Generalizing Tensor Decomposition for N-ary Relational Knowledge Bases

Yu Liu  
BNRist, EE, Tsinghua University  
Beijing, China  
liuyu2419@126.com

Quanming Yao  
4Paradigm Inc  
Hong Kong  
yaoquanming@4paradigm.com

Yong Li  
BNRist, EE, Tsinghua University  
Beijing, China  
liyong07@tsinghua.edu.cn

ABSTRACT

With the rapid development of knowledge bases (KBs), link prediction task, which completes KBs with missing facts, has been broadly studied in especially binary relational KBs (a.k.a knowledge graph) with powerful tensor decomposition related methods. However, the ubiquitous n-ary relational KBs with higher-arity relational facts are paid less attention, in which existing translation based and neural network based approaches have weak expressiveness and high complexity in modeling various relations. Tensor decomposition has not been considered for n-ary relational KBs, while directly extending tensor decomposition related methods of binary relational KBs to the n-ary case does not yield satisfactory results due to exponential model complexity and their strong assumptions on binary relations. To generalize tensor decomposition for n-ary relational KBs, in this work, we propose GETD, a generalized model based on Tucker decomposition and Tensor Ring decomposition. The existing negative sampling technique is also generalized to the n-ary case for GETD. In addition, we theoretically prove that GETD is fully expressive to completely represent any KBs. Extensive evaluations on two representative n-ary relational KB datasets demonstrate the superior performance of GETD, significantly improving the state-of-the-art methods by over 15%. Moreover, GETD further obtains the state-of-the-art results on the benchmark binary relational KB datasets.

1 INTRODUCTION

In the past decade, the emerging of numerous web-scale knowledge bases (KBs) such as Freebase [3], Wikidata [34], YAGO [31] and Google’s Knowledge Graph (KG) [28], has inspired various applications, e.g., question answering [23], recommender systems [42] and natural language processing (NLP) [22]. Most of these KBs are constructed based on binary relations with triplet facts represented as (head entity, relation, tail entity). However, due to enormous missing facts, KBs face a fundamental issue of incompleteness, which drives KB completion researches especially link prediction: predicting whether two entities are related based on known facts in KBs [2]. Extensive studies have been proposed to address this problem, including translational distance models [6, 15, 21, 38], neural network models [8, 27, 29], and tensor decomposition models [2, 17, 25, 32, 40]. Among them, the mathematically principled tensor decomposition models achieve the best performance, with strong capability to capture latent interactions between entities and relations in KBs.

Despite the great attention in the binary relational KBs, higher-arity relational KBs 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 with a Person, a Product, and a Seller. Sports_award is a 4-ary relation, involved with 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 [39], more than 1/3 of the entities in Freebase participate in the n-ary relation. Besides, since higher-arity relations with more knowledge are closer to natural language, link prediction in n-ary relational KBs provides an excellent potential for question answering related NLP applications [9].

To handle link prediction in n-ary relational KBs directly, two categories of models based on translational distance and neural network are proposed recently. In terms of translational distance models, m-TransH [39] directly extends TransH [38] for binary relations to the n-ary case. RAE [43] further integrates m-TransH with multi-layer perceptron (MLP) by considering the relatedness of entities. However, since the distance-based scoring function of TransH enforces constraints on relations, it fails to represent some binary relations in KBs [17]. Accordingly, its extensions of m-TransH and RAE are not able to represent some n-ary relations, which impairs the performance. In terms of neural network models, NaLP [11] leverages neural networks for n-ary relational fact modeling and entity relatedness evaluation, and obtains the state-of-the-art results. Nevertheless, NaLP owes good performance to an enormous amount of parameters, which contradicts the linear time and space requirement for relational models in KBs [5]. Therefore, existing

CCS CONCEPTS
• Information systems → Relational database model; • Computing methodologies → Reasoning about belief and knowledge; Statistical relational learning.

KEYWORDS
Link Prediction, N-ary Relation, Knowledge Base, Tucker Decomposition, Tensor Ring Decomposition

ACM Reference Format:
Yu Liu, Quanming Yao, and Yong Li. 2020. Generalizing Tensor Decomposition for N-ary Relational Knowledge Bases. In Proceedings of The Web Conference 2020 (WWW ’20), April 20–24, 2020, Taipei, Taiwan. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380188

This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.
WWW ’20, April 20–24, 2020, Taipei, Taiwan © 2020 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.
ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380188
methods do not provide an efficient solution for link prediction in n-ary relational KBs, and it is still an open problem to be addressed.

Although tensor decomposition models have been proved to be very powerful in binary relational KBs by both the state-of-art results [2] and theoretical guarantees on full expressiveness [17, 37], there is no work to our knowledge adopting this type of model for link prediction in n-ary relational KBs. 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 [17, 32] leverage some tricky operations in scoring functions for great performance, 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.

To solve link prediction problem in n-ary relational KBs as well as tackle above challenges, we generalize tensor decomposition for n-ary relational KBs. Specifically, we first extend TuckER [2], the state-of-the-art model for binary relations, to n-TuckER for n-ary relations, with Tucker decomposition utilized [33]. Note that the higher-order core tensor in n-TuckER grows exponentially with the arity, and excessively complex models usually overfit, which implies poor performance. Thus, motivated by the benefits of tensor ring (TR) decomposition [46] for neural network compression in computer vision [26, 36], we integrate TR with n-TuckER together. By representing the higher-order core tensor into a sequence of 3rd-order tensors, TR significantly reduces the model complexity with performance enhanced. The overall model is termed as GETD, Generalized Tensor Decomposition for n-ary relational KBs. Since most KBs only provide positive observations, we also generalize the existing negative sampling technique for efficiently training on GETD. Considering the importance for a link prediction model to have enough expressive power to represent various true and false facts in KBs, we further prove that GETD is fully expressive.

The main contributions of this paper are summarized as follows:

• We investigate tensor decomposition for link prediction in n-ary relational KBs, 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 model for n-ary relational KBs. GETD integrates TR decomposition with Tucker decomposition, which scales well with both the size and the arity of the KB. We also generalize the negative sampling technique from the binary to the n-ary case. To the best of our knowledge, GETD is the first model that leverages tensor decomposition techniques for n-ary relational link prediction.

• We prove that GETD is fully expressive for n-ary relational KBs, which is able to cover all types of relations, and can completely separate true facts from false ones.

• We conduct extensive experiments on two representative n-ary relational KB datasets. The results demonstrate that GETD outperforms state-of-the-art n-ary relational link prediction models by 15%. Furthermore, GETD achieves close and even better performance on two standard binary relational KB datasets compared with existing state-of-the-art models.

• We organize the rest of this paper as follows. Section 2 gives a systematic review on the related works of link prediction in KBs. Section 3 introduces the background of tensor decomposition and notations. After that, the framework of GETD and theoretical analyses are presented in Section 4. Section 5 evaluates the performance on representative KB datasets and provides extensive analyses. In light of our results, this paper is concluded in Section 6.

2 RELATED WORK
We classify the related works into two categories of binary relational KBs and n-ary relational KBs.

2.1 Link Prediction on Binary Relational KBs
Basically, link prediction models 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, 15, 21, 38], neural network models [8, 27, 29], and tensor decomposition models [2, 17, 25, 32, 40].

Translational distance models measure the entity distance after a translational operation carried out by the relation [35], and various translational operations are exploited for distance-based scoring functions [6, 15, 21, 38]. However, most translational distance models are found to have restrictions on relations [17, 35], thus can only represent part of relations.

Neural network models [8, 27, 29] subtly design the scoring function with various neural network structures, which always require a great many parameters to completely represent all relations [17, 35], 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 link prediction task is framed as a 3rd-order binary tensor completion problem, 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. For example, DistMult [40] uses Canonical Polyadic (CP) decomposition [13] with the equivalence of head and tail embeddings for the same entity, however, fails to capture the asymmetric relation. Furthermore, Simple [17] takes advantage of the inverse of relations to address the asymmetric relation, while ComplEx [32] leverages complex-valued embeddings for solution. Recently, Tucker decomposition [33] is adopted in TuckER [2] for link prediction, 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. According to the discussion, generalizing tensor decomposition is promising for n-ary relational link prediction.

2.2 Link Prediction on N-ary Relational KBs
Existing works on n-ary relational link prediction can be categorized into two classes based on the scoring function: translational distance models [39, 42] and neural network models [11].
The translational distance models of m-TransH [39] and RAE [43] are the first series of works in this field. Based on the distance translation idea, m-TransH is proposed by extending TransH [38] 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 link prediction. 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.

The neural network model, NaLP [11], is recently proposed for the state-of-the-art performance in n-ary relational link prediction. In NaLP, entity embeddings of an n-ary relational fact are first passed to the convolution for feature extraction, then the overall relatedness is modeled by FCN, whose output represents the evaluation score. However, a great many parameters involved in NaLP make the training intractable. Moreover, the latent connections between similar relations are not considered, which further leads to inferior empirical performance.

As previously discussed, tensor decomposition is a potential solution for n-ary relational link prediction, 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 tricky operations. For instance, to model all types of relations, the relation inverse in SimplE [17] and complex-valued embeddings in ComplEx [32] 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 [17, 32]. Besides, other models like DistMult [40] force the relation to be symmetric, thus, not able to completely represent n-ary relational KBs. A recent work [10] explores DistMult with convolution to the n-ary case, but the interaction of entities and relations is not fully captured. Through our investigation, GETD is the first generalized tensor decomposition model for n-ary relational KBs with both performance and model complexity satisfied.

## 3 BACKGROUND AND NOTATION

### 3.1 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 $a_i$ denote the i-th column of a matrix $A$, $X_{i_1i_2...i_n}$ denote the $(i_1, i_2, ..., i_n)$-th element of a higher-order tensor $X \in \mathbb{R}^{I_1 \times \ldots \times I_n}$, where $I_1$ is the dimensionality of the i-th mode. Especially, given a 3rd-order tensor $Z \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the $i_2$-th lateral slice matrix of $Z$ is denoted by $Z(i_2)$ in the size of $I_1 \times I_3$, a.k.a., $Z_{i_2}$, where the colon indicates all elements of a mode.

| Symbol | Definition |
|--------|------------|
| $X$    | nth-order tensor $\in \mathbb{R}^{I_1 \times \ldots \times I_n}$ |
| $x_{i_1i_2...i_n}$ | $(i_1, i_2, ..., i_n)$-th element of $X$ |
| $\mathcal{G}$ | nth-order core tensor $\in \mathbb{R}^{I_1 \times \ldots \times I_n}$ |
| $A^{(k)}$ | k-mode factor matrix $\in \mathbb{R}^{I_k \times I_n}$ |
| $a_i^{(k)}$ | j-th column vector of $A^{(k)}$ |
| $Z_{i_2}$ | $i_2$-th lateral slice matrix of $Z_{i_2} \in \mathbb{R}^{I_1 \times I_3}$ |
| $r = [r_1, r_2, \ldots, r_n]$ | TR-ranks |
| $n_e$, $n_r$ | the number of entities/relations in the KB |
| $d_e$, $d_r$ | entity/relation embedding dimensionality |
| $n_l$ | 2nd-mode dimensionality of $Z_i$ |
| $\circ$ | vector outer product |
| $\otimes$ | tensor n-mode product |
| $\langle \rangle$ | multi-linear dot product |
| $\text{trace} \{\cdot\}$ | matrix trace operator |

As for the operation on tensors, $\circ$ represents the vector outer product, and $x_i$ represents the tensor i-mode product. $\langle \rangle$ represents the multi-linear dot product, written as $\langle a^{(1)}, a^{(2)}, \ldots, a^{(n)} \rangle = \sum_i a_i^{(1)} a_i^{(2)} \cdots a_i^{(n)}$. $\text{trace} \{A\} = \sum_i a_{ii}$. More details about these operations and tensor properties can be referred to [19]. The related notations frequently used in this paper are listed in Table 1.

### 3.2 Tucker Decomposition

Tucker decomposition was initially proposed for three-order tensor decomposition [33]. It can be generalized to higher order, which decomposes a higher-order tensor into a set of factor matrices and a relatively small core tensor. Given an nth-order tensor $X \in \mathbb{R}^{I_1 \times \ldots \times I_n}$, Tucker decomposition can be denoted as,

$$
X \approx \mathcal{G} \times_1 A^{(1)} \times_2 A^{(2)} \cdots \times_n A^{(n)}
$$

$$
= \sum_{j_1=1}^{J_1} \sum_{j_2=1}^{J_2} \cdots \sum_{j_n=1}^{J_n} g_{j_1j_2\ldots j_n} a_{j_1}^{(1)} \circ a_{j_2}^{(2)} \circ \cdots \circ a_{j_n}^{(n)},
\tag{1}
$$

where $\mathcal{G} \in \mathbb{R}^{J_1 \times \ldots \times J_n}$ is the core tensor, $J_k$ is the rank of k-th mode, and $(A^{(k)}) \in \mathbb{R}^{J_k \times I_k}$ is the set of factor matrices. Usually, $J_1, \ldots, J_n$ are smaller than $I_1, \ldots, I_n$. Thus the number of parameters is reduced compared with the approximated tensor $X$.

### 3.3 Tensor Ring (TR) Decomposition

Although Tucker decomposition approximates a higher-order tensor with fewer parameters, the number of parameters scales exponentially to the tensor order. Tensor ring (TR) decomposition [46], on the other hand, represents a higher-order tensor by a sequence of 3rd-order latent tensors multiplied circularly. Given an nth-order tensor $X \in \mathbb{R}^{I_1 \times \ldots \times I_n}$, TR decomposition can be expressed in an element-wise form as,

$$
x_{i_1i_2...i_n} \approx \text{trace} \{Z_1(i_1)Z_2(i_2)\cdots Z_n(i_n)\} = \text{trace} \left[ \prod_{k=1}^{n} Z_k(i_k) \right],
\tag{2}
$$
where \( \{Z_k \mid Z_k \in \mathbb{R}^{r_k \times l_k \times r_{k+1}}, r_1 = r_{n+1}\}_{k=1}^n \) is the set of TR latent tensors, and \( Z_k(i_k) \) is in the size of \( \mathbb{R}^{r_k \times l_k \times r_{k+1}} \) accordingly. For convenience, we also denote the above TR decomposition as \( \text{TR}(Z_1, \ldots, Z_k) \). Especially, the size of latent factors, concatenated and denoted by \( r = [r_1, r_2, \ldots, r_n] \) is called TR-ranks.

4 GETD: DESIGN AND MODEL

Borrowing the concept of n-ary relation [7, 9], the n-ary relational fact can be defined as follows,

**Definition 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 \( (i_1, i_2, \ldots, i_n) \) \( \subseteq \mathcal{R} \times \mathcal{E} \times \mathcal{E} \times \cdots \times \mathcal{E} \), where \( \mathcal{R} \) and \( \mathcal{E} \) are called relation domain and entity domain.

Especially, \( i_k \) is the k-th entity to the relation \( i_r \), belonging to the k-th 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.

Then, the link prediction problem can be defined as follows,

**Problem 1 (Link prediction).** Given an incomplete n-ary relational KB \( S = \{(i_1, i_2, \ldots, i_n)\} \), the link prediction problem aims to infer missing facts based on \( S \).

In practice, given the relation and any \( n - 1 \) entities in an n-ary relational fact, the problem is simplified as predicting the missing entity, e.g. predicting the 1-st entity of the incomplete n-ary relational fact \( (i_r, ?, i_2, \ldots, i_n) \).

4.1 Rethinking Tucker for KBs

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. \( x_{i_r, i_1, i_2, \ldots, i_n} \) equal to one means the specific n-ary relational fact is true, and zero for false/missing. The approximated low-rank scoring tensor is denoted by \( \mathcal{X} \in \mathbb{R}^{n_r \times n_e \times n_e \times \cdots \times n_e} \). Accordingly, the link prediction on \( (i_r, ?, i_2, i_3, \ldots, i_n) \) can be answered by the entity with maximum value or score in corresponding mode vector of the scoring tensor.

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

The scoring function is defined as,

\[
\phi(i_r, i_1, i_2, \ldots, i_n) = \hat{X}_{i_r, i_1, i_2, \ldots, i_n} = W_{i_r} \times_1 r_{i_1} \times_2 e_{i_1} \times_3 e_{i_2} \times \cdots \times_n e_{i_n},
\]

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 [17, 32].

4.2 The GETD Model

In this part, the model construction of GETD with the scoring function is first introduced. Since the existing negative sampling technique only limits on binary relational KBs, then we deal with negative samples for the n-ary case.

4.2.1 The Scoring Function.** 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 expressiveness of the model.

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.

First, in the outer layer of GETD, the original KB tensor is decomposed via Tucker decomposition following (3), which reserves Tucker interaction way as well as strong expressiveness. Subsequently, in the inner layer, the intermediate core tensor \( W \)
is flexibly reshaped to a $k$th-order tensor $\hat{\mathbf{W}} \in \mathbb{R}^{n_1 \times \cdots \times n_k}$ with $\prod_{i=1}^{k} n_i = d^e_e (k \geq n + 1)$ satisfied. Then TR decomposes the reshaped tensor $\hat{\mathbf{W}}$ into $k$ latent 3rd-order tensors $\{Z_i \}_{i=1}^{k}$, reducing the number of parameters. The main framework of GETD is shown in Figure 1 (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,

$$w_{ij_2\ldots j_k} = \text{trace}(Z_i(j_1)Z_2(j_2)\cdots Z_k(j_k)). \quad (4)$$

Overall, the scoring function of GETD can be expressed as,

$$\phi(i, i_1, i_2, \ldots, i_n) = \hat{\mathbf{W}} \times_1 \mathbf{r}_i \times_2 \mathbf{e}_{i_1} \times_3 \mathbf{e}_{i_2} \cdots \times_{n+1} \mathbf{e}_{i_n} = \text{TR}(Z_1, \ldots, Z_k) \times_1 \mathbf{r}_i \times_2 \mathbf{e}_{i_1} \times_3 \mathbf{e}_{i_2} \cdots \times_{n+1} \mathbf{e}_{i_n}. \quad (5)$$

The model complexity of GETD is

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

which is much lower than n-TuckER, and discussed in detail later. Accordingly, with Tucker interaction way as well as low model complexity, GETD not only guarantees strong expressive capability, but also avoids the overfitting problem with many parameters, which improves the testing performance.

4.2.2 Dealing with Negative Samples. In KBs, usually only have positive observations, i.e., which relation exists among different sets of entities. Thus, even with the designed scoring function in (5), we cannot train an embedding model due to lack of negative observations. In embedding of binary relational KBs, given a positive triplet $(i_r, i_1, i_2)$, good candidates of negative samples [24] are

$$\mathcal{N}(i_r, i_1, i_2) = \mathcal{N}^{(1)}_{i_r, i_1, i_2} \cup \mathcal{N}^{(2)}_{i_r, i_1, i_2} \equiv \{(i_r, i_1, i_2) \notin \mathcal{S} \mid i_1 \in \mathcal{E} \} \cup \{(i_r, i_1, i_2) \notin \mathcal{S} \mid i_2 \in \mathcal{E} \}. \quad (6)$$

Then, negative samples can be sampled from (6) by either fixed or dynamic distribution [45]. More recently, multi-class log-loss [16, 20] has developed as a replacement for the above sampling scheme, which can offer better learning performance. Specifically, it considers all candidates in (6) simultaneously, i.e.,

$$\mathcal{L}(i_r, i_1, i_2) = \mathcal{L}^{(1)}_{i_r, i_1, i_2} + \mathcal{L}^{(2)}_{i_r, i_1, i_2}, \quad (7)$$

where

$$\mathcal{L}^{(j)}_{i_r, i_1, \ldots, i_n} = -\phi(i_r, i_1, \ldots, i_n) + \log \left( \sum_{x \in \mathcal{N}^{(j)}_{i_r, i_1, \ldots, i_n}} \phi(x) \right).$$

Here, we extend the above multiclass log-loss to n-ary relational KBs. For one positive n-ary relational example $(i_r, i_1, i_2, \ldots, i_n)$, n groups of negative sample candidates are generated from corresponding n entity domains, defined as

$$\mathcal{N}(i_r, i_1, i_2, \ldots, i_n) = \bigcup_{m=1}^{n} \{(i_r, i_1, i_2, \ldots, i_m) \notin \mathcal{S} \mid i_m \in \mathcal{E} \}. \quad (8)$$

Accordingly, with negative samples given in (8), the loss function of GETD is defined as,

$$\mathcal{L}(i_r, i_1, i_2, \ldots, i_n) = \sum_{j=1}^{n} \mathcal{L}^{(j)}_{i_r, i_1, i_2, \ldots, i_n}. \quad (9)$$

### 4.3 Training

GETD is 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. 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 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 sample set $\mathcal{N}(i_r, i_1, i_2, \ldots, i_n)$ following (8), as shown in line 6. Then, the score of the observation as well as the negative samples are computed using (5) in line 7, which are further utilized to compute the multiclass log-loss with (9) in lines 8-9. Finally, the algorithm updates the model parameters according to the loss gradients.

### 4.4 Complexity Analysis

According to the model description, the entity and relation embedding dimensions and TR-ranks are $O(n_e d_e + n_r d_r)$ 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 TR latent tensors cost $O(k n_{\max}^3)$ parameters in sum. Thus, the model complexity of GETD is obtained as $O(n_e d_e + n_r d_r + k n_{\max}^3)$, while GETD also retains the efficiency benefits of tensor model product in linear time complexity.

Moreover, due to the constraint $\prod_{i=1}^{k} n_i = d^e_e (k \geq n + 1)$ in GETD, if TR latent tensors are in the same shape, we can obtain the equation of $n_{\max} = (d^e_e d_r)^{1/k}$. When applied in large-scale KBs with over thousands of entities [4, 31], GETD 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.

We compare GETD with state-of-the-art n-ary relational link prediction models, in terms of scoring function, expressiveness and the model complexity in Table 2. Among the models, n-TuckER and n-CP are the extensions of Tucker [2] and CP [13], respectively.
A link prediction model is fully expressive if for any ground truth over entities and relations, there exist embeddings that accurately separate the true n-ary relational facts from the false ones, i.e., the link prediction model can recover any given KB tensors by the assignment of entity and relation embeddings [2, 17, 32, 35]. As stated before, GETD has linear model complexity $O(n ed^2 + nd^3)$ to KB sizes in practical. Therefore, GETD achieves the lowest model complexity in tensor decomposition models with the best performance.

As introduced in Section 2, RAE [43] is a translational distance model, and NaLP [11] is a neural network model. Although these two models achieve similar model complexity to GETD, they are short of expressive power, thus perform badly, which is proved by link prediction performance in Section 5 later. Thus, GETD performs best on both model complexity and expressiveness.

### 4.5 Full Expressiveness

A link prediction 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 link prediction model can recover any given KB tensors by the assignment of entity and relation embeddings [2, 17, 32, 35].

The full expressiveness guarantees the completeness of link prediction and KB completion. Especially, if a link prediction 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 [12]. For instance, DistMult is not fully expressive, and forces relations to be symmetric, i.e., it can represent KBs with only symmetric relations [17], 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.

**Theorem 1.** For any ground truth over entities $E$ and relations $R$, there exists a GETD model that represents that ground truth (See proof in Appendix A).

### 5 EXPERIMENTS AND RESULTS

#### 5.1 Experimental Setup

##### 5.1.1 Datasets

We evaluate our model with two real datasets across 3-ary and 4-ary relational KBs, one synthetic dataset as well as two benchmark datasets on binary relations, which are introduced as follows.

- **WikiPeople** [11]: This is a public n-ary relational dataset extracted from Wikidata concerning entities of type human. WikiPeople is quite practical, where data incompleteness, insert and update are universal. 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** [43]: This is a public n-ary relational dataset developed from Freebase, whose facts are in good quality. 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 as 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 relational facts, named as Synthetic10-3 and Synthetic10-4, whose KB tensors are randomly generated by CP decomposition with tensor rank equal to one [19]. There are only 10 entities and 2 relations in Synthetic10.

- **WN18** [4]: This binary relational dataset is a subset of WordNet, a database with lexical relations between words.

- **FB15k** [5]: This binary relational dataset is a subset of Freebase, a database of real world facts including films, sports, etc.

Besides, facts in first three datasets are randomly split into train/valid/test sets by a proportion of 8:1:1. The train/valid/test sets of WN18 and FB15k provided in [6] are used for evaluation. The datasets statistics are summarized in Table 3.
5.1.2 Metrics. We evaluate the link prediction performance with two standard metrics: mean reciprocal rank (MRR) and Hits@k, \( k \in \{1, 3, 10\} \) \[2, 6, 11, 17, 32, 40\]. 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 link prediction model. The entities in all entity domains are tested. The ranking of the testing fact is obtained by 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.

5.1.3 Baselines. We compare GETD with the following n-ary relational link prediction baselines:

- **RAE** \[43\] is a translational distance model, extending TransH \[38\] to m-TransH with relatedness combined\(^1\).
- **NaLP** \[11\] is a neural network model, which achieves the state-of-the-art n-ary relational link prediction performance\(^2\).
- **n-CP** is an extension of CP decomposition \[13\], firstly applied in n-ary relational link prediction in this paper.
- **n-TuckER** is an extension of TuckER \[2\] with Tucker decomposition utilized, also firstly applied in n-ary relational link prediction in this paper.

Besides, we compare GETD with state-of-the-art models in binary relational KBs, including TransE \[6\], DistMult \[40\], ConvE \[8\], ComplEx \[32\], SimpLE \[17\], and TuckER \[2\].

5.1.4 Implementation. The implementation of GETD is available at Github\(^3\). For experimental fairness, we fix entity and relation embedding sizes of GETD, n-CP and n-TuckER. For 3-ary relational datasets 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 \( d_e = d_r = 25 \), \( k = 5 \), \( r_i = n_i = 25 \). Besides, batch normalization \[14\] and dropout \[30\] 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 \([0.001, 0.1]\) 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. Above three models are trained with Adam \[18\] using early stopping based on validation set MRR with no improvement for 10 epochs. As for RAE and NaLP, we use the optimal settings reported in \[43\] and \[11\], respectively.

5.2 N-ary Relational Link Prediction

Table 4 and 5 present the link prediction results on two datasets across 3-ary and 4-ary relational KBs. From the results, we can observe that our proposed GETD achieves the best performance on all metrics across all datasets. Especially, tensor decomposition models of GETD, n-CP and n-TuckER always outperform the translational distance model RAE and the neural network model NaLP. For example, on JF17K, compared with existing state-of-the-art model NaLP, GETD improves MRR by 0.22 and Hits@1 by 55% for 3-ary relational facts, while the improvement for 4-ary relational facts is 0.09 and 12%. For WikiPeople, GETD improves MRR by 0.07 and Hits@1 by 55% for 3-ary relational facts, while the improvement for 4-ary relational facts is 0.09 and 12%. For WikiPeople-3 and WikiPeople-4, GETD improves MRR by 0.07 and Hits@1 by 55% for 3-ary relational facts, while the improvement for 4-ary relational facts is 0.09 and 12%. For WikiPeople-3 and WikiPeople-4, these considerable improvements further confirm the strong expressive power of the proposed tensor decomposition models. Moreover, the great performance on WikiPeople indicates that GETD is robust and able to handle practical KB issues like data incompleteness, insert and update. Note that the relatively less improvement on 4-ary relational facts may partly owe to the sparsity of higher-arity facts in datasets.

As for the three tensor decomposition models, n-CP is relatively weak due to the difference of embeddings in different entity domains \[32\]. while GETD and n-TuckER capture the interaction between entities and relations with TR latent tensors or core tensors. On

---

\(^1\)https://github.com/lijp12/SIR

\(^2\)https://github.com/gp2014/NaLP

\(^3\)https://github.com/luyuaas/GETD

| Model | WikiPeople-3 | WikiPeople-4 |
|-------|--------------|--------------|
|       | MRR | Hits@10 | Hits@3 | Hits@1 | MRR | Hits@10 | Hits@3 | Hits@1 |
| RAE   | 0.239 | 0.379 | 0.252 | 0.168 | 0.150 | 0.273 | 0.149 | 0.080 |
| NaLP  | 0.301 | 0.445 | 0.327 | 0.226 | 0.342 | 0.540 | 0.400 | 0.237 |
| n-CP  | 0.330 | 0.496 | 0.356 | 0.250 | 0.265 | 0.445 | 0.315 | 0.169 |
| n-TuckER | 0.365 | 0.548 | 0.400 | 0.274 | 0.362 | 0.570 | 0.432 | 0.246 |
| GETD  | 0.373 | 0.558 | 0.401 | 0.284 | 0.386 | 0.596 | 0.462 | 0.265 |

| Model | JF17K-3 | JF17K-4 |
|-------|---------|---------|
|       | MRR | Hits@10 | Hits@3 | Hits@1 | MRR | Hits@10 | Hits@3 | Hits@1 |
| RAE   | 0.505 | 0.644 | 0.532 | 0.430 | 0.707 | 0.835 | 0.751 | 0.636 |
| NaLP  | 0.515 | 0.679 | 0.552 | 0.431 | 0.719 | 0.805 | 0.742 | 0.673 |
| n-CP  | 0.700 | 0.827 | 0.736 | 0.635 | 0.787 | 0.890 | 0.821 | 0.733 |
| n-TuckER | 0.727 | 0.852 | 0.761 | 0.664 | 0.804 | 0.902 | 0.841 | 0.748 |
| GETD  | 0.732 | 0.856 | 0.764 | 0.669 | 0.810 | 0.913 | 0.844 | 0.755 |
the other hand, GETD also outperforms n-TuckER owing to the simplicity with much fewer parameters, while the parameter-cost 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 on the largest dataset WikiPeople-3 with a Titan-XP GPU. An epoch training takes about 28s and total training takes 1h, while inference takes only 5s. Overall, the results show the efficiency and robustness of GETD for link prediction in n-ary relational KBs.

5.3 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 2(a) and 2(b), respectively.

![Figure 2](image2.png)

Figure 2: Overfitting in n-TuckER observered from MRR (left) and loss (right). Evaluated on WikiPeople-4.

From the results, we can clearly observe the overfitting phenomenon in training process of n-TuckER. In Figure 2(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 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 and n-CP test loss curves. It mainly caused by the model complexity that, the numbers of parameters in GETD 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 3. 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 core tensor in n-TuckER costs $50 \times 50 \times 50 = 6.25$ million parameters, while the TR latent tensors in GETD cost only $4 \times 1 \times 50 = 200$ 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 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 in Table 4 and 5.

5.4 Influence of Parameters
Since the embedding sizes are important factors to link prediction models with expressiveness [2, 8, 32], while TR-ranks, as well as the reshaped tensor order are unique hyper-parameters of GETD, determining the model complexity and performance, now we investigate the impacts of these parameters.

5.4.1 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 4.

![Figure 4](image4.png)

Figure 4: MRR (left) and number of parameters (right) under different embedding sizes. Evaluated on WikiPeople-4.

According to Figure 4(a), GETD always outperforms n-TuckER and n-CP. The MRR of GETD 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 increases MRR by 49% for n-CP, and 8% for n-TuckER. Moreover, the MRR of GETD reaches 0.372 at embedding size 15, which is better than the performance of n-TuckER at embedding size over 30. On the other hand, GETD uses the least parameters in three models, which is shown in Figure 4(b). For embedding size 30, n-TuckER costs 24 million parameters with core tensor using $30^3 = 24.3$ million, n-CP costs 1.14 million parameters, while GETD
Table 6: Link prediction results on WN18 and FB15k. Results of TransE and DistMult are copied from [32]. Other results are copied from the corresponding original papers [2, 8, 17, 32].

| 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 [40]  | 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 [32]   | 0.941 | 0.947   | 0.945  | 0.936  | 0.692 | 0.840   | 0.759  | 0.599  |
| SimpLE [17]    | 0.492 | 0.947   | 0.944  | 0.939  | 0.727 | 0.838   | 0.773  | 0.660  |
| TuckER [2]     | 0.953 | 0.958   | 0.955  | 0.949  | 0.795 | 0.892   | 0.833  | 0.741  |
| GETD           | 0.948 | 0.954   | 0.950  | 0.944  | 0.824 | 0.888   | 0.847  | 0.787  |

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 accord with complexity analysis in Section 4.4, and further indicate that GETD with relatively small embedding sizes is able to obtain good performance, which can be applied for large-scale KBs.

5.4.2 Influence of TR-ranks $r$. Since the TR-ranks can largely determine the number of TR latent tensor parameters, and make GETD model more flexible, we reveal the relationship between link prediction performance and TR-ranks on WikiPeople-4 and JF17K-3, as shown in Figure 5. 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 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 can be further reduced to control model complexity for large-scale KBs.

Figure 5: MRR of GETD on WikiPeople-4 (left) and JF17K-3 (right) under different TR-ranks.

5.4.3 Influence of Reshaped Tensor Order $k$. As a key step of connecting Tucker and TR in GETD, the effect of reshaped tensor order is investigated on WikiPeople-4 and JF17K-3, exhibited in Figure 6.

For WikiPeople-4, the embedding size is set to 25 and thus the original core tensor is $W \in \mathbb{R}^{25 \times 25 \times 25 \times 25}$, leading to the 5th-order reshaped tensor $\tilde{W} \in \mathbb{R}^{25 \times 25 \times 25 \times 25 \times 25}$, the 6th-order reshaped tensor $\tilde{W} \in \mathbb{R}^{5 \times 25 \times 25 \times 25 \times 25}$, etc. It can be observed that, GETD with different orders 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 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 [19]. For example, the 6th-order reshaped tensor becomes $\tilde{W} \in \mathbb{R}^{64 \times 64 \times 64 \times 64 \times 64 \times 64}$, and the size of each mode for 8th-order tensor becomes 4. Similarly, GETD with the least order of reshaped tensor achieves the highest MRR. Moreover, Figure 6(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 requires that $\prod_{i=1}^{k} n_i = d^a d^r$, the higher order $k$ also means the smaller $n_{max}$, which reduces the number of parameters. Thus, the reshaped tensor order in GETD should be appropriately determined considering both link prediction performance and model complexity.

5.5 Binary Relational Link Prediction

To investigate the robustness as well as representation capability of our proposed GETD model, we evaluate the performance of GETD model on WN18 and FB15k. The experimental settings are the same as in n-ary relational link prediction. The embedding sizes of GETD are set to 200, which is similar to the setting in TuckER [33]. 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 6 summarizes the results of GETD and the state-of-the-art models on two datasets. According to the results, GETD achieves the second best performance on WN18, with a quite small MRR gap of 0.005 to TuckER. Moreover, TR latent tensors in GETD costs
3 \cdot 50 \times 200 \times 50 = 1.5 \text{ million parameters, only } 1/8 \text{ of core tensor parameters in TUCKER} (200 \times 200 \times 200 = 8 \text{ million}). Thus, GETD is able to obtain better performance with larger embedding sizes but similar number of parameters. As for FB15k, GETD outperforms all state-of-the-art models, and increases MRR by 0.03. Also, GETD increases the toughest metric Hits@1 by 4% on FB15k. These results demonstrate that, GETD is robust and works well in representing KBs with different arity relations.

6 CONCLUSION

This work proposed GETD, a fully expressive tensor decomposition model for link prediction in n-ary relational KBs. Based on the expressiveness of Tucker decomposition as well as the flexibility of tensor ring decomposition, GETD is able to capture the latent interactions between entities and relations with a small number of parameters. Experimental results demonstrate that GETD outperforms the state-of-the-art models for n-ary relational link prediction and achieves close and even better performance on standard binary relational KB datasets.

Considering the benefits of parameter reduction with higher-order reshaped tensor, we plan to extend GETD with appropriate optimization techniques so that GETD with higher-order reshaped tensor can also achieve comparable performance. Besides, GETD only uses observed facts for link prediction, while incorporating GETD with background knowledge such as logical rules and entity properties may bring performance enhancement. Finally, we also consider using automated machine learning techniques to search the scoring function from the data [41, 44].

ACKNOWLEDGMENTS

This work was supported in part by The National Key Research and Development Program of China under grant 2018YFB1800804, the National Nature Science Foundation of China under U1836219, 61971267, 61972223, 61861136003, Beijing National Science Foundation under L182038, Beijing National Research Center for Information Science and Technology under 20031887521, and research fund of Tsinghua University - Tencent Joint Laboratory for Internet Innovation Technology.

A PROOFS

A.1 Preliminaries

Now, we first introduce lemmas that will be used later in the proof.

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

**Proof.** Let \( e_i, e_{i_1}, \ldots, e_{i_n} \) be the \( n_e \)-dimensional one-hot binary vector representation of entities \( i, i_1, \ldots, i_n \), and \( r_j \) the \( n_r \)-dimensional one-hot binary vector representation of a relation \( i_j \). We let the \( i_{j_1}\text{-th}, i_{j_2}\text{-th}, \ldots, i_{j_\mathcal{R}}\text{-th element respectively of the corresponding vectors } r_{j}, e_{i_1}, e_{i_2}, \ldots, e_{i_{n}}\text{ be } 1\text{ and all other elements } 0\text{. Further, we set the } w_{i,j} \text{ of the core tensor to } 1\text{ if the fact } (i, i_1, i_2, \ldots, i_{n})\text{ holds and } 0\text{ otherwise. Thus the tensor product of these entity embeddings and the relation embedding with the core tensor, accurately represents the original KB tensor.} \quad \Box

**Lemma 2.** Given any k-th 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 \( r_{\mathcal{CP}} = \prod_{i=1}^{k} n_i \) that completely decomposes \( \mathcal{W} \).

**Proof.** Since \( \mathcal{W} \in \{0,1\}^{n_1 \times n_2 \times \cdots \times n_k} \), we can use \( r_{\mathcal{CP}} = \prod_{i=1}^{k} n_i \) zero/one-hot tensors \( \{\mathcal{W}^{(1)}, \mathcal{W}^{(2)}, \ldots, \mathcal{W}^{(k)}\} \) to represent \( \mathcal{W} \), s.t., \( \mathcal{W}^{(1)} = w_{1,-1} \), while other elements in \( \mathcal{W}^{(1)} \) are zeros, \( \mathcal{W}^{(r)} = w_{1,1,\ldots,1} \), while other elements in \( \mathcal{W}^{(r)} \) are zeros, etc. Finally, \( \mathcal{W}^{(r)} \) are zeros. Moreover, \( \mathcal{W}^{(r)} \) can be decomposed by rank-one CP decomposition as,

\[
\mathcal{W}^{(r)} = u^{(1)} \circ u^{(2)} \circ \cdots \circ u^{(k)},
\]

s.t.
\[
w_{i_1,j_1,\ldots,j_k} = u^{(1)}(j_1) \\times u^{(2)}(j_2) \cdots \times u^{(k)}(j_k),
\]

where \( u^{(1)} \) is the \( n_1 \)-dimensional zero/one-hot vector, and \( u^{(j)}(j) \) is the \( j \)-th element of the vector. Therefore, by assigning \( u^{(1)}(j_1) = u^{(2)}(j_2) = \cdots = u^{(k)}(j_k) = w_{i_1,j_1,\ldots,j_k} \), and other elements in \( u^{(j)} \) being zeros, the binary tensor \( \mathcal{W} \) can be completely decomposed by CP decomposition via the set of vectors \( \{u^{(i)}\}_{i=1}^{r_{\mathcal{CP}}} \), where

\[
\mathcal{W} = \sum_{r_{\mathcal{CP}}} u^{(i)} \circ u^{(2)} \circ \cdots \circ u^{(k)}.
\]

**Lemma 3.** Canonical polyadic (CP) decomposition can be viewed as a special case of tensor ring (TR) decomposition. Given a k-th order binary tensor \( \mathcal{W} \in \{0,1\}^{n_1 \times n_2 \times \cdots \times n_k} \) with its CP decomposition as (12), it can also be written in TR decomposition form as,

\[
\mathcal{W} = \sum_{r_{\mathcal{TR}}} u^{(i)} \circ u^{(2)} \circ \cdots \circ u^{(k)} = TR(\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_k),
\]

s.t.
\[
Z_i(j_i) = \text{diag}(u^{(i)}(j_i)), \quad \forall i = 1, 2, \ldots, m
\]

where \( \mathcal{Z}_i = \{0,1\}^{n_1 \times n_2 \times \cdots \times n_k} \) and \( u^{(i)}(j_i) = [u^{(i)}(j_1), u^{(i)}(j_2), \ldots, u^{(i)}(j_k)] \).

See proof in [46].

A.2 Theorem 1

**Proof.** According to Lemma 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 \( \mathcal{W} \in \{0,1\}^{n_1 \times n_2 \times \cdots \times n_k} \).

In GETD, \( \mathcal{W} \) is reshaped into a k-th order reshaped tensor \( \hat{\mathcal{W}} \in \{0,1\}^{n_1 \times n_2 \times \cdots \times n_k} \), which is further decomposed by TR decomposition. Keeping the embedding settings as the ones in Lemma 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 2, \( \mathcal{W} \) 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 3, which derives TR latent tensors \( \{\mathcal{Z}_i\}_{i=1}^{m} \in \{0,1\}^{n_1 \times n_2 \times \cdots \times n_k} \). Overall, following the settings of embeddings in Lemma 1 and TR latent tensors in Lemma 3, GETD is proved to be fully expressive with entity embeddings of dimensionality \( n_e = |E| \) and relation embeddings of dimensionality \( d_r = |\mathcal{R}| \). \quad \Box
