review on the related works of KB modeling. After that, the framework of GETD and theoretical analyses are presented in Sections 4 and 5, respectively. Section 6 evaluates the performance on representative KB datasets and provides extensive analyses. In light of our results, this article is concluded in Section 7.

2 Background and Notation

2.1 Tensors and Notations

A tensor is a multi-order array, which generalizes the scalar (0th-order tensor), the vector (1st-order tensor), and the matrix (2nd-order tensor) to higher orders. We represent scalars with lowercase letters, vectors with boldface lowercase letters, matrices with boldface uppercase letters, and higher-order tensors with boldface Euler script letters. For indexing, let $\boldsymbol{a}_i$ denote the ith column of a matrix $\boldsymbol{A}$, $x_{i_1 i_2 \cdots i_p}$ denote the $(i_1, i_2, \cdots, i_p)$-th element of a higher-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_p}$, where I_i is the dimensionality of the ith mode. Especially, given a 3rd-order tensor $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the i_2-th lateral slice matrix of $\boldsymbol{\mathcal{Z}}$ is denoted by $\boldsymbol{Z}(i_2)$ in the size of $I_1 \times I_3$, a.k.a., $\boldsymbol{Z}_{:i_2:}$ where the colon indicates all elements of a mode.

As for the operation on tensors, $\circ$ represents the vector outer product, and $\times_i$ represents the tensor i-mode product. $\langle \cdot \rangle$ represents the multi-linear dot product, written as $\langle \boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_p \rangle = \sum_i a_{1,i} a_{2,i} \cdots a_{p,i}$. $trace\{\cdot\}$ is the matrix trace operator, written as $trace\{\boldsymbol{A}\} = \sum_i a_{ii}$. More details about these operations and tensor properties can be referred to [24]. The related notations frequently used in this article are listed in Table A1 in appendix.

2.2 Tucker Decomposition

Tucker decomposition was initially proposed for three-order tensor decomposition [44]. It can be generalized to a higher order, which decomposes a higher-order tensor into a set of factor matrices and a relatively small core tensor. Given a pth-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_p}$, Tucker decomposition can be denoted as:

$$\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{A}^{(1)} \times_2 \boldsymbol{A}^{(2)} \times_3 \cdots \times_p \boldsymbol{A}^{(p)} = \sum_{j_1=1}^{J_1} \sum_{j_2=1}^{J_2} \cdots \sum_{j_p=1}^{J_p} g_{j_1 j_2 \cdots j_p} \boldsymbol{a}_{j_1}^{(1)} \circ \boldsymbol{a}_{j_2}^{(2)} \circ \cdots \circ \boldsymbol{a}_{j_p}^{(p)}, \quad (1)$$

where $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{J_1 \times \cdots \times J_p}$ is the *core tensor*, J_q is the rank of qth mode, and $\{\boldsymbol{A}^{(q)} \mid \boldsymbol{A}^{(q)} \in \mathbb{R}^{I_q \times J_q}\}_{q=1}^{p}$ is the set of *factor matrices*. Usually, $J_1, \cdots, J_p$ are smaller than $I_1, \cdots, I_p$. Thus the number of parameters is reduced compared with the approximated tensor $\boldsymbol{\mathcal{X}}$.

2.3 TR Decomposition

Although Tucker decomposition approximates a higher-order tensor with fewer parameters, the number of parameters scales exponentially to the tensor order. TR decomposition [55], on the other hand, represents a higher-order tensor by a sequence of 3rd-order latent tensors multiplied circularly. Given a pth-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_p}$, TR decomposition can be expressed in an

element-wise form as:

$$x_{i_1 i_2 \cdots i_p} \approx trace\{Z_1(i_1)Z_2(i_2)\cdots Z_p(i_p)\} = trace\left\{\prod_{q=1}^{p} Z_q(i_q)\right\}, \tag{2}$$

where $\{\mathcal{Z}_q \,|\, \mathcal{Z}_q \in \mathbb{R}^{r_q \times I_q \times r_{q+1}}, r_1 = r_{p+1}\}_{q=1}^{p}$ is the set of TR latent tensors and $Z_q(i_q)$ is in the size of $\mathbb{R}^{r_q \times r_{q+1}}$ accordingly. For convenience, we also denote the above TR decomposition as $\boldsymbol{TR}(\mathcal{Z}_1, \cdots, \mathcal{Z}_q)$. Especially, the size of latent factors, concatenated and denoted by $\boldsymbol{r} = [r_1, r_2, \cdots, r_p]$ is called *TR-ranks*.

3 Related Work

We introduce in this section the related works for KB modeling, which are classified into two categories of binary relational KBs and n-ary relational KBs.

3.1 Binary Relational KB Modeling

Basically, existing approaches embed entities and relations into low-dimensional vector spaces and define a scoring function with embeddings to measure if a given fact is true or false. Based on the scoring function design, the typical works in binary relational KBs can be categorized into three groups: translational distance models [6, 19, 26, 50], neural network models [8, 37, 39], and tensor decomposition models [2, 22, 34, 43, 52].

Translational distance models measure the entity distance after a translational operation carried out by the relation [47], and various translational operations are exploited for distance-based scoring functions [6, 19, 26, 50]. However, most translational distance models are found to have restrictions on relations [22, 47], thus can only represent part of relations.

Neural network models [8, 37, 39] subtly design the scoring function with various neural network structures, which always require a great many parameters to completely represent all relations [22, 47], increasing training complexity and impractical for large-scale KBs.

With solid theory and great performance, tensor decomposition models are more prevalent methods. In this aspect, the KB modeling is framed as a 3rd-order binary tensor completion problem with the KB tensor $\mathcal{X} \in \mathbb{R}^{n_r \times n_e \times n_e}$, where each element corresponds to a triple, one for true facts while zero for false/missing facts respectively. Thus, various tensor decomposition models are proposed to approximate the 3rd-order tensor $\mathcal{X}$. For example, traditional **canonical polyadic (CP)** decomposition [16] assigns two different factor matrices to entity embeddings at head and tail of triplets, and one factor matrix to relation embeddings:

$$\mathcal{X} = \sum_{r=1}^{R} \boldsymbol{a}_r^{\text{rel}} \circ \boldsymbol{a}_r^{\text{head}} \circ \boldsymbol{a}_r^{\text{tail}}, \tag{3}$$

where R is the rank of tensor decomposition and $\boldsymbol{a}_r^{\text{rel}} \in \mathbb{R}^{n_r}$, $\boldsymbol{a}_r^{\text{head}} \in \mathbb{R}^{n_e}$, and $\boldsymbol{a}_r^{\text{tail}} \in \mathbb{R}^{n_e}$ are column vectors from embedding matrices of relation, head, and tail, respectively.

Similarly, DistMult [52] uses CP decomposition [16] with the equivalence of head and tail embeddings for the same entity, however, fails to capture the asymmetric relation:

$$\mathcal{X} = \sum_{r=1}^{R} \boldsymbol{a}_r^{\text{rel}} \circ \boldsymbol{a}_r^{\text{ent}} \circ \boldsymbol{a}_r^{\text{ent}}, \tag{4}$$

where $\boldsymbol{a}_r^{\text{ent}}$ is column vector from embedding matrix of entity.

Furthermore, SimplE [22] takes advantage of the inverse of relations to address the asymmetric relation, with an inverse relation for each original relation proposed:

$$\mathcal{X} = \sum_{r=1}^{R} \boldsymbol{a}_r^{\text{rel}} \circ \boldsymbol{a}_r^{\text{head}} \circ \boldsymbol{a}_r^{\text{tail}} + \boldsymbol{a}_r^{\text{rel\_inv}} \circ \boldsymbol{a}_r^{\text{tail}} \circ \boldsymbol{a}_r^{\text{head}}, \tag{5}$$

where $\boldsymbol{a}_r^{\text{rel\_inv}}$ is column vector from embedding matrix of inverse relation.

ComplEx [43] leverages complex-valued embeddings for solution, in which the entity embeddings used at head and tail are conjugate:

$$\mathcal{X} = \text{Re}\left(\sum_{r=1}^{R} \boldsymbol{a}_r^{\text{rel}} \circ \boldsymbol{a}_r^{\text{ent}} \circ \bar{\boldsymbol{a}}_r^{\text{ent}}\right), \tag{6}$$

where $\text{Re}(\cdot)$ denotes the real part, and $\bar{\boldsymbol{a}}_r^{\text{ent}}$ is the complex conjugate of $\boldsymbol{a}_r^{\text{ent}}$ [25].

Recently, Tucker decomposition [44] is adopted in TuckER [2] for KB modeling and achieves the state-of-the-art performance. Compared with former works only using entity and relation embeddings to capture the knowledge in KBs, TuckER additionally introduces the core tensor to model interactions between entities and relations, which further improves the expressiveness:

$$\mathcal{X} = \mathcal{G} \times_1 \boldsymbol{A}^{\text{rel}} \times_2 \boldsymbol{A}^{\text{ent}} \times_3 \boldsymbol{A}^{\text{ent}}, \tag{7}$$

where $\mathcal{G} \in \mathbb{R}^{R \times R \times R}$ is the core tensor, $\boldsymbol{A}^{\text{rel}} \in \mathbb{R}^{n_r \times R}$ and $\boldsymbol{A}^{\text{ent}} \in \mathbb{R}^{n_e \times R}$ are embedding matrices of relation and entity, respectively.

According to the discussion, generalizing tensor decomposition is promising for n-ary relational KB modeling.

3.2 N-ary Relational KB Modeling

Existing works on n-ary relational KB modeling can be categorized into two classes based on the scoring function: translational distance models [51, 53] and neural network models [14, 36].

The translational distance models of m-TransH [51] and RAE [54] are the first series of works in this field. Based on the distance translation idea, m-TransH is proposed by extending TransH [50] for the n-ary case, where entities are all projected onto the relation-specific hyperplane, and the scoring function is defined by the weighted sum of projection results. RAE further improves m-TransH with the relatedness assumption that, the likelihood of two entities co-participating in a common n-ary relational fact is important for plausibility measure. MLP is utilized to model the relatedness and coupled into the scoring function. Since these models are directly extended from the binary case, the restrictions on relations are also inherited with limited representation capability to KBs.

On the other hand, NaLP [14] was the first to introduce a neural network approach for n-ary relational KB modeling. In NaLP, the entity embeddings of an n-ary relational fact are initially processed through a convolutional layer for feature extraction, followed by a **fully connected network (FCN)** that models the overall relatedness, producing an evaluation score as output. Recent works, such as HINGE [36] and NeuInfer [13], build upon the principles established by NaLP and achieve state-of-the-art performance. HINGE emphasizes the primary information encoded in the triplet component of n-ary relational facts, utilizing **convolutional neural networks (CNNs)** for this purpose. Similarly, StarE [12] and CoRelatE [17] apply **graph convolutional networks (GCNs)** [37, 45] to model triplets; however, they fall short of capturing the entire n-ary relational fact. A recent work TransEQ [27] combines GCNs and traditional scoring functions for n-ary relational KB modeling. These neural network models typically involve a large number

of parameters and require time-consuming pairwise operations, rendering the training process intractable.

As previously discussed, tensor decomposition is a potential solution for n-ary relational KB modeling, while directly extending current binary relational tensor decomposition models to the n-ary case is challenging with various bottlenecks. First, most CP-based models achieve great performance mainly due to carefully designed scoring functions with customized operations. For instance, to model all types of relations, the relation inverse in SimplE [22] and complex-valued embeddings in ComplEx [43] are all binary relation-constrained operations, which cannot find equivalents when it comes to the n-ary case. Second, some direct extensions introduce tremendous parameters like TuckER [2] to the n-ary case with exponential model complexity, which is impractical and easily affected by noise [22, 43]. Besides, other models like DistMult [52] force the relation to be symmetric, thus are not able to completely represent n-ary relational KBs. HypE [11] explores DistMult with convolution to the n-ary case, but the interaction between entities and relations is not fully captured, leading to inferior empirical performance. Besides, RAM [30] identifies the role semantics in n-ary relations and develops a multilinear product-based scoring function for n-ary relational KB modeling. GETD [28, 29] applied Tucker decomposition and TR decomposition to model the interaction in n-ary relational KBs, where the scoring function is written as:

$$\mathcal{X} = \boldsymbol{TR}(\mathcal{Z}_1, \cdots, \mathcal{Z}_k) \times_1 \boldsymbol{A}^{\text{rel}} \times_2 \boldsymbol{A}^{\text{ent}} \times_3 \cdots \times_{n+1} \boldsymbol{A}^{\text{ent}}, \tag{8}$$

where $\mathcal{Z}_1, \cdots, \mathcal{Z}_k$ are TR latent tensors.

S2S [9] develops automated learning to search sparse tensor decomposition for n-ary relational KBs, which however is time-consuming due to search process:

$$\mathcal{X} = \mathcal{Z}^n \times_1 \boldsymbol{A}^{\text{rel}} \times_2 \boldsymbol{A}^{\text{ent}} \times_3 \cdots \times_{n+1} \boldsymbol{A}^{\text{ent}}, \tag{9}$$

where $\mathcal{Z}^n$ is the sparse core tensor to be searched.

Compared with the preliminary version [28], this version investigates the comprehensive research problem of n-ary relational KB modeling with both single-arity relations and mixed-arity relations. Especially, this one comprises a substantial amount of additional algorithmic, theoretical, and experimental efforts and contributions. First, we design a system framework, which further extends preliminary work for modeling KBs with mixed-arity relations. The complete framework makes our work more practical and systematic. Second, we provide the detailed theoretical proof of full expressiveness to the proposed GETD framework including single-arity and mixed-arity cases, which is valuable for future research on KB modeling expressiveness. Last but not least, we further include two new datasets and baselines and report a series of experimental results that examine the performance of GETD for mix-arity case and visualize parameters for better understanding on n-ary relational KBs.

4 GETD: Design and Framework

Borrowing the concept of n-ary relation [7, 10], the n-ary relational fact can be defined as follows:

Definition 4.1 (n-ary Relational Fact). Given an n-ary relational KB with the set of relations $\mathcal{R}$ and the set of entities $\mathcal{E}$, an n-ary relational fact is an $(n+1)$-tuple $(i_r, i_1, i_2, \cdots, i_n) \subseteq \mathcal{R} \times \underbrace{\mathcal{E} \times \cdots \times \mathcal{E}}_{n}$,

where $\mathcal{R}$ and $\mathcal{E}$ are called relation domain and entity domain.

Especially, i_j is the jth entity to the relation i_r, belonging to the jth entity domain. In the binary case of (i_r, i_1, i_2), i_1 and i_2 are head entity and tail entity, and i_r is the relation, respectively. For example, in the mixed-arity relational KB in Figure 1(c), there exist a binary relational fact of (**friend**, Hinton, LeCun) as well as a ternary relational fact of (**receiveAwardIn**, Hinton, Turing Award, 2018).

Then, the n-ary relational KB modeling can be specified as **knowledge base completion (KBC)** problem, which is defined as follows,

Problem 1 *(n-ary Relational KBC). Given an incomplete n-ary relational KB* $\mathcal{S} = \{(i_r, i_1, i_2 \cdots, i_n)\}$, *the n-ary relational KBC problem aims to infer missing facts with* $\mathcal{S}$.

In the following, we first provide insight on directly applying Tucker decomposition for n-ary relational KB modeling, then introduce our proposed GETD framework for modeling single-arity KBs and mixed-arity KBs, respectively. Furthermore, the negative sampling and corresponding training procedure are presented.

4.1 Rethinking Tucker for KB Modeling

From the point view of tensor completion, an n-ary relational KB can be represented as a binary valued $(n+1)$th-order KB tensor $\mathcal{X} \in \{0,1\}^{n_r \times n_e \times n_e \times \cdots \times n_e}$ ($n_r = |\mathcal{R}|, n_e = |\mathcal{E}|$), whose 1st-mode is the relation mode, while the other modes are entity modes in the n-ary relational fact. $x_{i_r i_1 i_2 \cdots i_n}$ equal to one means the specific n-ary relational fact is true, and zero for false/missing. Accordingly, KB modeling is to calculate the approximated low-rank scoring tensor $\hat{\mathcal{X}} \in \mathbb{R}^{n_r \times n_e \times n_e \times \cdots \times n_e}$ with $\|\mathcal{X} - \hat{\mathcal{X}}\|$ minimized, where the parameters are learnt *via* gradient descent.

Especially, the state-of-the-art binary relational model TuckER [2] can be directly extended to the n-ary case termed as n-TuckER, with relation embedding matrix $\boldsymbol{R} = \boldsymbol{A}^{(1)} \in \mathbb{R}^{n_r \times d_r}$, and entity embedding matrix $\boldsymbol{E}$ that is equivalent for each mode entities, i.e., $\boldsymbol{E} = \boldsymbol{A}^{(2)} = \cdots = \boldsymbol{A}^{(n+1)} \in \mathbb{R}^{n_e \times d_e}$, where d_r and d_e represent the dimensionality of relation and entity embedding vectors respectively. The scoring function is defined as,

$$\phi(i_r, i_1, i_2, \cdots, i_n) = \hat{x}_{i_r i_1 i_2 \cdots i_n} = \mathcal{W} \times_1 \boldsymbol{r}_{i_r} \times_2 \boldsymbol{e}_{i_1} \times_3 \boldsymbol{e}_{i_2} \cdots \times_{n+1} \boldsymbol{e}_{i_n}, \tag{10}$$

where $\mathcal{W} \in \mathbb{R}^{d_r \times d_e \times d_e \times \cdots \times d_e}$ is the $(n+1)$th-order core tensor, $\boldsymbol{r}_{i_r}$ and $\{\boldsymbol{e}_i\}_{i=i_1}^{i_n}$ are the rows of $\boldsymbol{R}$ and $\boldsymbol{E}$ representing the relation and the entity embedding vectors. Such a straightforward design inevitably leads to a model complexity of

$$O(n_e d_e + n_r d_r + d_e^n d_r),$$

which grows exponentially with d_e. Besides the unacceptable complexity in parameters and increased training difficulty, n-TuckER also faces the dilemma that, excessively complex models are easily affected by noise and prone to overfitting, leading to poor testing performance [22, 43].

4.2 The GETD-S Model: Single-Arity KB Modeling

In this part, we introduce the GETD-S model for n-ary relational KBs with single-arity relations, which is shown in Figure 2.

Despite the model complexity and overfitting, leveraging the $(n+1)$th-order core tensor to capture the interaction of entities and relations is instructive that the similarity between entities and relations is encoded in core tensor element. Such Tucker interaction way ensures the strong expressive capability of representing various facts in KBs. It can be envisioned that a model with the Tucker interaction way as well as low complexity is promising for n-ary relational KB modeling.

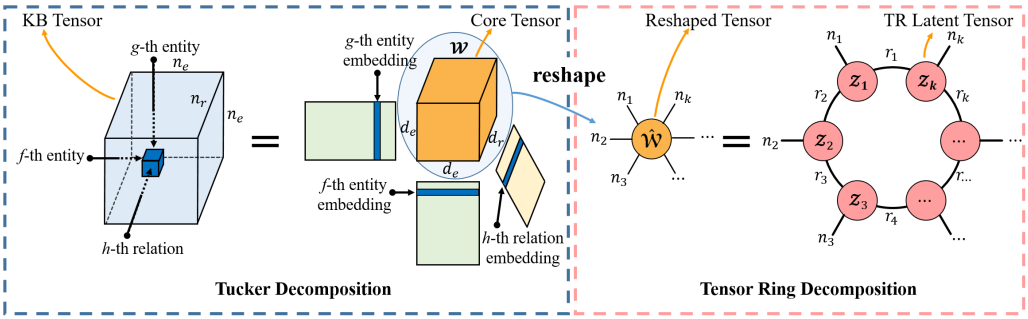


Fig. 2. The illustration of GETD-S model for single-arity KBs, consisting of two major components: the outer layer with Tucker decomposition and the inner layer with TR decomposition.

To achieve this, TR decomposition draws our attention that a higher-order tensor can be decomposed by quite a few parameters in 3rd-order latent tensor sequences. This motivates the general construction of GETD-S.

First, in the outer layer of GETD-S, the original KB tensor is decomposed *via* Tucker decomposition following Equation (10), which reserves Tucker interaction way as well as strong expressiveness. Subsequently, to further reduce the parameter complexity of TR decomposition for the core tensor, in the inner layer, the intermediate core tensor $\mathcal{W}$ is flexibly reshaped to a kth-order tensor $\hat{\mathcal{W}} \in \mathbb{R}^{n_1 \times \cdots \times n_k}$, where k denotes the order of reshaped tensor with $\prod_{i=1}^{k} n_i = d_e^n d_r$ ($k \geq n+1$) satisfied. Then TR decomposes the reshaped tensor $\hat{\mathcal{W}}$ into k latent 3rd-order tensors $\{\mathcal{Z}_i \mid \mathcal{Z}_i \in \mathbb{R}^{r_i \times n_i \times r_{i+1}}, r_1 = r_{k+1}\}_{i=1}^{k}$, reducing the number of parameters. The illustration of GETD-S is shown in Figure 2 (in $n=2$ case). Specifically, the left part of the figure depicts the construction of outer layer with Tucker decomposition, while the right part presents the TR construction procedure of inner layer. The corresponding expression is,

$$\hat{w}_{j_1 j_2 \cdots j_k} = trace\{Z_1(j_1) Z_2(j_2) \cdots Z_k(j_k)\}. \tag{11}$$

Overall, the scoring function of GETD-S can be expressed as:

$$\begin{aligned} \phi(i_r, i_1, i_2, \cdots, i_n) &= \hat{\mathcal{W}} \times_1 \boldsymbol{r}_{i_r} \times_2 \boldsymbol{e}_{i_1} \times_3 \boldsymbol{e}_{i_2} \cdots \times_{n+1} \boldsymbol{e}_{i_n} \\ &= \boldsymbol{TR}(\mathcal{Z}_1, \cdots, \mathcal{Z}_k) \times_1 \boldsymbol{r}_{i_r} \times_2 \boldsymbol{e}_{i_1} \times_3 \boldsymbol{e}_{i_2} \cdots \times_{n+1} \boldsymbol{e}_{i_n}. \end{aligned} \tag{12}$$

The model complexity of GETD-S is

$$O(n_e d_e + n_r d_r + k n_{\max}^3), \quad \text{s. t. } n_{\max} = \max_{i=1,\cdots,k} n_i,$$

which is much lower than n-TuckER. $n_{\max}$ is the maximum TR-rank and discussed in detail later. Accordingly, with Tucker interaction way as well as low model complexity, GETD-S not only guarantees strong expressive capability, but also avoids the overfitting problem with many parameters, which improves the testing performance. Note that due to the fixed order ($n+1$, and n represents a constant) of core tensor $\mathcal{W}$, GETD-S can only model n-ary relational KBs with single-arity relations.

4.3 The GETD-M Model: Mixed-Arity KB Modeling

To further model n-ary relational KBs with mixed-arity relations like the example in Figure 1(c), here we present the GETD-M model, the extension of GETD-S.

As described before, the outer layer of GETD-S is constructed based on n-TuckER. According to Equation (10), if we extend n-TuckER to mixed-arity KBs, the corresponding scoring function for

m-ary relational facts in the mixed-arity KB can be defined as:

$$\phi^{(m)}(i_r, i_1, i_2, \cdots, i_m) = \mathcal{W}^{(m)} \times_1 \boldsymbol{r}_{i_r} \times_2 \boldsymbol{e}_{i_1} \times_3 \boldsymbol{e}_{i_2} \cdots \times_{m+1} \boldsymbol{e}_{i_m}, \ \forall m = 2, \cdots, M, \tag{13}$$

where m is the arity of the considered relation, $\mathcal{W}^{(m)} \in \mathbb{R}^{d_r \times d_e \cdots \times d_e}$ is $(m+1)$-th order core tensor, and M is the maximum arity of relations in the mixed-arity KB. However, the order of core tensor $\mathcal{W}$ in n-TuckER is fixed to $n+1$ when applied in n-ary relational KBs, which is not able to process mixed-arity relational facts simultaneously. Besides, initializing multiple core tensors of different orders for mixed-arity KBs is also impractical, which ignores the mutual effect across different arities and further leads to inferior empirical performance. Thus, the key challenge of extending GETD-S to the mixed-arity case is dealing with variable order core tensors with mutual effect considered.

Motivated by the inner layer of GETD-S that TR decomposition represents an nth-order tensor by n 3rd-order tensors, we introduce M 3rd-order TR tensors in GETD-M, termed as TR tensor group $\{\mathcal{Z}_i \,|\mathcal{Z}_1 \in \mathbb{R}^{r^* \times d_r \times r^*}, \mathcal{Z}_{i \neq 1} \in \mathbb{R}^{r^* \times d_e \times r^*}\}_{i=1}^{M+1}$. The TR-ranks of TR tensors are set to r^* for flexible decomposition. Subsequently, with TR decomposition applied, GETD-M utilizes first three TR tensors in the group to recover the 3rd-order core tensor $\mathcal{W}^{(2)}$ for binary relational facts, and utilizes first four TR tensors in the group to recover the 4th-order core tensor $\mathcal{W}^{(3)}$ for 3-ary relational facts, etc. The whole group of TR tensors are utilized to recover the $(M+1)$-th order core tensor $\mathcal{W}^{(M)}$ for M-ary relational facts.

Figure 3 illustrates the extension idea of GETD-M, which indicates the solution to mixed-arity KBs. The node represents a tensor whose order is denoted by the number of edges and the number beside the edges specifies the size of each mode. The connection between two nodes denotes a tensor product on a specific mode. Since we consider the mixed-arity KBs with a maximum arity of M, there are M KB tensor to be decomposed, as shown in blue circles. First, GETD-M leverages Tucker decomposition to decompose the KB tensors for different arities. Moreover, GETD-M initializes TR tensor group with M TR tensors, which are chosen for TR decomposition based on the relation arity. Especially, the TR tensors used for lower-ary relational facts are the subset of that for higher-ary relational facts. This operation can reduce parameter complexity as well as learn shared information across relational facts in different arities. Therefore, TR tensor group as well as shared embeddings in the outer layer encode the mutual effect across different arities.

Thus, the scoring function of GETD-M for m-ary relational facts in the mixed-arity KB can be expressed as follows:

$$\begin{aligned}
&\phi^{(m)}(i_r, i_1, \cdots, i_m) \\
&= \mathcal{W}^{(m)} \times_1 \boldsymbol{r}_{i_r} \times_2 \boldsymbol{e}_{i_1} \times_3 \boldsymbol{e}_{i_2} \cdots \times_{m+1} \boldsymbol{e}_{i_m} \\
&= \mathcal{TR}(\mathcal{Z}_1, \cdots, \mathcal{Z}_{m+1}) \times_1 \boldsymbol{r}_{i_r} \times_2 \boldsymbol{e}_{i_1} \times_3 \boldsymbol{e}_{i_2} \cdots \times_{m+1} \boldsymbol{e}_{i_m}, \ \forall m = 2, \cdots, M.
\end{aligned} \tag{14}$$

Note that the reshape operation in Equation (12) is omitted for flexible processing across different arities of relational facts. The model complexity of GETD-M is

$$O(n_e d_e + n_r d_r + d_r r^{*2} + M d_e r^{*2}), \tag{15}$$

which is comparable to GETD-S, and discussed in later section. Hence, with TR tensor group developed, the GETD framework successfully extends GETD-S to GETD-M for mixed-arity KBs, which is quite practical for KB modeling.

4.4 Model Training

In this part, we generalize negative sampling in binary relational KB modeling to the n-ary case and present the modeling training algorithm for better understanding.

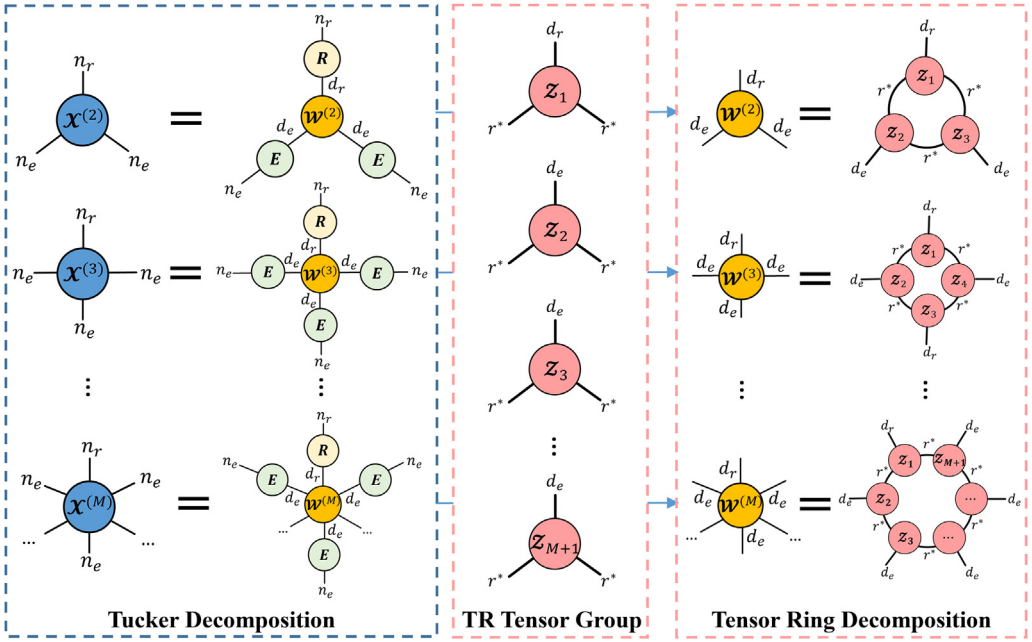


Fig. 3. The illustration of GETD-M model for mixed-arity KBs, consisting of TR tensor group and TR decomposition for different arity cases. M is the maximum arity of mixed-arity KB. r^* is the rank of TR tensors.

In KBs, we usually only have positive observations, i.e., which relation exists among different sets of entities. Thus, even with the designed scoring functions, we cannot train an embedding model due to a lack of negative observations. In binary relational KBs, given a positive triplet (i_r, i_1, i_2), good candidates of negative samples [33] are

$$\mathcal{N}_{(i_r,i_1,i_2)} \equiv \mathcal{N}^{(1)}_{(i_r,i_1,i_2)} \cup \mathcal{N}^{(2)}_{(i_r,i_1,i_2)} \equiv \{(i_r, \bar{i_1}, i_2) \notin \mathcal{S} \,|\, \bar{i_1} \in \mathcal{E}\} \cup \{(i_r, i_1, \bar{i_2}) \notin \mathcal{S} \,|\, \bar{i_2} \in \mathcal{E}\}, \tag{16}$$

where the symbols of $\bar{i_1}$ and $\bar{i_2}$ denote the negations to i_1 and i_2, respectively.

Following this, for one positive n-ary relational fact $x := (i_r, i_1, i_2, \cdots, i_n)$, n groups of negative candidates are generated from corresponding n entity domains, defined as

$$\mathcal{N}_x \equiv \bigcup_{m=1}^{n} \mathcal{N}_x^{(m)} \equiv \bigcup_{m=1}^{n} \{(i_r, \cdots, \bar{i_m}, \cdots) \notin \mathcal{S} \,|\, \bar{i_m} \in \mathcal{E}\}. \tag{17}$$

As for the training loss, motivated by the gains of multiclass log-loss in binary relational KB modeling [21, 25], we extend the loss to the n-ary case with all candidates in Equation (16) simultaneously considered, defined as

$$\mathcal{L}_x = \sum_{j=1}^{n} -\log \left[e^{\phi(x)} / (e^{\phi(x)} + \sum_{y \in \mathcal{N}_x^{(j)}} e^{\phi(y)}) \right]. \tag{18}$$

Then both GETD-S and GETD-M are trained in a mini-batch way, where all observed facts and each entity domain therein are considered for training. Algorithm 1 presents the pseudo-code of the training algorithm for GETD-S. With the embedding dimensions and TR-ranks as input, the embeddings of entities and relations as well as TR latent tensors, are randomly initialized before training in line 1. During the training, line 3 samples a mini-batch $\mathcal{S}_{\text{batch}}$ of size m_b, in which each

Algorithm 1: Training Algorithm for GETD-S

Input: training set $\mathcal{S} = \{(i_r, i_1, i_2 \cdots, i_n)\}$,
reshaped tensor order k, TR-ranks $\boldsymbol{r}$,
entity/relation embedding dimension d_e/d_r;

1 initialize embeddings $\boldsymbol{E}$, $\boldsymbol{R}$, TR tensors $\{\mathcal{Z}_i\}_{i=1}^k$;
2 **for** $t = 1, 2, \cdots, n_{epoch}$ **do**
3 sample a mini-batch $\mathcal{S}_{\text{batch}} \subseteq \mathcal{S}$ of size m_b;
4 $\mathcal{L} \leftarrow 0$;
5 **for** $x := (j_r, j_1, j_2, \cdots, j_n) \in \mathcal{S}_{\text{batch}}$ **do**
6 construct negative sample set $\mathcal{N}_x$;
7 $\phi(x) \leftarrow$ compute the score using (12);
8 $\mathcal{L}_x \leftarrow$ compute the loss using (18)
9 $\mathcal{L} \leftarrow \mathcal{L} + \mathcal{L}_x$;
10 update parameters of embeddings and TR latent tensors w.r.t. the gradients using $\nabla\mathcal{L}$;

Output: embeddings $\boldsymbol{E}$, $\boldsymbol{R}$ and TR tensors $\{\mathcal{Z}_i\}_{i=1}^k$.

observation is considered for training in lines 4–10. Specifically, for each n-ary relational fact in $\mathcal{S}_{\text{batch}}$, the algorithm constructs the negative candidates following Equation (17), as shown in line 6. Then, the score of the observation as well as the negative candidates are computed using Equation (12) in line 7, which are further utilized to compute the multiclass log-loss with Equation (18) in lines 8–9. Finally, the algorithm updates the model parameters according to the loss gradients. Similarly, Algorithm A1 in Appendix A presents the pseudo-code of the training algorithm for GETD-M. The predefined TR tensor group $\{\mathcal{Z}_i\}_{i=1}^k$ is initialized in line 1. The major difference with Algorithm 1 is that training data with different mixed-arity relational facts should be sampled according to the arity, as shown in lines 3–4.

5 Theoretical Understanding

In this section, we present a detailed analysis on the complexity of the GETD framework for the single-arity and mixed-arity cases, and further prove the full expressiveness.

5.1 Complexity Analysis

According to the GETD-S model description in Section 4.2, the entity and relation embeddings cost $O(n_e d_e + n_r d_r)$ parameters. Since each TR latent tensor is 3rd-order with $\mathcal{Z}_i \in \mathbb{R}^{r_i \times n_i \times r_{i+1}}$ and TR-rank r_i is usually smaller than n_i, the k TR latent tensors cost $O(k n_{\max}^3)$ parameters in sum, where $n_{\max} = \max_{i=1,\cdots,k} n_i$. Thus, the model complexity of GETD-S is obtained as $O(n_e d_e + n_r d_r + k n_{\max}^3)$. Moreover, due to the constraint $\prod_{i=1}^k n_i = d_e^n d_r$ in GETD-S, if TR latent tensors are in the same shape, we can obtain the equation of $n_{\max} = (d_e^n d_r)^{1/k}$. When applied in large-scale KBs with over thousands of entities [4, 41], GETD-S with high reshaped tensor order (larger k) derives that $k n_{\max}^3 \ll n_e d_e$, which reduces to the linear model complexity of $O(n_e d_e + n_r d_r)$ to KB sizes. Besides, GETD-S also retains the efficiency benefits of tensor mode product in linear time complexity.

As for the mixed-arity KB with the maximum arity of M, the GETD-M model generalizes the TR latent tensors in GETD-S with TR tensor group, which costs $O(d_r r^{*2} + M d_e r^{*2})$ parameters. In practical, the TR-rank r^* is set much smaller than d_e and the maximum arity is no more than 9, which derives that $d_r r^{*2} + M d_e r^{*2} \ll n_e d_e$. Hence, GETD-M achieves linear model complexity to KB sizes.

Table 1. A Comparison of State-of-the-Art N-ary Relational KB Modeling Approaches

Type	Model	Fully Expressive	Mixed-Arity	Model Complexity
Translational Distance Model	m-TransH [51]	✘	✓	$O(n_e d_e + 2n_r d_e)$
	RAE [54]	✘	✓	$O(n_e d_e + 2n_r d_e)$
Neural Network Model	NaLP [14]	✘	✓	$O(n_e d_e + Mn_r d_r)$
	HINGE [36]	✘	✓	$O(n_e d_e + Mn_r d_r)$
TensorDecompositionModel	n-CP	✓	✓	$O(Mn_e d_e + n_r d_e)$
	n-DistMult	✘	51	$O(n_e d_e + n_r d_r)$
	n-TuckER	51	✘	$O(n_e d_e + n_r d_r + d_e^M d_r)$
	GETD-S (ours)	✓	✘	$O(n_e d_e + n_r d_r + k n_{\max}^3)$
	GETD-M (ours)	✓	✓	$O(n_e d_e + n_r d_r + d_r r^{*2} + M d_e r^{*2})$

The mixed-arity indicates whether a model can model KBs with mixed-arity relations. n_e and n_r are the number of entities and relations, while d_e and d_r are the dimensionality of entity and relation embeddings, respectively. M is the maximum arity of relations in KBs. k and $n_{\max}$ are the number and maximum size of TR latent tensors. r^* is the TR-rank.

In Table 1, we compare the GETD framework with state-of-the-art n-ary relational modeling approaches based on expressiveness, mixed-arity representation, and model complexity. The table shows that both translational distance models and neural network models lack full expressiveness, which limits their learning capacity and results in inferior performance. In contrast, the GETD framework is fully expressive, as demonstrated later in the article. It's important to note that we only present the complexity of relation and entity-specific parameters here, while additional parameters in neural network models are omitted. Among the compared models, n-CP, n-DistMult, and n-TuckER extend CP [16], DistMult [52], and Tucker [2], respectively. n-CP requires different embeddings for a single entity across various domains, resulting in a complexity of $O(Mn_e d_e + n_r d_e)$. Meanwhile, n-DistMult can only model symmetric relations and does not achieve full expressiveness. n-TuckER incurs exponential complexity due to its higher-order core tensor, which is impractical for large-scale n-ary relational KBs. Additionally, the large number of parameters in n-TuckER makes it susceptible to overfitting, as shown in the experimental results in Section 6. In contrast, both models in the GETD framework exhibit linear complexity with respect to KB size, making the GETD framework superior in terms of both model complexity and expressiveness.

5.2 Full Expressiveness

For KB modeling, a model is fully expressive if for any ground truth over all entities and relations, there exist embeddings that accurately separate the true n-ary relational facts from the false ones, i.e., the model can recover any given KB tensors by the assignment of entity and relation embeddings [2, 22, 43, 47]. In this part, we prove the full expressiveness of both models in the GETD framework.

The full expressiveness guarantees the completeness of KB modeling. Especially, if a model is not fully expressive, it means that the model can only represent a part of KBs with prior constraints, which leads to unwarranted inferences [15]. For instance, DistMult is not fully expressive, and forces relations to be symmetric, i.e., it can represent KBs with only symmetric relations [22], while KBs with asymmetric and inverse relations cannot be completely represented. Thus, the upper bound of learning capacity from a not fully expressive model is low. In contrast, fully expressive models enable KB representation with various types of relations, fully representing the knowledge.

Formally, we have the following theorem to establish the full expressiveness of GETD-S in the GETD framework.

Theorem 5.1. *For any ground truth over entities $\mathcal{E}$ and relations $\mathcal{R}$ in an n-ary relational KB with single-arity relations, there exists a GETD-S model that represents that ground truth.*

Moreover, the following theorem is introduced to establish full expressiveness of GETD-M in the GETD framework.

Theorem 5.2. *For any ground truth over entities $\mathcal{E}$ and relations $\mathcal{R}$ in an n-ary relational KB with mixed-arity relations, there exists a GETD-M model that represents that ground truth.*

The full expressiveness is a desired property for model learning and generalization. Especially, in practice, the embedding dimensionality for modeling KBs is much smaller than the bound stated above, because the KB tensor follows a certain structure instead of random assignment [2]. The proofs of two theorems are provided in Appendix B.

6 Experiments and Results

In this section, we first describe the experimental setup of datasets, metrics, baselines and implementation details. Then, we evaluate the proposed GETD-S model on single-arity KBs with model performance and parameter impact investigated. Furthermore, we evaluate the proposed GETD-M model on mixed-arity KBs from similar aspects and embedding visualization.

6.1 Experimental Setup

6.1.1 Datasets. We evaluate our model with two real datasets for mixed-arity KBs, one synthetic dataset and two real datasets for single-arity KBs, as well as two benchmark datasets on binary relations, which are introduced as follows.

WikiPeople [14]. Mixed-arity KB dataset. This is a public n-ary relational dataset extracted from Wikidata concerning entities of type *human*.

JF17K [54]. Mixed-arity KB dataset. This is a public n-ary relational dataset from Freebase.

WikiPeople-3/4. Single-arity KB dataset. Due to the sparsity of higher-arity (≥ 5) facts in WikiPeople, we filter out all 3-ary and 4-ary relational facts therein, named as WikiPeople-3 and WikiPeople-4, respectively.

JF17K-3/4. Single-arity KB dataset. Similar to WikiPeople, the higher-arity facts in JF17K are also sparse, thus we filter out all 3-ary and 4-ary relational facts therein, named JF17K-3 and JF17K-4, respectively.

Synthetic10. To assess the relationship between the number of parameters and overfitting, we construct the toy dataset across 3-ary and 4-ary relations, named Synthetic10-3 and Synthetic10-4, whose KB tensors are randomly generated by CP decomposition with tensor rank equal to one [24]. There are only 10 entities and 2 relations in Synthetic10.

WN18 [4]. Binary KB dataset is a subset of WordNet with lexical relations between words.

FB15k [5]. Binary KB dataset is a subset of Freebase, a database of real-world facts including films, sports, etc.

The train/valid/test sets of WikiPeople provided in [14], of WN18 and FB15k provided in [6] are used for evaluation. Since JF17K lacks a validation set, we randomly select 20% of the training set as validation. The facts in other datasets are randomly split into train/valid/test sets by a proportion of 8:1:1. The datasets statistics of mixed-arity KBs are summarized in Table 3, while the ones of single-arity KBs are summarized in Table 2.

6.1.2 Metrics. We evaluate models on the KBC task with two standard metrics: **mean reciprocal rank (MRR)** and Hits@k, $k \in \{1, 3, 10\}$ [2, 14, 22]. For each testing n-ary relational fact, one of its entities is removed and replaced by all entities in $\mathcal{E}$, leading to $|\mathcal{E}|$ tuples, which are scored by the model. The entities in all entity domains are tested. The ranking of the testing fact is obtained by

Table 2. Dataset Statistics

Dataset	Arity	#Entities	#Relations	#Train	#Valid	#Test
WikiPeople	2–9	47,765	702	305,725	38,223	38,281
JF17K	28,645	2–6	322	61,104	15,275	24,568
WikiPeople-3	3	12,270	66	20,656	2,582	2,582
WikiPeople-4	4	9,528	50	12,150	1,519	1,519
JF17K-3	3	11,541	104	27,635	3,454	3,455
JF17K-4	4	6,536	23	7,607	951	951
Synthetic10-3	3	10	2	400	50	50
Synthetic10-4	4	10	2	1,200	150	150
WN18	2	40,943	18	141,442	5,000	5,000
FB15k	2	14,951	1,345	483,142	50,000	59,071

Here "-3" and "-4" denote the 3-ary and 4-ary relational KB datasets, respectively.

Table 3. Detailed Arity-based Statistics of Mixed-arity KBs

Dataset	Arity	#Arity-2	#Arity-3	#Arity-4	#Arity-5+
WikiPeople	2–9	337,914	25,820	15,188	3,307
JF17K	2–6	54,627	34,544	9,509	2,267

Here "#Arity-5+" denotes the number of facts whose relation is 5-ary and above.

sorting evaluation scores in descending order. MRR is the mean of the inverse of rankings over all testing facts, while Hits@k measures the proportion of top k rankings. Both metrics are in filtered setting [4]: the ranking of the testing fact is calculated among facts not appeared in train/valid/test sets. The aim is to achieve high MRR and Hits@k.

6.1.3 Baselines. We compare the GETD framework with the n-ary relational KB modeling baselines:

— *RAE* [54] is a translational distance model, extending m-TransH [51] with relatedness further considered.
— *NaLP* [14] is a neural network model, which utilizes 1D CNNs and FCNs for modeling KBs.
— *n-CP* is an extension of CP decomposition [16].
— *n-TuckER* is an extension of TuckER [2] with Tucker decomposition utilized.
— *n-DistMult* is an extension of DistMult [52] with symmetric scoring function design.
— *HINGE* [36] is a combination model of 2D CNNs and FCNs with primary information in triplets considered.
— *RAM* [30] identifies role semantics in n-ary relations and applies multilinear product to capture semantic interactions.
— *S2S* [9] uses neural architecture search techniques to search the scoring function for KB modeling, which consumes much more time and computation resources.

Note that this article firstly proposes n-CP, n-TuckER, and n-DistMult for n-ary relational KB modeling. Besides, we compare GETD-S with state-of-the-art models in binary relational KBs, including TransE [6], DistMult [52], ConvE [8], ComplEx [43], SimplE [22], and TuckER [2].

6.1.4 Implementation. The implementation code is available at Github.[2] For experimental fairness, we fix entity and relation embedding sizes of the GETD framework and tensor decomposition baselines, while the sizes of other baselines follow their original settings for model performance. As for GETD-S model in single-arity KBC, in WikiPeople-3 and JF17K-3, we set entity and relation embedding sizes to $d_e = d_r = 50$, reshaped tensor order to $k = 4$, TR-ranks and TR latent tensor dimensions to $r_i = n_i = 50$, while due to the quite smaller numbers of entities and relations, the settings in 4-ary relational datasets are $d_e = d_r = 25$, $k = 5$, $r_i = n_i = 25$. As for GETD-M model in mixed-arity KBC, the embedding sizes as well as TR latent tensor dimensions on WikiPeople are set to $d_e = d_r = n_i = 50$ with TR-ranks r^* equal to 10, and the settings on JF17K are $d_e = d_r = n_i = r^* = 25$. Due to the hardware limitation, we learn a small-scale GETD-M for 5-ary and above relational facts on both datasets with $d_e = d_r = n_i = r^* = 10$. Besides, batch normalization [18] and dropout [40] are used to control overfitting. All hyperparameters except embedding sizes are tuned with *Optuna* [1], a Bayesian hyperparameter optimization framework, and the search space of learning rate is $[0.0001, 0.1]$ with learning rate decay chosen from $\{0.9, 0.995, 1\}$, and dropout ranges from 0.0 to 0.5. Each model is evaluated with 50 groups of hyperparameter settings. GETD and other tensor decomposition models are trained with Adam [23] using early stopping based on validation set MRR with no improvement for 10 epochs. As for other baselines, we use the optimal settings reported in their original papers.

6.2 Single-Arity KB Modeling Evaluation

In this part, we evaluate the KBC performance of the GETD framework especially GETD-S model on single-arity KBs. Moreover, the overfitting phenomenon as well as parameter influence are investigated. The binary relational KBC performance is also studied.

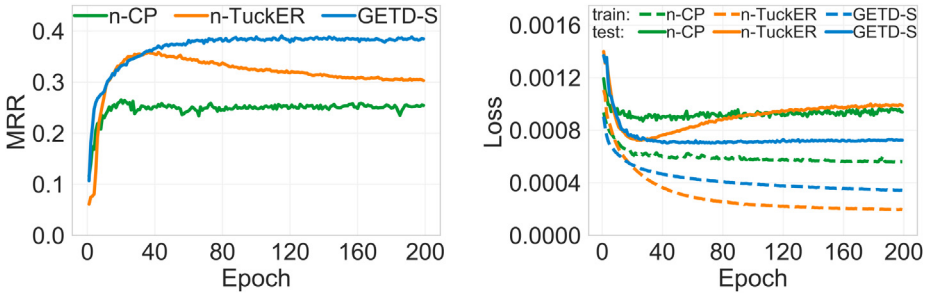
6.2.1 Single-Arity KBC. Table 4 presents the single-arity KBC results across two datasets for 3-ary and 4-ary relational KBs. The results indicate that GETD-S performs comparably to S2S on all metrics, with a performance gap of less than 0.01 between the two models across benchmarks. GETD-S is more efficient due to its use of tensor decomposition rather than resource-intensive neural architecture search techniques. Tensor decomposition models like n-CP and n-TuckER consistently outperform the translational distance model RAE and the neural network model NaLP. For instance, on the JF17K dataset, GETD-S improves the MRR by 0.22 and Hits@1 by 55% for 3-ary relational facts compared to NaLP, while improvements for 4-ary facts are 0.09 and 12%, respectively. In the WikiPeople dataset, GETD-S enhances MRR by 0.07 and Hits@1 by 25% for WikiPeople-3, and by 0.04 and 12% for WikiPeople-4. These significant improvements highlight the strong expressive capability of the proposed tensor decomposition models. In contrast, the recent RAM model struggles to generalize to single-arity benchmarks, performing worse than many tensor decomposition models. The performance gap between RAM and GETD-S exceeds 0.03 across benchmarks, primarily due to RAM's reliance on a shared latent space across arities, which does not apply to single-arity contexts. The robust performance of GETD-S on WikiPeople demonstrates its ability to address practical KB challenges, such as data incompleteness, insertion, and updating. However, improvements on 4-ary relational facts are less pronounced than those on 3-ary facts, likely due to the smaller training datasets available in JF17K-4 and WikiPeople-4 compared to JF17K-3 and WikiPeople-3.

As for the tensor decomposition models, n-CP is relatively weak due to the difference of embeddings in different entity domains [43], while GETD-S and n-TuckER capture the interaction between entities and relations with TR latent tensors or core tensors. On the other hand, GETD-S also outperforms n-TuckER owing to the simplicity with much fewer parameters, while the parameter-cost

[2] https://github.com/liuyuaa/GETD

Table 4. Single-arity KBC Results on WikiPeople and JF17K Datasets

Model	WikiPeople-3			WikiPeople-4			JF17K-3			JF17K-4		
	MRR	H@10	H@1	MRR	H@10	H@1	MRR	H@10	H@1	MRR	H@10	H@1
RAE [54]	0.239	0.379	0.168	0.150	0.273	0.080	0.505	0.644	0.430	0.707	0.835	0.636
NaLP [14]	0.301	0.445	0.226	0.342	0.540	0.237	0.515	0.679	0.431	0.719	0.805	0.673
n-CP	0.330	0.496	0.250	0.265	0.445	0.169	0.700	0.827	0.635	0.787	0.890	0.733
n-TuckER	0.365	0.548	0.274	0.362	0.570	0.246	0.727	0.852	0.664	0.804	0.902	0.748
RAM [30]	0.339	0.480	0.269	0.191	0.285	0.138	0.685	0.821	0.617	0.788	0.881	0.733
S2S [9]	0.386	0.559	0.299	0.391	0.600	0.270	0.740	0.860	0.676	0.822	0.924	0.761
GETD-S	0.373	0.558	0.284	0.386	0.596	0.265	0.732	0.856	0.669	0.810	0.913	0.755



(a) MRR v.s. epoch.

(b) Loss v.s. epoch.

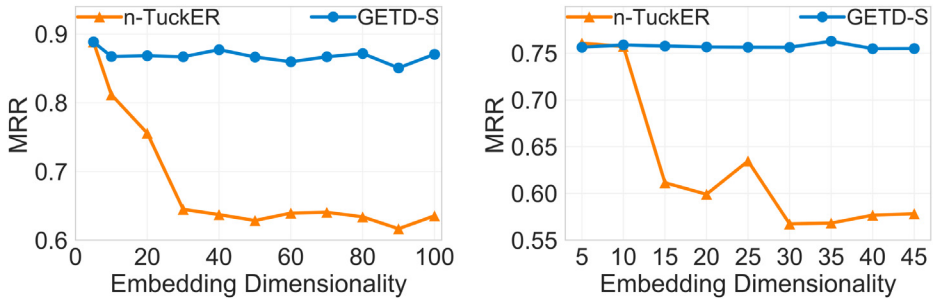
Fig. 4. Overfitting in n-TuckER observed from MRR (left) and loss (right). Evaluated on WikiPeople-4.

core tensor in n-TuckER increases the complexity of optimization and further overfits. Without the early stopping trick, the performance of n-TuckER seriously degrades and quickly overfits, which is shown in the following. Besides, we run GETD-S on the largest dataset WikiPeople-3 with a Titan-XP GPU. An epoch training takes about 28 s and total training takes 1 h, while inference takes only 5 s. Overall, the results show efficiency and robustness of GETD-S for single-arity KBC.

6.2.2 Overfitting Phenomenon. Models like n-TuckER with a large amount of parameters easily overfit to the training data, impairing the testing performance. To verify this, we cast the early stopping trick in three tensor decomposition models, and test if there exists the overfitting phenomenon using WikiPeople-4. Accordingly, the training curves in terms of MRR and loss are plotted in Figure 4(a) and (b), respectively.

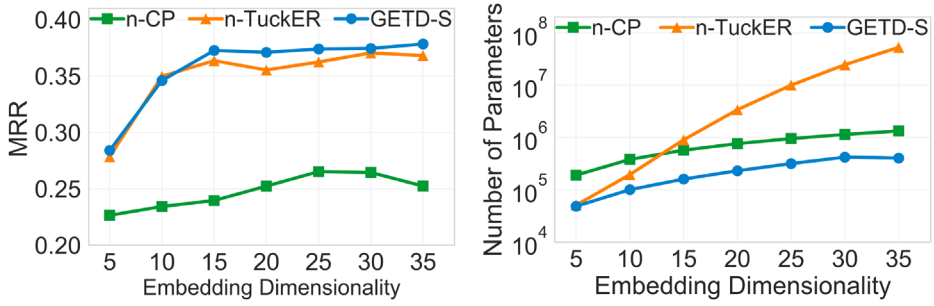
From the results, we can clearly observe the overfitting phenomenon in training process of n-TuckER. In Figure 4(a), as training going on, the MRR of n-TuckER increases first and then quickly decreases, while the MRR of the other two models increases to convergence. Moreover, GETD-S outperforms n-CP due to its strong expressive power. As for loss curves, the train losses of all three models keep decreasing, while the test loss of n-TuckER increases after 20 epochs of training, compared with the convergence in GETD-S and n-CP test loss curves. It is mainly caused by the model complexity that, the numbers of parameters in GETD-S and n-CP are 0.4 million and 0.9 million, while 10 million in n-TuckER.

To further reveal the relationship between the overfitting and the number of parameters, we evaluate the MRR for different embedding sizes on Synthetic10 in Figure 5. The early stopping is cast, and the MRR after 200 epochs of training are reported. The results of n-TuckER in both 3-ary and 4-ary relational datasets show that, increasing of embedding sizes results in a quality fall in the case of MRR, which means overfitting of n-TuckER. This phenomenon is mainly caused by low-rank property of KB tensors. Taking $d_e = d_r = 50$ in Synthetic10-3 as an example, the



(a) Synthetic10-3. (b) Synthetic10-4.

Fig. 5. MRR for n-TuckER and GETD-S for different embedding sizes. Evaluated on Synthetic10-3 (left) and Synthetic10-4 (right).



(a) MRR v.s. d_e, d_r. (b) #Paramters v.s. d_e, d_r.

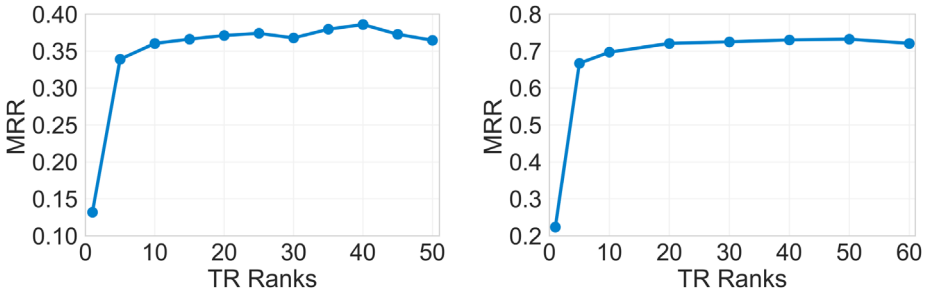
Fig. 6. MRR (left) and number of parameters (right) under embedding sizes. Evaluated on WikiPeople-4.

core tensor in n-TuckER costs $50 \times 50 \times 50 \times 50 = 6.25$ million parameters, while the TR latent tensors in GETD-S cost only $4 \cdot 1 \times 50 \times 1 = 200$ parameters with TR-ranks equal to one. Therefore, using n-TuckER with large embedding sizes to approximate the low-rank KB tensors is intractable and prone to overfitting. However, for general KBs, embedding sizes should be large enough for strong expressive power, which is a contradiction. In comparison, GETD-S is capable of coping with overfitting and expressiveness together based on embedding sizes as well as TR-ranks, which is much more flexible. The flexibility as well as expressiveness thus support the great performance of GETD-S in Table 4.

6.2.3 Influence of Parameters. Since the embedding sizes are important factors to link prediction models with expressiveness [2, 8, 43], while TR-ranks, as well as the reshaped tensor order are unique hyper-parameters of GETD-S, determining the model complexity and performance, now we investigate the impacts of these parameters.

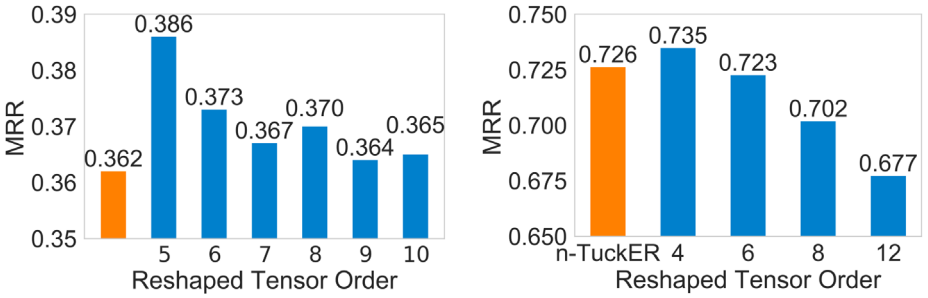
Influence of Embedding Sizes d_e, d_r. The MRR and the number of parameters of three tensor decomposition models under different embedding sizes are evaluated on WikiPeople-4 with TR-ranks equal to embedding sizes. The results are plotted in Figure 6.

According to Figure 6(a), GETD-S always outperforms n-TuckER and n-CP. The MRR of GETD-S increases globally with the increase of embedding sizes, and gradually becomes smooth. While the MRR of n-TuckER is not stable even with large embedding sizes when early stopping applied. For example, at embedding size 35, GETD-S increases MRR by 49% for n-CP, and 2.8% for n-TuckER. Moreover, the MRR of GETD-S reaches 0.372 at embedding size 15, which is better than



(a) WikiPeople-4. (b) JF17K-3.

Fig. 7. MRR of GETD on WikiPeople-4 (left) and JF17K-3 (right) under different TR-ranks.



(a) WikiPeople-4, $d_e = d_r = 25$. (b) JF17K-3, $d_e = d_r = 64$.

Fig. 8. MRR of GETD-S on WikiPeople-4 (left) and JF17K-3 (right) under different reshaped tensor orders.

the performance of n-TuckER at embedding size over 30. On the other hand, GETD-S uses the least parameters in three models, which is shown in Figure 6(b). For embedding size 30, n-TuckER costs 24 million parameters with core tensor using $30^5 = 24.3$ million, n-CP costs 1.14 million parameters, while GETD-S only costs 0.42 million parameters with TR latent tensors using $5 \cdot 30^3 = 0.13$ million, 1.7% of n-TuckER parameters. The results are in accord with complexity analysis in Section 5.1 and further indicate that GETD-S with relatively small embedding sizes is able to obtain good performance, which can be applied for large-scale KBs.

Influence of TR-ranks r. Since the TR-ranks can largely determine the number of TR latent tensor parameters, and make GETD-S model more flexible, we reveal the relationship between link prediction performance and TR-ranks on WikiPeople-4 and JF17K-3, as shown in Figure 7. From the results, we can observe that the link prediction performance is affected only when TR-ranks are very small (less than 5), indicating that GETD-S is not sensitive to TR-ranks. When TR-ranks vary from 20 to 60 on JF17K-3, the MRR is rather stable, and a similar trend can be found on WikiPeople-4. This implies that TR tensors with TR-ranks about 20 are often enough to capture the latent interactions between entities and relations for given datasets. Based on this, the number of parameters for GETD-S can be further reduced to control model complexity for large-scale KBs.

Influence of Reshaped Tensor Order k. As a key step of connecting Tucker and TR in GETD-S, the effect of reshaped tensor order is investigated on WikiPeople-4 and JF17K-3, exhibited in Figure 8.

For WikiPeople-4, the embedding size is set to 25 and thus the original core tensor is $\mathcal{W} \in \mathbb{R}^{25\times25\times25\times25\times25}$, leading to the 5th-order reshaped tensor $\hat{\mathcal{W}} \in \mathbb{R}^{25\times25\times25\times25\times25}$, the 6th-order reshaped tensor $\hat{\mathcal{W}} \in \mathbb{R}^{5\times25\times25\times25\times25\times5}$, etc. It can be observed that, GETD-S with different orders

Table 5. Binary Relational KBC Results on WN18 and FB15k

Model	WN18				FB15k			
	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
TransE [6]	0.454	0.934	0.823	0.089	0.380	0.641	0.472	0.231
DistMult [52]	0.822	0.936	0.914	0.728	0.654	0.824	0.733	0.546
ConvE [8]	0.943	0.956	0.946	0.935	0.657	0.831	0.723	0.558
ComplEx [43]	0.941	0.947	0.945	0.936	0.692	0.840	0.759	0.599
SimplE [22]	0.942	0.947	0.944	0.939	0.727	0.838	0.773	0.660
RotatE [42]	0.949	0.959	0.952	0.944	0.797	0.884	0.830	0.746
TuckER [2]	0.953	0.958	0.955	0.949	0.795	0.892	0.833	0.741
GETD-S	0.948	0.954	0.950	0.944	0.824	0.888	0.847	0.787

Results of TransE and DistMult are copied from [43]. Other results are copied from the corresponding original papers [2, 8, 22, 43].

of reshaped tensors always achieves higher MRR compared with n-TuckER, and the best one increases MRR by 0.024, which is decomposed by 5 TR latent tensors. Overall, the expressive power of GETD-S decreases with the increase of reshaped tensor order.

As for JF17K-3, the embedding size is set to 64 so that the reshaped tensor is cubic, i.e., each mode is in the same size [24]. For example, the 6th-order reshaped tensor becomes $\hat{\mathcal{W}} \in \mathbb{R}^{16\times16\times16\times16\times16\times16}$, and the size of each mode for 8th-order tensor becomes 4. Similarly, GETD-S with the least order of reshaped tensor achieves the highest MRR. Moreover, Figure 8(b) clearly shows the negative correlation between MRR and reshaped tensor order, which is in accord with the above results. This phenomenon is mainly because the higher order involves more TR latent tensors, increasing the optimization complexity. On the other hand, since GETD-S requires that $\prod_{i=1}^{k} n_i = d_e^n d_r$, the higher order k also means the smaller $n_{\max}$, which reduces the number of parameters. Thus, the reshaped tensor order in GETD-S should be appropriately determined considering both link prediction performance and model complexity.

6.2.4 Binary Relational KBC. To investigate the robustness as well as representation capability of our proposed GETD-S model, we evaluate the performance of GETD-S model on WN18 and FB15k. The experimental settings are the same as in n-ary relational link prediction. The embedding sizes of GETD-S are set to 200, which is similar to the setting in TuckER [44]. The reshaped tensor order k is 3, TR-ranks r_i and TR latent tensor dimensions n_i are set to 50 and 200, respectively.

Table 5 summarizes the results of GETD-S and the state-of-the-art models on two datasets. According to the results, GETD-S achieves the second-best performance on WN18, with a quite small MRR gap of 0.005 to TuckER. Moreover, TR latent tensors in GETD-S costs $3\cdot50\times200\times50 = 1.5$ million parameters, only 1/8 of core tensor parameters in TuckER ($200 \times 200 \times 200 = 8$ million). Thus, GETD-S is able to obtain better performance with larger embedding sizes but similar number of parameters. As for FB15k, GETD-S outperforms all state-of-the-art models, and increases MRR by 0.03. Also, GETD-S increases the toughest metric Hits@1 by 4% on FB15k. These results demonstrate that, GETD-S is robust and works well in representing KBs with different arity relations.

6.3 Mixed-Arity KB Modeling Evaluation

In this part, we evaluate the KBC performance of GETD-M on mixed-arity KBs. The parameter study and embedding visualization are also presented for better understanding.

Table 6. Mixed-arity KBC Results on WikiPeople and JF17K

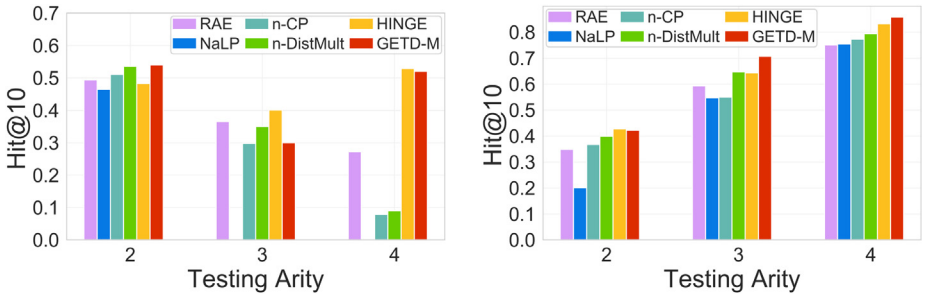
Model	WikiPeople				JF17K			
	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
RAE [54]	0.253	0.463	0.343	0.117	0.396	0.561	0.433	0.312
NaLP [14]	0.338	0.466	0.364	0.272	0.310	0.450	0.334	0.239
n-CP	0.313	0.451	0.372	0.223	0.400	0.542	0.431	0.324
n-DistMult	0.318	0.478	0.391	0.213	0.452	0.599	0.482	0.375
HINGE [36]	0.333	0.477	0.361	0.259	0.472	0.618	0.490	0.397
RAM [30]	0.380	0.539	0.445	0.279	0.539	0.690	0.573	0.463
S2S [9]	0.372	0.533	0.439	0.277	0.528	0.690	0.570	0.457
GETD-M	0.345	0.510	0.415	0.237	0.489	0.643	0.521	0.409

Results of NaLP, HINGE, RAM, and S2S on WikiPeople are copied from original papers.

6.3.1 Mixed-Arity KBC. A comparison of the testing performance of GETD-M and state-of-the-art n-ary modeling approaches is shown in Table 6. According to the results, GETD-M model outperforms most baselines on both two datasets. On the other hand, with the latent space design and role semantics further identified, RAM [30] outperforms baselines and our proposed GETD-M model. Nevertheless, it should be mentioned that both latent space and role semantics can be combined with GETD-M for performance improvement, which is beyond the scope of this work. Specifically, the proposed designs in RAM focus on the input of scoring function, while the tensor decomposition-based scoring function of GETD-M can be leveraged with these inputs for interaction modeling. As for S2S [9], it performs worse than RAM even with NAS techniques used. Overall, we can find that the SOTA model RAM [30] achieves better performance on mixed-arity benchmarks but fails to generalize to single-arity benchmarks without information sharing, as compared in Table 4. An interesting future work is combining the input designs of RAM with GETD-M for performance improvement. As for the AutoML-based model S2S [9], due to its resource-consuming property, it is unfair for performance comparison with GETD-M. Moreover, S2S only achieves comparable performance with GETD-S on single-arity benchmarks, which implies the effectiveness of the proposed GETD+ framework.

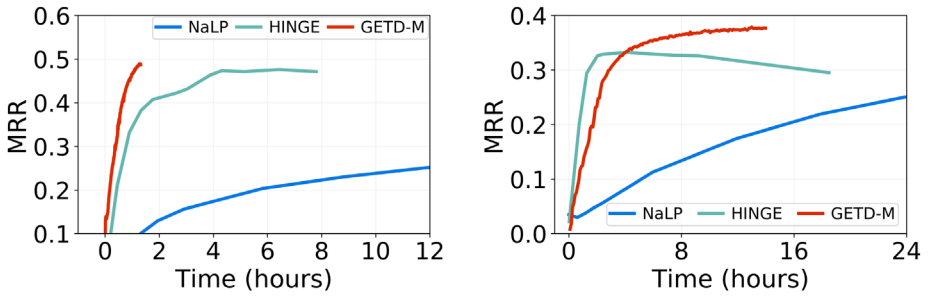
According to the design of n-CP, n-DistMult and GETD-M, these tensor decomposition models similarly share the entity and relation embeddings across arities. However, the results in Table 6 show quite large gaps (over 7% with MRR) between the performance of GETD-M and other tensor decomposition models, which owes to the gains of mutual effect encoded by the TR tensor group. Simply sharing the embeddings in n-CP and n-DistMult may not be able to overcome the noise caused by different arities of relational facts, while the TR tensor group plays a role of the base space. By selecting certain tensors in the group for different arities, GETD-M successfully captures the positive effect across arities with noise removed.

To analyze performance across different arities, we present the Hits@10 breakdown for both datasets in Figure 9. Since 5-ary and higher relational facts are sparse in the testing sets, we focus on lower-arity relations, specifically 2-ary, 3-ary, and 4-ary facts. Note that the breakdown performance of NaLP on WikiPeople is not reported in [14]. The results show that GETD-M improves Hits@10 by 9% and 3% for 3-ary and 4-ary relational facts, respectively, compared to the baselines on JF17K. Additionally, the prediction performance of all models on JF17K increases with higher arity. This improvement occurs because the number of entities and relations in higher-arity relational facts is much lower, making learning easier. In contrast, WikiPeople exhibits a different trend due to its unique composition. Over 88% of the facts in WikiPeople involve binary relations, which simplifies



(a) WikiPeople. (b) JF17K.

Fig. 9. Breakdown performance across relations of different arities on WikiPeople and JF17K.



(a) WikiPeople. (b) JF17K.

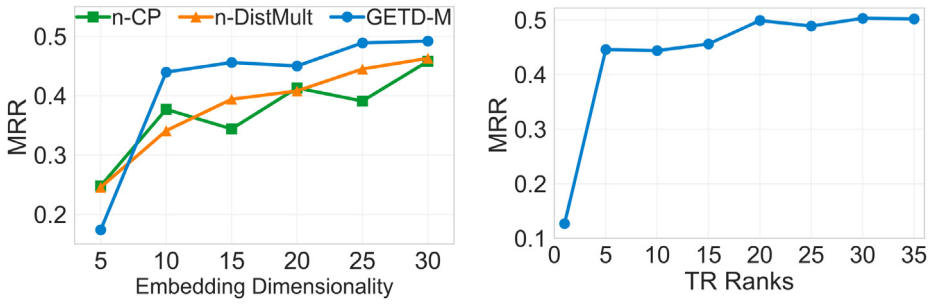
Fig. 10. Comparison on clock time of model training vs. testing MRR between GETD-M and baselines on WikiPeople and JF17K.

learning for these cases. Furthermore, 3-ary and 4-ary relational facts in WikiPeople often include entities related to time and space points, which are more challenging to model and lead to lower accuracy for higher arities. Despite this, GETD-M still achieves over a 7% increase in Hits@10 for 4-ary relational facts compared to the baselines, demonstrating its effectiveness in joint learning within mixed-arity KBs and its strong expressiveness for time and space-related points.

To evaluate the training efficiency of our proposed GETD-M model compared with other baselines, in Figure 10, we plot the clock time comparison of models (two best baselines are selected for comparison). According to the results, GETD-M consistently outperforms the baselines on both datasets and typically converges rapidly, thereby validating the effectiveness of TR decomposition for complexity reduction. However, it's worth noting that the convergence rate is also influenced by factors such as the learning rate, scheduler, implementation, and so on. As a result, GETD-M may exhibit slightly longer clock times initially compared to the baselines, as illustrated by the clock time curves of GETD-M and HINGE on JF17K.

6.3.2 Influence of Parameters. As described before, both the embedding sizes and the TR-ranks determine the expressiveness as well as model complexity of GETD-M. Therefore, we discuss the influence of these parameters in this part.

Influence of Embedding Sizes d_e, d_r. The performance of tensor decomposition models under different embedding sizes is evaluated on JF17K with TR-ranks equal to embedding sizes. The results are shown in Figure 11(a). The MRR of three models increases with the increase of embedding sizes and converges at large embedding sizes. GETD-M outperforms baselines significantly when



(a) Embedding sizes d_e , d_r. (b) TR-ranks $\boldsymbol{r}^*$.

Fig. 11. Influence of embedding sizes and TR-ranks for GETD-M.

embedding sizes are over 10. For instance, GETD-M increases MRR by over 6% for baselines at embedding size 30. Especially, GETD-M with embedding size 10 reaches comparable performance to n-CP and n-DistMult with embedding size 30, which demonstrates the strong expressiveness.

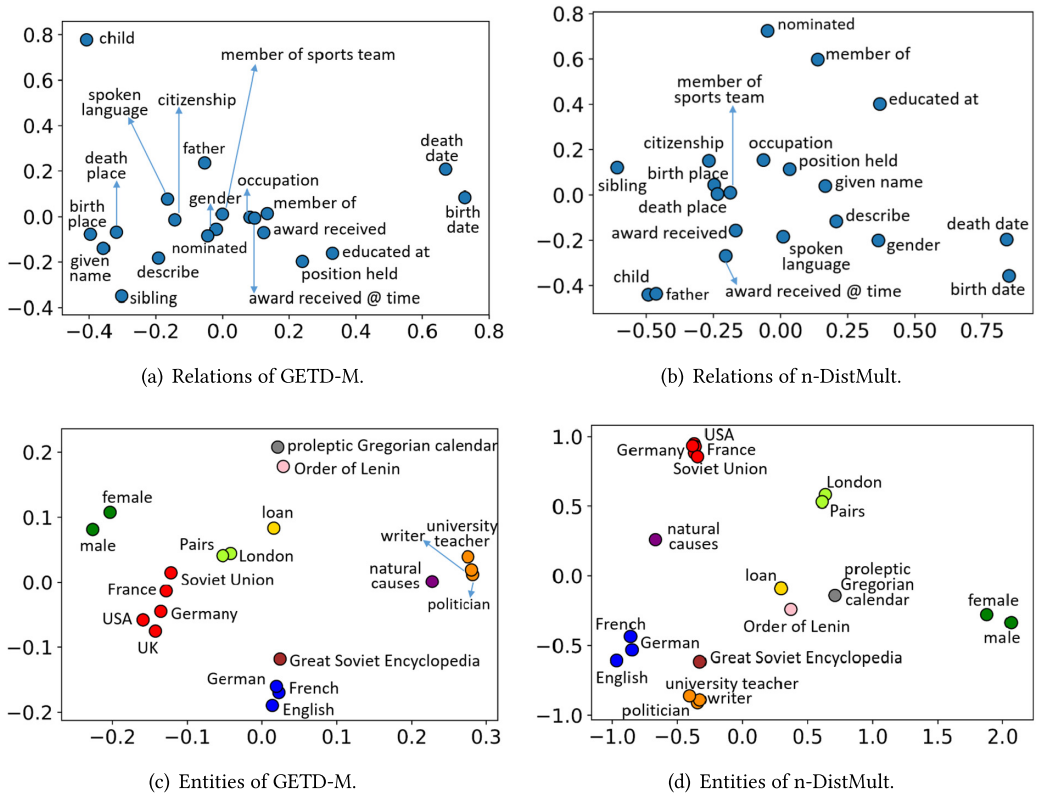
Influence of TR-ranks r^*. Considering the model complexity of GETD-M in Equation (15), the parameter cost has quadratic relationship with TR-ranks $\boldsymbol{r}^*$. Thus, in Figure 11(b), we show the performance of GETD-M with different TR-ranks evaluated on JF17K. The obtained results are in accord with the results of GETD in Figure 7. The MRR increases slowly when TR-ranks exceed 5, which indicates that even small TR-ranks for GETD-M are able to recover the core tensor with complex interactions between entities and relations. Also, we can reduce TR-ranks of GETD-M for storage space saving and training acceleration.

According to the results in Figure 11, embedding sizes play a much more important role in GETD-M than TR-ranks. A possible explanation is that TR-ranks only determine the inner layer with TR tensor groups, while embedding sizes determine the representation space of entities, relations and the recovered core tensor, which are closely related to the interaction and expressiveness.

6.3.3 Embedding Visualization. To further confirm the expressiveness of GETD-M, the top 20 relation/entity embeddings of GETD-M and n-DistMult on WikiPeople are visualized in Figure 12 through **principal component analysis (PCA)**.

From the first two figures we can observe that most similar relations such as birth/death place, award received (@ time), and date of birth/death are correctly clustered by both models. However, the anti-symmetric relation pair of father and child is visualized differently. Specifically, the child $(-0.4, 0.8)$ and the father $(0.0, 0.2)$ relations are opposed spatially in GETD-M, while assigned closely $(-0.5, -0.4)$ in n-DistMult. As described before, DistMult based models cannot represent anti-symmetric relations due to the same entity embeddings at different positions. Hence, n-DistMult learns the close embeddings for child/father relations due to their sharing semantics, which leads to wrong representation in practical use. In contrast, GETD-M correctly learns the opposite relations, which further validates the strong expressiveness.

As for entity embedding visualization in Figure 12(c) and (d), most entities with similar semantics are clustered correctly by both models. Note that there are three groups of entities with close connections, the country group (red), the capital group (green), and the language group (blue). Moreover, these groups of embeddings are all shown in the third quadrant of GETD-M embedding visualization, while shown in a distance in n-DistMult embedding visualization. Considering the co-occurrence of these entities in KBs, the learnt embeddings of GETD-M successfully capture this. The visualization results indicate robustness and expressiveness of GETD-M in mixed-arity KBs.



(a) Relations of GETD-M.

(b) Relations of n-DistMult.

(c) Entities of GETD-M.

(d) Entities of n-DistMult.

Fig. 12. Entity and relation embedding visualization of GETD-M and DistMult through PCA on WikiPeople dataset (semantic-related entities are shown in the same color).

7 Conclusion

This work proposed GETD, a fully expressive tensor decomposition framework for n-ary relational KB modeling including the single-arity and mixed-arity cases. Based on the expressiveness of Tucker decomposition as well as the flexibility of TR decomposition, the framework designs GETD-S for single-arity KBs to capture the latent interactions between entities and relations, costing a small number of parameters. Furthermore, the framework extends to GETD-M for mixed-arity KBs. Experimental results demonstrate that the GETD framework outperforms the state-of-the-art models for n-ary relational KB modeling and achieves comparable performance on standard binary relational KB datasets.

Considering the uneven distribution of different arities of relational facts in KBs, we plan to improve the GETD framework by better modeling the mutual effect across arities. Besides, GETD only uses observed facts for KB modeling, while incorporating GETD with background knowledge such as logical rules and entity properties may bring performance enhancement.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *KDD*, 2623–2631.

[2] Ivana Balažević, Carl Allen, and Timothy M. Hospedales. 2019. TuckER: Tensor factorization for knowledge graph completion. In *EMNLP*, 5184–5193.

[3] 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.
[4] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Mach. Learn.* 94, 2 (2014), 233–259.
[5] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Irreflexive and hierarchical relations as translations. arXiv:1304.7158. Retrieved from https://arxiv.org/abs/1304.7158
[6] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*, 2787–2795.
[7] E. F. Codd. 1970. A relational model of data for large shared data banks. *Commun. ACM* 13, 6 (1970), 377–387.
[8] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In *AAAI*, 1811–1818.
[9] Shimin Di, Quanming Yao, and Lei Chen. 2021. Searching to sparsify tensor decomposition for N-ary relational data. In *WWW*, 4043–4054.
[10] Patrick Ernst, Amy Siu, and Gerhard Weikum. 2018. HighLife: Higher-arity fact harvesting. In *WWW*, 1013–1022.
[11] Bahare Fatemi, Perouz Taslakian, David Vazquez, and David Poole. 2021. Knowledge hypergraphs: Prediction beyond binary relations, In *IJCAI*, 2191–2197.
[12] Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. 2020. Message passing for hyper-relational knowledge graphs. In *EMNLP*, 7346–7359.
[13] Saiping Guan, Xiaolong Jin, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. 2020. Neuinfer: Knowledge inference on N-ary facts. In *ACL*, 6141–6151.
[14] Saiping Guan, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2019. Link prediction on N-ary relational data. In *WWW*, 583–593.
[15] Víctor Gutiérrez-Basulto and Steven Schockaert. 2018. From knowledge graph embedding to ontology embedding? An analysis of the compatibility between vector space representations and rules. arXiv:1805.10461. Retrieved from https://arxiv.org/abs/1805.10461
[16] Frank L. Hitchcock. 1927. The expression of a tensor or a polyadic as a sum of products. *J. Math. Phys.* 6, 1–4 (1927), 164–189.
[17] Yan Huang, Haili Sun, Ke Xu, Songfeng Lu, Tongyang Wang, and Xinfang Zhang. 2021. CoRelatE: Learning the correlation in multi-fold relations for knowledge graph embedding. *Knowl.-Based Syst.* 213 (2021), 106601.
[18] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *ICML*, 448–456.
[19] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *AAAI*, 985–991.
[20] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S. Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition and applications. *IEEE TNNLS* 33, 2 (2021), 494–514.
[21] Armand Joulin, Edouard Grave, Piotr Bojanowski, Maximilian Nickel, and Tomas Mikolov. 2017. Fast linear model for knowledge graph embeddings. arXiv:1710.10881. Retrieved from https://arxiv.org/abs/1710.10881
[22] Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *NeurIPS*, 4289–4300.
[23] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *ICLR*, 1-15.
[24] Tamara G. Kolda and Brett W. Bader. 2009. Tensor decompositions and applications. *SIAM Rev.* 51, 3 (2009), 455–500.
[25] Timothee Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. In *ICML*, 2863–2872.
[26] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2181–2187.
[27] Yu Liu, Shu Yang, Jingtao Ding, Quanming Yao, and Yong Li. 2024. Generalizing hyperedge expansion for hyper-relational knowledge graph modeling. arXiv:2411.06191. Retrieved from https://arxiv.org/abs/2411.06191
[28] Yu Liu, Quanming Yao, and Yong Li. 2020. Generalizing tensor decomposition for N-ary relational knowledge bases. In *WWW*, 1104–1114.
[29] Yu Liu, Quanming Yao, and Yong Li. 2020. Multiary relational knowledge base completion via tensor decomposition. In *IJCAI Tensor Network Representations in Machine Learning Workshop*, 1-4.
[30] Yu Liu, Quanming Yao, and Yong Li. 2021. Role-aware modeling for N-ary relational knowledge bases. In *WWW*, 2660–2671.
[31] Robert Logan, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife Hillary: Using knowledge graphs for fact-aware language modeling. In *ACL*, 5962–5971.
[32] 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.

[33] 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.
[34] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, 3104482–3104584.
[35] Yu Pan, Jing Xu, Maolin Wang, Jinmian Ye, Fei Wang, Kun Bai, and Zenglin Xu. 2019. Compressing recurrent neural networks with tensor ring for action recognition. In *AAAI*, 4683–4690.
[36] Paolo Rosso, Dingqi Yang, and Philippe Cudré-Mauroux. 2020. Beyond triplets: Hyper-relational knowledge graph embedding for link prediction. In *WWW*, 1885–1896.
[37] 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*, 593–607.
[38] Amit Singhal. 2012. Introducing the knowledge graph: Things, not strings. Official Google Blog, 5. Rerieved from https://blog.google/products/search/introducing-knowledge-graph-things-not/
[39] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NeurIPS*, 926–934.
[40] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J Mach Learn Res* 15, 1 (2014), 1929–1958.
[41] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *WWW*, 697–706.
[42] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2018. RotatE: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 697–706.
[43] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*, 2071–2080.
[44] Ledyard R. Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika* 31, 3 (1966), 279–311.
[45] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks. In *ICLR*, 1-15.
[46] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledge base. *Commun. ACM* 57, 10 (2014), 78–85.
[47] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. In *TKDE* 29, 12 (2017), 2724–2743.
[48] Wenqi Wang, Yifan Sun, Brian Eriksson, Wenlin Wang, and Vaneet Aggarwal. 2018. Wide compression: Tensor ring nets. In *CVPR*, 9329–9338.
[49] Yanjie Wang, Rainer Gemulla, and Hui Li. 2018. On multi-relational link prediction with bilinear models. In *AAAI*, 4227–4234.
[50] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 1112–1119.
[51] 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*, 1300–1307.
[52] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*, 1-12.
[53] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *SIGKDD*, 353–362.
[54] Richong Zhang, Junpeng Li, Jiajie Mei, and Yongyi Mao. 2018. Scalable instance reconstruction in knowledge bases via relatedness affiliated embedding. In *WWW*, 1185–1194.
[55] Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki. 2016. Tensor ring decomposition. arXiv:1606.05535. Retrieved from https://arxiv.org/abs/1606.05535

Appendices

A Notation and Training Algorithm

The notation table and training algorithm of GETD-M are provided here.

B Proofs

Now, we prove Theorem 5.1 as follows.

Proof. Now, we first introduce a few lemmas.

Algorithm A1: Training Algorithm for GETD-M

Input: training set $S = \{S_2, S_3, \cdots, S_m\}$ with $\mathcal{S}_m = \{(i_r, i_1, i_2, \cdots, i_m)\}$, TR-ranks $\boldsymbol{r}$, entity/relation embedding dimension d_e/d_r;

initialize embeddings $\boldsymbol{E}$, $\boldsymbol{R}$, TR tensors $\{\mathcal{Z}_i\}_{i=1}^{M+1}$;
for $t = 1, 2, \cdots, n_{epoch}$ **do**
  for $m = 1, 2, \cdots, M$ **do**
    sample a mini-batch $\mathcal{S}_{\text{batch}} \subseteq \mathcal{S}_m$ of size m_b;
    $\mathcal{L} \leftarrow 0$;
    for $x \coloneqq (j_r, j_1, j_2, \cdots, j_m) \in \mathcal{S}_{\text{batch}}$ **do**
      construct negative sample set $\mathcal{N}_x$;
      $\phi(x) \leftarrow$ compute the score using (14);
      $\mathcal{L}_x \leftarrow$ compute the loss using (18)
      $\mathcal{L} \leftarrow \mathcal{L} + \mathcal{L}_x$;
    update parameters of embeddings and TR latent tensors w.r.t. the gradients using $\nabla\mathcal{L}$;

Output: embeddings $\boldsymbol{E}$, $\boldsymbol{R}$ and TR tensors $\{\mathcal{Z}_i\}_{i=1}^{M+1}$.

Table A1. List of Commonly Used Notations

Symbol	Definition
$\mathcal{X}$	pth-order tensor $\in \mathbb{R}^{I_1 \times \cdots \times I_p}$
$x_{i_1 i_2 \cdots i_p}$	$(i_1, i_2, \cdots, i_p)$-th element of $\mathcal{X}$
$\mathcal{G}$	pth-order core tensor $\in \mathbb{R}^{J_1 \times \cdots \times J_p}$
$\boldsymbol{A}^{(q)}$	q-mode factor matrix $\in \mathbb{R}^{I_q \times J_q}$
$\boldsymbol{a}_j^{(q)}$	jth column vector of $\boldsymbol{A}^{(q)}$
$\mathcal{Z}_q$	qth TR latent tensor $\in \mathbb{R}^{r_q \times n_q \times r_{q+1}}$
$Z_q(i_q)$	i_q-th lateral slice matrix of $\mathcal{Z}_q$, $\in \mathbb{R}^{r_q \times r_{q+1}}$
$\boldsymbol{r} = [r_1, r_2, \cdots, r_n]$	TR-ranks
n_e, n_r	The number of entities/relations in the KB
d_e, d_r	Entity/relation embedding dimensionality
n_i	2nd-mode dimensionality of $\mathcal{Z}_i$
$\circ$	Vector outer product
$\times_p$	Tensor p-mode product
$\langle \cdot \rangle$	Multi-linear dot product
$trace\{\cdot\}$	Matrix trace operator

Lemma B.1. *For any ground truth over entities $\mathcal{E}$ and relations $\mathcal{R}$, there exists an n-TuckER model with entity embeddings of dimensionality $d_e = |\mathcal{E}|$ and relation embeddings of dimensionality $d_r = |\mathcal{R}|$ that represents that ground truth.*

Proof. Let entity and relation embedding matrices be identity matrices of $\boldsymbol{E} = \boldsymbol{I} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$ and $\boldsymbol{R} = \boldsymbol{I} \in \mathbb{R}^{|\mathcal{R}| \times |\mathcal{R}|}$, respectively. Further, we set the $w_{i_r i_1 i_2 \cdots i_n}$ of the core tensor to 1 if the fact $(i_r, i_1, i_2, \cdots, i_n)$ holds and 0 otherwise. Thus the tensor product of these entity embeddings and the relation embedding with the core tensor, accurately represents the original KB tensor. □

Lemma B.2. *Given any kth-order binary tensor* $\mathcal{W} \in \{0,1\}^{n_1 \times n_2 \times \cdots \times n_k}$, *there exists a CP model with rank* $r_{CP} = \prod_{i=1}^{k} n_i$, *that exactly decomposes* $\mathcal{W}$.

Proof. Since $\mathcal{W} \in \{0,1\}^{n_1 \times n_2 \times \cdots \times n_k}$, we can use $r_{CP} = \prod_{i=1}^{k} n_i$ zero/one-hot tensors $\{\mathcal{W}^{(r)} \mid \mathcal{W}^{(r)} \in \{0,1\}^{n_1 \times n_2 \times \cdots \times n_k}\}_{r=1}^{r_{CP}}$ to represent $\mathcal{W}$, s.t., $w^{(1)}_{1\cdots 1} = w_{1\cdots 1}$, while other elements in $\mathcal{W}^{(1)}$ are zeros, $w^{(r)}_{j_1^{(r)} j_2^{(r)} \cdots j_k^{(r)}} = w_{j_1^{(r)} j_2^{(r)} \cdots j_k^{(r)}}$, while other elements in $\mathcal{W}^{(r)}$ are zeros, etc. Finally, $w^{(r_{CP})}_{n_1 n_2 \cdots n_k} = w_{n_1 n_2 \cdots n_k}$, while other elements in $\mathcal{W}^{(r_{CP})}$ are zeros. Moreover, $\mathcal{W}^{(r)}$ can be decomposed by rank-one CP decomposition as,

$$\mathcal{W}^{(r)} = \boldsymbol{u}_r^{(1)} \circ \boldsymbol{u}_r^{(2)} \circ \cdots \circ \boldsymbol{u}_r^{(k)}, \tag{19}$$

$$\text{s.t. } w^{(r)}_{j_1^r j_2^r \cdots j_k^r} = u_r^{(1)}(j_1^r) \cdot u_r^{(2)}(j_2^r) \cdot \cdots \cdot u_r^{(k)}(j_k^r), \tag{20}$$

where $\boldsymbol{u}_r^{(i)}$ is the n_i-dimensional zero/one-hot vector, and $u_r^{(i)}(j)$ is the jth element of the vector. Therefore, by assigning $u_r^{(1)}(j_1^r) = u_r^{(2)}(j_2^r) = \cdots = u_r^{(k)}(j_k^r) = w^{(r)}_{j_1^r j_2^r \cdots j_k^r}$, and other elements in $\boldsymbol{u}_r^{(i)}$ being zeros, the binary tensor $\mathcal{W}$ can be exactly decomposed by CP decomposition *via* the set of vectors $\{\boldsymbol{u}_r^{(i)} \mid i = 1, \cdots, k, r = 1, \cdots, r_{CP}, r_{CP} = \prod_{i=1}^{k} n_i\}$ as,

$$\mathcal{W} = \sum_{r=1}^{r_{CP}} \mathcal{W}^{(r)} = \sum_{r=1}^{r_{CP}} \boldsymbol{u}_r^{(1)} \circ \boldsymbol{u}_r^{(2)} \circ \cdots \circ \boldsymbol{u}_r^{(k)}. \tag{21}$$

□

Lemma B.3 [55]. *CP decomposition can be viewed as a special case of TR decomposition. Given a kth-order binary tensor* $\mathcal{W} \in \{0,1\}^{n_1 \times n_2 \times \cdots \times n_k}$ *with its CP decomposition as Equation (21), it can also be written in TR decomposition form as,*

$$\mathcal{W} = \sum_{r=1}^{r_{CP}} \boldsymbol{u}_r^{(1)} \circ \boldsymbol{u}_r^{(2)} \circ \cdots \circ \boldsymbol{u}_r^{(k)} = \boldsymbol{TR}(\mathcal{Z}_1, \mathcal{Z}_2, \cdots, \mathcal{Z}_k),$$
$$\text{s.t. } \boldsymbol{Z}_i(j_i) = diag(\boldsymbol{u}^{(i)}(j_i)), \quad \forall i = 1, 2, \cdots, m, \tag{22}$$

where $\mathcal{Z}_i \in \{0,1\}^{r_{CP} \times n_i \times r_{CP}}$, $\boldsymbol{u}^{(i)}(j_i) = [u_1^{(i)}(j_i), \cdots, u_{r_{CP}}^{(i)}(j_i)]$.

According to Lemma B.1, n-TuckER is fully expressive by setting the embeddings as well as the core tensor, in which the core tensor is set to an $(n+1)$ -th order binary tensor $\mathcal{W} \in \{0,1\}^{n_r \times n_e \times n_e \times \cdots \times n_e}$.

In GETD-S, $\mathcal{W}$ is reshaped into a kth-order reshaped tensor $\hat{\mathcal{W}} \in \{0,1\}^{n_1 \times \cdots \times n_k}$, which is further decomposed by TR decomposition. Keeping the embedding settings as the ones in Lemma B.1, we only need to prove that TR decomposition is able to recover any given tensor $\hat{\mathcal{W}}$. On the other hand, with Lemma B.2, $\hat{\mathcal{W}}$ is able to be completely recovered *via* CP decomposition. Moreover, the CP decomposition can be written as a special case of TR decomposition by Lemma B.3, which derives TR latent tensors $\{\mathcal{Z}_i \mid \mathcal{Z}_i \in \{0,1\}^{r_{CP} \times n_i \times r_{CP}}\}_{i=1}^{k}$. Overall, following the settings of embeddings in Lemma B.1 and TR latent tensors in Lemma B.3, GETD-S is proved to be fully expressive with entity embeddings of dimensionality $n_e = |\mathcal{E}|$ and relation embeddings of dimensionality $d_r = |\mathcal{R}|$.

□

We then prove Theorem 5.2 as follows.

Proof. For GETD-M, the entity and relation embeddings follow the assignment in Lemma B.1. We treat facts with the same arity relations as a sub KB, then the GETD-M model can been seen as an ensemble of GETD-S models for different arity KBs. Especially, the TR tensor group is constructed with separate subgroups of tensors for different arities, whose values follow the assignment in Lemma B.3. Thus, based on Theorem 5.1, GETD-M is fully expressive. □

Received 9 August 2022; revised 9 November 2024; accepted 3 December 2024