Scalable Label Propagation for Multi-relational Learning on Tensor Product Graph

Zhuliu Li*
Department of Computer Science and
Engineering, University of Minnesota
Minneapolis, MN
lixx3617@umn.edu

Raphael Petegrosso*
Department of Computer Science and
Engineering, University of Minnesota
Minneapolis, MN
peteg001@umn.edu

Shaden Smith
Department of Computer Science and
Engineering, University of Minnesota
Minneapolis, MN
shaden@cs.umn.edu

David Sterling
Department of Radiation Oncology,
University of Minnesota
Minneapolis, MN
sterl035@umn.edu

George Karypis
Department of Computer Science and
Engineering, University of Minnesota
Minneapolis, MN
karypis@cs.umn.edu

Rui Kuang
Department of Computer Science and
Engineering, University of Minnesota
Minneapolis, MN
kuang@cs.umn.edu

ABSTRACT

Label propagation on the tensor product of multiple graphs can infer multi-relations among the entities across the graphs by learning labels in a tensor. However, the tensor formulation is only empirically scalable up to three graphs due to the exponential complexity of computing tensors. In this paper, we propose an optimization formulation and a scalable Lowrank Tensor-based Label Propagation algorithm (LowrankTLP). The optimization formulation minimizes the rank-k approximation error for computing the closed-form solution of label propagation on a tensor product graph with efficient tensor computations used in LowrankTLP. LowrankTLP takes either a sparse tensor of known multi-relations or pairwise relations between each pair of graphs as the input to infer unknown multi-relations by semi-supervised learning on the tensor product graph. We also accelerate LowrankTLP with parallel tensor computation which enabled label propagation on a tensor product of 100 graphs of size 1000 within 150 seconds in simulation. LowrankTLP was also successfully applied to multi-relational learning for predicting author-paper-venue in publication records, alignment of several protein-protein interaction networks across species and alignment of segmented regions across up to 7 CT scan images. The experiments prove that LowrankTLP indeed well approximates the original label propagation with high scalability.

Source code: https://github.com/kuanglab/LowrankTLP

KEYWORDS

multi-relational learning, tensor product graph, label propagation, link prediction, tensor decomposition and completion

1 INTRODUCTION

Label propagation has been widely used for semi-supervised learning on the similarity graph of labeled and unlabeled samples [20, 24, 25]. As illustrated in Figure 1 (top left), label propagation propagates training labels on a graph to learn a vector $\boldsymbol{y}$ predicting the labels of each vertex. A generalization of label propagation on the Kronecker product of two graphs (also called bi-random walk) can infer the pairwise relations in a matrix Y between the vertices from the two graphs as shown in Figure 1 (bottom left). This approach

has been applied to aligning biological and biomedical networks [17, 22]. Similar link prediction problems on Kronecker product graphs also exist in matching images [6], collaborative filtering [11], citation network analysis [12] and multi-language translation [23]. When applied on the tensor product of n graphs to predict the matching across the n graphs in an n-way tensor $\mathcal{Y}$ as shown in Figure 1 (right), label propagation accomplishes n-way relational learning from knowledge graphs [14]. Label propagation on the tensor product graph explores the graph topologies for associating vertices across the graphs assuming the global relations among the vertices reveal the vertex identities [22]. However, the tensor formulation of label propagation is only empirically scalable up to three graphs even if the graphs are sparse [12, 16]. In particular, each multiplication with tensor will exponentially increase the number of non-zero entries and after several iterations, a dense tensor is expected.

To tackle the scalability issue, we introduce an optimization formulation to minimize rank-k approximation error for a low-rank computation of the closed-form solution of label propagation on a tensor product graph, and a Lowrank Tensor-based Label Propagation algorithm (LowrankTLP) using efficient tensor operations to compute the approximated solution. We also show that our optimization formulation based on the rank-k tensor product graph incurs less approximation error than the direct rank-k approximation of the global similarity matrix for computing the closed-form solution. Finally, we implemented a parallel LowrankTLP using SPLATT library [18] for parallel shared-memory tensor operations to increase the scalability by a large magnitude.
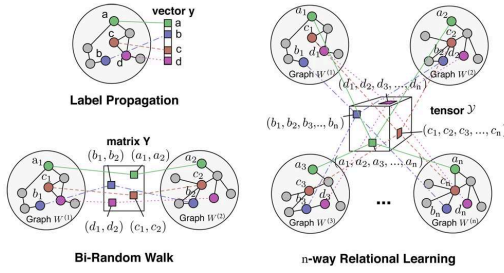
2 PRELIMINARIES

Below is a list of tensor notations and definitions. Four useful lemmas are also given in Appendix A. For more general information of tensors, we direct the readers to the survey paper [9].

(1) **Notations and operators:**

vector:	$\boldsymbol{x}$	Hadamard product:	$*$	outer product:	$\circ$
matrix:	X	Kronecker product:	$\otimes$	the i-mode product:	$\times_i$
tensor:	$\mathcal{X}$	Khatri–Rao product:	$\odot$	vectorization of tensor:	$\overrightarrow{\mathcal{X}}$

*Equal contribution

Figure 1: Label propagation generalized on tensor product graphs. Top left: label propagation on a graph predicts the labels of the vertices for semi-supervised learning; Bottom left: label propagation on the Kronecker product of two graphs predicts links between the vertices across the two graphs; Right: label propagation on an n-way tensor product graph learns n-way multi-relations across the vertices in n graphs. Each n vertices in the same color is represented as an entry in the n-way tensor.

(2) **CANDECOMP/PARAFAC decomposition (CP):** An n-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_n}$ of rank R can be written as

$$\mathcal{X} = \sum_{r=1}^{R} a_r^{(1)} \circ a_r^{(2)} \circ \cdots \circ a_r^{(n)}$$
$$= [\![A^{(1)}, A^{(2)}, \ldots A^{(n)}]\!],$$

where $a_r^{(i)}$ is the r-th column of factor matrix $A^{(i)} \in \mathbb{R}^{I_i \times R}$. *Vectorization property:* The vectorization of CP-form is $\overrightarrow{\mathcal{X}} = (A^{(n)} \odot A^{(n-1)} \odot \cdots \odot A^{(1)})\mathbf{1}$, where $\mathbf{1}$ is a vector with all-ones.

(3) **Tucker Decomposition:** An n-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_n}$ can be decomposed into a core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \ldots \times R_n}$ and factor matrices $\{A^{(i)} \in \mathbb{R}^{I_i \times R_i} | i = 1, \ldots, n\}$ as

$$\mathcal{X} = \mathcal{G} \times_1 A^{(1)} \times_2 A^{(2)} \cdots \times_n A^{(n)}$$
$$= [\![\mathcal{G}; A^{(1)}, A^{(2)}, \ldots, A^{(n)}]\!],$$

where $\times_i$ is the i-mode matrix product of a tensor. *Vectorization property:* The vectorization of $\mathcal{X}$ is $\overrightarrow{\mathcal{X}} = (A^{(n)} \otimes A^{(n-1)} \ldots \otimes A^{(1)})\overrightarrow{\mathcal{G}}$.

(4) **Tensor product graph (TPG):** Given the adjacency matrices of n distinct graphs $\{W^{(i)} \in \mathbb{R}^{I_i \times I_i} | i = 1, \ldots, n\}$, the tensor product graph is defined as $W = \otimes_{i=1}^{n} W^{(i)} \in \mathbb{R}^{(\prod_i^n I_i) \times (\prod_i^n I_i)}$, with edge $w_{\{a_1, a_2, \ldots, a_n\}, \{b_1, b_2, \ldots, b_n\}} = \prod_{l=1}^{n} w_{a_l, b_l}^{(l)}$ encoding the similarity between a pair of n-way tuples of graph vertices $\{a_1, a_2, \ldots, a_n\}$ and $\{b_1, b_2, \ldots, b_n\}$, $\forall \{a_l, b_l\} \in [1 : I_l]$. See Figure 1 (right).

3 PROBLEM FORMULATION

Here, we define the problem of multi-relational learning with knowledge graphs and show an overview of our method in Figure 2.

- **Knowledge graphs** [Figure 2 (A)]: Adjacency matrices of n distinct graphs $\{W^{(i)} \in \mathbb{R}^{I_i \times I_i} | i = 1, \ldots, n\}$ with the symmetric normalized form $S^{(i)} = [D^{(i)}]^{-\frac{1}{2}} W^{(i)} [D^{(i)}]^{-\frac{1}{2}}$, where $D^{(i)}$ is the degree matrix of the i-th graph.

- **Input options** [Figure 2 (B)]: There are usually two possible kinds of inputs in real applications.
  - *Option 1 (known multi-relations)*: an n-way sparse tensor $\mathcal{Y}^0$ stores a small labeled (known) subset of all n-way relations $\{\{i_1, i_2, \ldots, i_n\} | \forall i_j \in [1 : I_j], j \in [1, n]\}$ where i_j denotes the i-th vertex of graph $W^{(j)}$. The zero entries are unlabeled (unknown) n-way relations.
  - *Option 2 (pairwise relations)*: the non-negative matrices $\{R_{ij} \in \mathbb{R}^{I_i \times I_j} | i, j \in [1, n]\}$ contain the pairwise similarities between the vertices of graph $W^{(i)}$ and $W^{(j)}$. We convert the matrices into a rank-r CP-form of $\mathcal{Y}_0$ as input as following: we first apply symmetric NMF (symNMF) [5] on a symmetric matrix built by stacking all R_{ij}'s to obtain a nonnegative factor matrix $F \in \mathbb{R}^{(\Sigma_{i=1}^{n} I_i) \times r}$. Then, the rank-$r$ CP-form of $\mathcal{Y}_0$ is approximated as $\mathcal{Y}_0 = [\![F^{(n)}, F^{(n-1)}, \ldots, F^{(1)}]\!]$ where $F^{(i)} \in \mathbb{R}^{I_i \times r}$ is the i-th submatrix of F. This is under the assumption that more similar pairwise relations between every pair of $\{i_a, i_b\} \subset \{i_1, i_2, \ldots, i_n\}$ imply a stronger relation at the $\{i_1, i_2, \ldots, i_n\}$-th entry of $\mathcal{Y}_0$.
- **Output** [Figure 2 (D)]: a spare tensor $\mathcal{Y}^*$ stores the predicted labels of the n-way relations queried by the user.
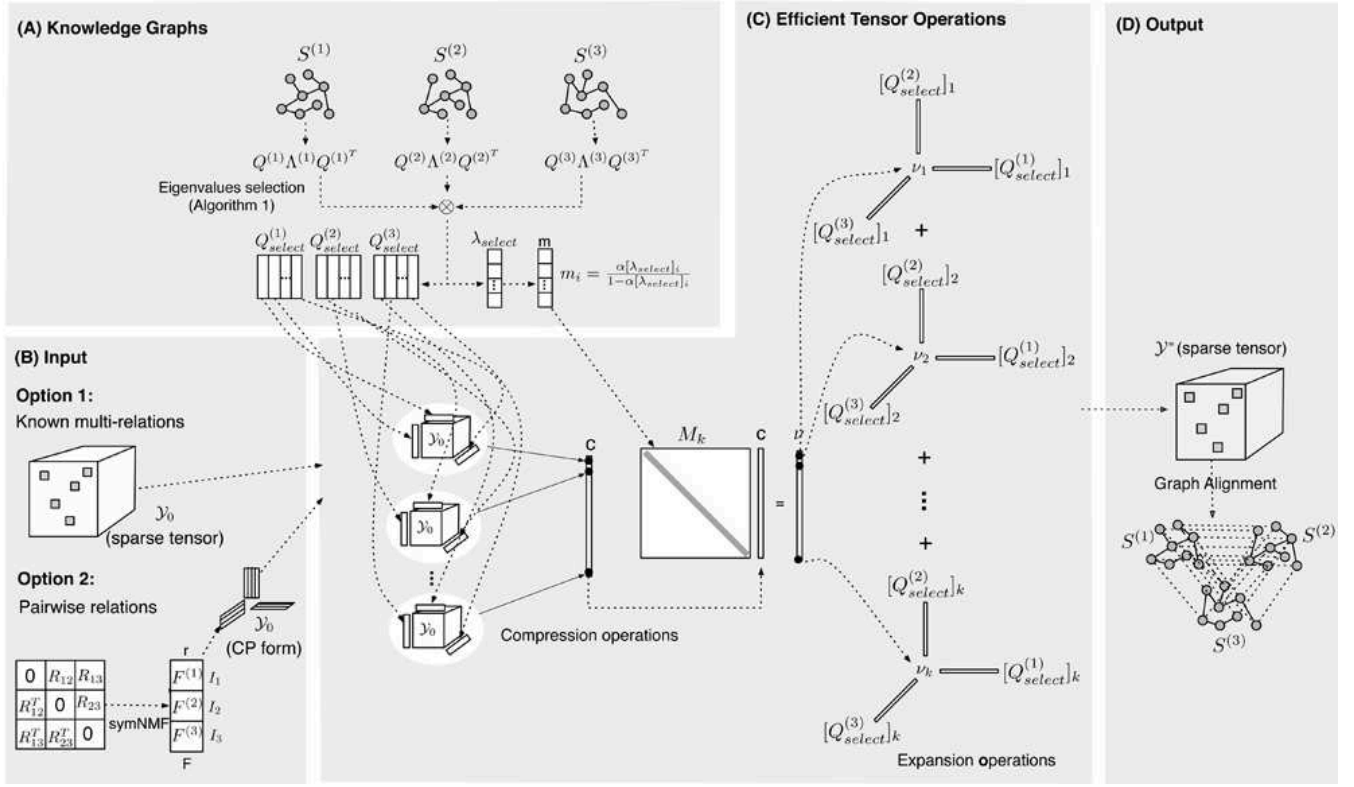
Our goal is to predict the multi-relations of a set of queried entries in the output $\mathcal{Y}^*$ given the known relations in the initial sparse-form or CP-form of the tensor $\mathcal{Y}_0$ based on the topological information carried by the edges of the TPG of the knowledge graphs as described in Section 2(4). This task will be solved by our scalable label propagation algorithm for manifold learning on the TPG.

4 RELATED WORK

Multi-relational learning with knowledge graphs remains a challenge due to the exponential number of possible multi-relations to evaluate even in a small number of graphs of medium sizes. Two categories of tensor-based techniques have been previously applied to the multi-relational learning problem.

The first category applies low-rank approximation on each individual graph to improve the scalability. For example, approximate link propagation (**ApproxLink**) [16] is a semi-supervised learning algorithm for link prediction on a pair of graphs, which is generalizable to multiple graphs. Transductive learning over product graph (**TOP**) [12] is a cross-graph relational learning (CGRL) algorithm introducing the product graph structure via a Gaussian random fields prior. In general, these methods are not suitable for applications with a large number of graphs: First, low-rank approximation of each individual graph does not guarantee a globally optimal approximation of the TPG; and second, the rank of the TPG is exponential in the number of graphs, and therefore not scalable to many graphs. Thus, these approximation methods do not preserve the original performance and still suffers scalability issues to learn from a large number of knowledge graphs.

The second category applies tensor completion with the known multi-relations in the initial tensor to predict the unknown multi-relations.

Figure 2: The general schema of LowrankTLP. (A) Knowledge Graphs: Three graphs, $S^{(1)}$, $S^{(2)}$ and $S^{(3)}$ are given. Then, Algorithm 1 will obtain the k optimal eigenvalues and the corresponding eigenvectors of the tensor product graph based on the eigendecomposition of each graph. (B) Input: Two options of initial tensor $\mathcal{Y}_0$ as input: known multi-relations in a sparse tensor or a CP-form computed with pairwise relations of every pair of graphs. (C) Efficient Tensor Operations: $\mathcal{Y}_0$ and the selected eigen-pairs are used to perform compression operations and expansion operations to obtain the closed-form solution of label propagation in a CP-form. (D) Output: The sparse tensor $\mathcal{Y}$ can be obtained by computing the queried multi-relations, which can also be used to align the graphs.

This problem is usually solved by graph-regularized tensor decomposition (**TD**) assuming that the tensors to be completed are low-rank and the regularization by graph Laplacian encourages the components in the factor matrices to be smooth among strongly connected vertices in the graph [3, 4, 13]. For example, the cross-mode regularization (**GraphTD**) [13] regularizes all the factor matrices together in the graph Laplacian of the TPG. Two major weaknesses of these tensor-completion methods are the lack of unique solution and that the performance tends to decrease quickly when only a small number of known relations are provided.

5　LABEL PROPAGATION ON TENSOR PRODUCT GRAPH

Here, we formally define the generalization of label propagation on TPG and it's closed-from solution. Let S be the normalized TPG as follows,

$$
\begin{aligned}
S &= (\otimes_{i=1}^n [D^{(i)}]^{-\frac{1}{2}})(\otimes_{i=1}^n W^{(i)})(\otimes_{i=1}^n [D^{(i)}]^{-\frac{1}{2}}) \\
&= \otimes_{i=1}^n ([D^{(i)}]^{-\frac{1}{2}} W^{(i)} [D^{(i)}]^{-\frac{1}{2}}) \text{ (by Lemma A.1)} \\
&= \otimes_{i=1}^n S^{(i)}.
\end{aligned}
$$

The iterations of label propagation on vectorized tensor $\overrightarrow{\mathcal{Y}}$ starting at the initialization $\overrightarrow{y^0}$ defined in section 3 is as follows:

$$
\overrightarrow{y^{t+1}} = \alpha(\otimes_{i=1}^n S^{(i)})\overrightarrow{y^t} + (1-\alpha)\overrightarrow{y^0}, \tag{1}
$$

where $\alpha \in (0,1)$ is a balancing parameter and t denotes the iteration. Applying the *vectorization property* of Tucker decomposition as introduced in Section 2(3), Equation (1) can be rewritten as

$$
\mathcal{Y}^{t+1} = \alpha \mathcal{Y}^t \times_1 S^{(n)} \times_2 S^{(n-1)} \cdots \times_n S^{(1)} + (1-\alpha)\mathcal{Y}^0. \tag{2}
$$

Even if $\mathcal{Y}^0$ is in sparse form or CP form, the density of tensor $\mathcal{Y}$ increases exponentially in each iteration. Therefore, the space complexity is $O(\prod_{i=1}^n I_i)$ for store the dense tensor and the time complexity is $O((\prod_{i=1}^n I_i)(\sum_{i=1}^n I_i))$ per iteration.

Since the eigenvalues of S are in $[-1, 1]$ and $\alpha \in (0, 1)$, iteration (1) converges to the following closed-form solution [24],

$$
\overrightarrow{y^*} = \lim_{t \to \infty} \overrightarrow{y^t} = (1-\alpha)(I - \alpha S)^{-1}\overrightarrow{y^0}. \tag{3}
$$

Furthermore, given the eigen-decomposition of each $S^{(i)}$ as $\{S^{(i)} = Q^{(i)}\Lambda^{(i)}Q^{(i)T} | i = 1, \dots n\}$, the eigen-decomposition of S can be

expressed as

$$S = Q\Lambda Q^T = (\otimes_{i=1}^n Q^{(i)})(\otimes_{i=1}^n \Lambda^{(i)})(\otimes_{i=1}^n Q^{(i)T})$$

with Lemma A.1 and A.4. Substituting S into Equation (3) we get

$$\overrightarrow{\boldsymbol{y}^*} = (1-\alpha)(\otimes_{i=1}^n Q^{(i)})(I - \alpha(\otimes_{i=1}^n \Lambda^{(i)}))^{-1}(\otimes_{i=1}^n Q^{(i)T})\overrightarrow{\boldsymbol{y}^0}. \quad (4)$$

It can be observed that computing the closed-form solution in Equation (4) from right to left needs $2n$ matrix-tensor products in total with the *vectorization property* of Tucker decomposition. Therefore, the space and time complexity will be the same as running two iterations of label propagation, apart from computing the eigen-decompositions of all the normalized graphs $\{S^{(i)}|i = 1, ..., n\}$. In [12, 16], Eckart-Young-Mirsky theorem [7] is applied on each individual graph to approximate their TPG. However, this approximation does not consider the overall spectral of the TPG and thus, leads to a large error in both approximation and multi-relational predictions. In other words, the direct application of Eckart-Young-Mirsky theorem to individual graphs results in a large perturbation on the whole transformation matrix $(I - \alpha S)^{-1}$, and thus, an inaccurate computation of the closed-form solution.

6 LOW-RANK LABEL PROPAGATION

In this section, we first introduce our optimization formulation and then the LowrankTLP algorithm illustrated in Figure 2 to solve the optimization problem. We also show that our formulation incurs less approximation error than the direct rank-k approximation of the global similarity matrix for computing the closed-form solution.

6.1 Learning rank-k approximation

We propose to approximate the closed-form solution in Equation (4) by minimizing the perturbation on transformation matrix $(I - \alpha S)^{-1}$ as follows,

$$\begin{aligned}
& \underset{eig(S_k)}{\text{minimize}} && ||(I - \alpha S)^{-1} - (I - \alpha S_k)^{-1}||_{2,F} \\
& \text{subject to} && \text{rank}(S_k) = k, \; eig(S_k) \subseteq eig(S),
\end{aligned} \quad (5)$$

where $eig(S_k)$ and $eig(S)$ denote the sets of eigen-pairs of S_k and S respectively. The objective is to find a low-rank matrix S_k defined by a subset of eigen-pairs of S that gives the lowest divergence on the overall transformation matrix $(I - \alpha S)^{-1}$ in the closed-form solution in Equation (3). Next, we will show this optimization problem has a global optimal solution and can be solved by Algorithm 1.

Let $S_k = Q_{1:k}\Lambda_{1:k}Q_{1:k}^T$ be the eigen-decomposition of S_k, where $\Lambda_{1:k}$ and $Q_{1:k}$ store the eigenpairs $\{(\lambda_j, \boldsymbol{q}_j)|j = 1, \ldots, k\}$ of S_k selected from $eig(S)$. Also, define diagonal matrix Λ_{rest} and matrix Q_{rest} to contain the remaining eigen-pairs $\{(\lambda_i, \boldsymbol{q}_i)|i = k+1, \ldots, N\}$ of $eig(S)$, where $N = \prod_{i=l}^n I_l$.

PROPOSITION 1. *Let $A = (I - \alpha S)^{-1}$ and its approximation $\hat{A} = (I - \alpha S_k)^{-1}$, we have*

$$\hat{A} = Q_{1:k}((I - \alpha\Lambda_{1:k})^{-1} - I)Q_{1:k}^T + I, \quad (6)$$

Proof: by definition of S_k we have $\hat{A} = (I - \alpha Q_{1:k}\Lambda_{1:k}Q_{1:k}^T)^{-1}$, which can be further expanded as $Q_{1:k}((I - \alpha\Lambda_{1:k})^{-1} - I)Q_{1:k}^T + I$ by Woodbury formula [8]. (End of Proof)

THEOREM 6.1. *The optimal k eigenvalues $\{\lambda_j|j = 1, \ldots, k\}$ that solve the optimization problem (5) are a subset of the union of the k largest (algebraic) and k smallest (algebraic) eigenvalues of S and satisfy the following condition*

$$\frac{\alpha|\lambda_j|}{1 - \alpha\lambda_j} \geq \frac{\alpha|\lambda_i|}{1 - \alpha\lambda_i}, \forall j \in [1, k], \forall i \in [k+1, N].$$

Proof: the perturbation can be obtained as

$$\hat{A} - A = Q_{\text{rest}}(I - (I - \alpha\Lambda_{\text{rest}})^{-1})Q_{\text{rest}}^T, \quad (7)$$

whose singular values are $\{\frac{\alpha|\lambda_i|}{1-\alpha\lambda_i}|i = k+1, \ldots, N\}$ and k zeros. Thus, it's spectral norm and Frobenius norm are

$$||\hat{A} - A||_2 = \frac{\alpha|\lambda^*|}{1 - \alpha\lambda^*} \quad \text{and} \quad ||\hat{A} - A||_F = \sqrt{\sum_{i=k+1}^N (\frac{\alpha|\lambda_i|}{1 - \alpha\lambda_i})^2}, \quad (8)$$

where λ^* could be either the largest positive value or the smallest negative value of $\lambda_{k+1}, \ldots, \lambda_N$. To minimize both norms in Equation (8), the k selected eigenvalues $\{\lambda_j|j = 1, \ldots, k\}$ should give the k largest $\{\frac{\alpha|\lambda_j|}{1-\alpha\lambda_j}|j = 1, \ldots, k\}$ among all the eigenvalues of S. In addition, since $\alpha \in (0, 1)$ and $\lambda_j \in [-1, 1], j = 1, \ldots, k$, the function $\frac{\alpha|\lambda_j|}{1-\alpha\lambda_j}$ is monotonically increasing in the positive orthant and decreasing in the negative orthant with λ_j. Thus, $\{\lambda_j|j = 1, \ldots, k\}$ must be in the union of the k largest (algebraic) eigenvalues and k smallest (algebraic) eigenvalues of S. (End of Proof)

Let $\lambda_j^{(i)}$ and $\boldsymbol{q}_j^{(i)}$ be the eigenvalue and its corresponding eigenvector of $S^{(i)}$ contributing to λ_j and $\boldsymbol{q}_j$. Since $S = \otimes_{i=1}^n S^{(i)}$, by Lemma A.4 we have

$$\lambda_j = \prod_{i=1}^n \lambda_j^{(i)} \quad \text{and} \quad \boldsymbol{q}_j = \otimes_{i=1}^n \boldsymbol{q}_j^{(i)}, j = 1, \ldots k. \quad (9)$$

THEOREM 6.2. *Given the vector $\boldsymbol{\lambda}^{(i)}$ of the eigenvalues of $S^{(i)}$ for $i = 1, \ldots n$ and function $\boldsymbol{top\_bot\_2k}(\boldsymbol{x})$ which returns the k largest numbers union the k smallest numbers in vector $\boldsymbol{x}$, we have*

$$\boldsymbol{top\_bot\_2k}(\otimes_{i=1}^n \boldsymbol{\lambda}^{(i)}) = \boldsymbol{top\_bot\_2k}(\boldsymbol{\lambda}^{(n)} \otimes \boldsymbol{top\_bot\_2k}(\Gamma^{(n-1)})),$$

$$\Gamma^{(i)} = \begin{cases} \boldsymbol{\lambda}^{(i)} \otimes \boldsymbol{top\_bot\_2k}(\Gamma^{(i-1)}), & \text{if } i = 2, \ldots, n-1 \\ \boldsymbol{\lambda}^{(1)}, & \text{if } i = 1. \end{cases}$$

Theorem 6.2 can be easily proved by induction based on the observation that the k largest elements in the outer product of two real vectors can only be among the multiplications between the top k numbers union the bottom k numbers in the two vectors. Thus, only the numbers in $\boldsymbol{top\_bot\_2k}(\Gamma^{(i-1)})$ are needed to compute the next $\Gamma^{(i)}$ in the recursion. Taking the elements in $\boldsymbol{top\_bot\_2k}(\Gamma^{(i-1)})$ in the multiplication with each $\boldsymbol{\lambda}^{(i)}$ guarantees that the numbers needed for computing the k largest elements in $\otimes_{i=1}^n \boldsymbol{\lambda}^{(i)}$ will be kept in $\Gamma^{(i)}$.

Based on Theorem 6.2 we propose Algorithm 1 to select $\{(\lambda_j^{(i)}, \boldsymbol{q}_j^{(i)})| i = 1, \ldots, n, j = 1, \ldots, k\}$ pairs efficiently in time $O(\sum_{i=1}^n (kI_i \log(kI_i)))$ and output a vector $\boldsymbol{\lambda}_{\text{select}}$ storing the selected eigenvalues of S and matrices $Q_{\text{select}}^{(i)}$ storing the selected eigenvectors among $\{Q^{(i)}$ —

Algorithm 1 Select Eigenvalues

1: **Input:** $\{S^{(i)}|i = 1, \ldots n\}$, α
2: **Output:** $\boldsymbol{\lambda}_{\text{select}}$ and $\{Q^{(i)}_{\text{select}}|i = 1, \ldots, n\}$
3: store the eigenvalues of $S^{(i)}$ in vector $\boldsymbol{\lambda}^{(i)}$, for $i = 1, \ldots n$.
4: $\Gamma \leftarrow \boldsymbol{\lambda}^{(1)}$
5: **for** $i = 2$ to n **do**
6:    $\Gamma \leftarrow \boldsymbol{\lambda}^{(i)} \otimes \textbf{top\_bot\_2k}(\Gamma)$
7: **end for**
8: $\boldsymbol{\lambda}_{\text{select}} \leftarrow k$ elements from Γ with the largest $\frac{\alpha|\Gamma_j|}{1-\alpha\Gamma_j}, j = 1, \ldots, k$

9: **for** $i = n$ to 1 **do**
10:    return $Q^{(i)}_{\text{select}}$ from $Q^{(i)}$ by looking-up indexes of the values output by function **top_bot_2k**().
11: **end for**

$i = 1, \ldots, n\}$ defined as

$$\boldsymbol{\lambda}_{\text{select}} = [\lambda_1, \lambda_2, \ldots, \lambda_k]^T \tag{10}$$

$$Q^{(i)}_{\text{select}} = [\boldsymbol{q}^{(i)}_1, \boldsymbol{q}^{(i)}_2, \ldots, \boldsymbol{q}^{(i)}_k]. \tag{11}$$

Define $M = (I - \alpha\Lambda_{1:k})^{-1} - I$ computed with $\boldsymbol{\lambda}_{\text{select}}$ as

$$M = \text{diag}\left(\left[\frac{\alpha\lambda_1}{1-\alpha\lambda_1}, \frac{\alpha\lambda_2}{1-\alpha\lambda_2}, \ldots, \frac{\alpha\lambda_k}{1-\alpha\lambda_k}\right]\right).$$

The matrix $Q_{1:k}$ can be computed from $\{Q^{(i)}_{\text{select}}|i = 1, \ldots, n\}$ as $Q_{1:k} = \odot^n_{i=1}Q^{(i)}_{\text{select}}$. By Equation (6), the closed-form solution can be approximated as

$$\overrightarrow{\mathcal{Y}^*} = (1-\alpha)\hat{A}\overrightarrow{\mathcal{Y}^0} \tag{12}$$

$$= (1-\alpha)(\odot^n_{i=1}Q^{(i)}_{\text{select}})M(\odot^n_{i=1}Q^{(i)}_{\text{select}})^T\overrightarrow{\mathcal{Y}^0} + (1-\alpha)\overrightarrow{\mathcal{Y}^0}. \tag{13}$$

6.2 LowrankTLP algorithm

Equation (13) implies a 2-step tensor computation of the closed-form solution given in Algorithm 2. The two tensor operations are also illustrated in Figure 2.

Compression step (line 4-15 in Algorithm 2)

- $\mathcal{Y}_0$ **is sparse (option 1)**: the role of $(\odot^n_{i=1}Q^{(i)}_{\text{select}})^T\overrightarrow{\mathcal{Y}^0}$ in Equation (13) is to compress the original tensor $\mathcal{Y}^0$ to a k-D vector $\boldsymbol{v}$ with its jth element

$$\boldsymbol{v}_j = \mathcal{Y}_0\bar{\times}_1\boldsymbol{q}^{(n)}_j\bar{\times}_2\boldsymbol{q}^{(n-1)}_j \ldots \bar{\times}_n\boldsymbol{q}^{(1)}_j, \tag{14}$$

where each $\bar{\times}_i$ denotes mode-i vector product of tensor. In Equation (14), the original tensor $\mathcal{Y}^0$ is compressed to a scalar by multiplying with n vectors which is similar to computing the core tensor in Tucker decomposition. Let the number of nonzeros in $\mathcal{Y}^0$ be $|\mathcal{Y}^0|$, the time complexity of the compression step is $O(|\mathcal{Y}^0|nk)$.

    The construction of $\boldsymbol{v}$ via Equation (14) performs j sequences of n-way tensor-vector products. The same kernel appears in a similar form involving $n-1$ products during the computation of the CP. We can leverage parallel algorithms which have been developed to compute the CP [19].

The resulting computation takes the form

$$Z \leftarrow Y^0_{(1)}(Q^{(2)}_{\text{select}} \odot \cdots \odot Q^{(n)}_{\text{select}}), \tag{15}$$

$$\boldsymbol{v}_j \leftarrow \boldsymbol{q}^{(1)T}_j\boldsymbol{z}_j \qquad \forall j = 1, \ldots, k, \tag{16}$$

where $Y^0_{(1)}$ denotes flattening $\mathcal{Y}^0$ to a matrix.

- $\mathcal{Y}_0$ **in CP-form (option 2)**: when the initial tensor $\mathcal{Y}_0$ is in the CP-form $[\![F^{(n)}, F^{(n-1)}, \ldots, F^{(1)}]\!]$ the k-D vector $\boldsymbol{v}$ can be obtained by

$$\boldsymbol{v} = (\odot^n_{i=1}Q^{(i)}_{\text{select}})^T(\odot^n_{i=1}F^{(i)})\mathbf{1} \tag{17}$$

$$= \textasteriskcentered^n_{i=1}(Q^{(i)T}_{\text{select}}F^{(i)})\mathbf{1}, \tag{18}$$

where Equation (17) is obtained by *vectorization property* of CP-form (Section 2(2)) and Equation (18) can be derived from Lemma A.2. Since each $Q^{(i)T}_{\text{select}}F^{(i)}$ takes $O(krI_i)$, the time complexity of the compression step becomes only $O(kr\sum^n_{i=1}I_i)$.

Expansion step: (line 18-21 in Algorithm 2)
After obtaining the k-D vector $\boldsymbol{v}$ which is then multiplied by the diagonal matrix M to obtain another k-D vector $\hat{\boldsymbol{v}}$, the second step is to compute

$$\overrightarrow{\boldsymbol{y}^*} = (1-\alpha)((\odot^n_{i=1}Q^{(i)}_{\text{select}})\hat{\boldsymbol{v}} + \overrightarrow{\boldsymbol{y}^0}). \tag{19}$$

The left term of (19) has the same form of the vectorized CP decomposition with factor matrices $\{Q^{(i)}_{\text{select}}|i = 1, \ldots, n\}$ (Section 2(2)). Thus, the tensorized version can be obtained as

$$\mathcal{Y}^* = (1-\alpha)([\![\hat{\boldsymbol{v}}'; Q^{(n)}_{\text{select}}, Q^{(n-1)}_{\text{select}}, \ldots Q^{(1)}_{\text{select}}]\!] + \mathcal{Y}^0), \tag{20}$$

where $\hat{\boldsymbol{v}}'$ is a reversal of the elements in $\hat{\boldsymbol{v}}$. From Equation (20) together with the initial tensor $\mathcal{Y}^0$, the k-D vector $\hat{\boldsymbol{v}}$ and matrices $\{Q^{(i)}_{\text{select}} \in \mathbb{R}^{I_i \times k}|i = 1, \ldots n\}$ store all the information for computing any entry of $\mathcal{Y}^*$ with a time complexity $O(nk)$.

    Combining the compression and expansion steps, the time complexities of the sparse-form input and CP-form input are $O(|\mathcal{Y}^0|nk)$ and $O(kr\sum^n_{i=1}I_i)$, respectively. The time complexity of computing the eigen-decomposition of all the normalized graph is $O(\sum^n_{i=1}I^3_i)$. Thus, assuming $|\mathcal{Y}^0| > \sum^n_{i=1}I^3_i$ the overall time complexity of Algorithm 2 is $O(|\mathcal{Y}^0|nk)$ for sparse input and $O(\sum^n_{i=1}(I^3_i + krI_i))$ for CP-form input.

The space required to store the eigenvectors of all the normalized graphs is $O(\sum^n_{i=1}I^2_i)$; to store the indexes of the selected eigen-pairs is $O(k)$; and to store the initial tensor is $O(|\mathcal{Y}^0|)$ and $O(\sum^n_{i=1}I_ir)$ for sparse and CP-form input, respectively. Thus, the overall space complexity is $O(|\mathcal{Y}^0|+k)$ for sparse input assuming $|\mathcal{Y}^0| > \sum^n_{i=1}I^2_i$ and $O(\sum^n_{i=1}I^2_i) + k)$ for CP-form input.

6.3 Error analysis

Instead of using our objective in Equation (5), another natural alternative is to directly find the best rank-k approximation of A. Theorem 6.3 below shows that our objective in Equation (5) is a better choice.

    THEOREM 6.3. *Let $\hat{A} = (I - \alpha S_k)^{-1}$, where the eigen-pairs of the rank-k matrix S_k are selected by Algorithm 1 and A_k is the best*

Algorithm 2 LowrankTLP

1: **Input:** $\{S^{(i)}|i = 1,\ldots n\}$, $\mathcal{Y}^0$, α, k and a multi-relation set $\Omega = \{\{i_1, i_2, \ldots, i_n\}|i_l \in [1 : I_l]\}$ queried by the user.
2: **Output:** Sparse tensor $\mathcal{Y}^*$
3: Apply Algorithm 1 to obtain $\boldsymbol{\lambda}_{\text{select}}$, $\{Q_{\text{select}}^{(i)}|i = 1, \ldots, n\}$
4: Initialize $\boldsymbol{v}$ to be a k-D vector with all zeros.
5: **if** $\mathcal{Y}^0$ is sparse **then**
6:     **for** j=1 to k **do**
7:        $\boldsymbol{v}_j \leftarrow \mathcal{Y}_0 \bar{\times}_1 \boldsymbol{q}_j^{(n)} \bar{\times}_2 \boldsymbol{q}_j^{(n-1)} \ldots \bar{\times}_n \boldsymbol{q}_j^{(1)}$
8:     **end for**
9: **else if** $\mathcal{Y}^0$ is in CP-form $[\![F^{(n)}, F^{(n-1)}, \ldots, F^{(1)}]\!]$ **then**
10:     $\Psi \leftarrow Q_{select}^{(1)T} F^{(1)}$
11:     **for** j=2 to k **do**
12:        $\Psi \leftarrow \Psi * Q_{select}^{(j)T} F^{(j)}$
13:     **end for**
14:     $\boldsymbol{v} \leftarrow \Psi \mathbf{1}$
15: **end if**
16: $m \leftarrow \alpha \boldsymbol{\lambda}_{\text{select}} / (1 - \alpha \boldsymbol{\lambda}_{\text{select}})$
17: $\hat{\boldsymbol{v}}' \leftarrow (\boldsymbol{v} * m)'$
18: Initialize $\mathcal{Y} = \{\}$ to be an empty tensor
19: **for** every tuple $\{i_1, i_2, \ldots, i_n\}$ in Ω **do**
20:     $\mathcal{Y}_{i_n, i_{n-1}, \ldots, i_1}^* \leftarrow (1-\alpha)(\sum_{j=1}^k \hat{\boldsymbol{v}}'_j \prod_{l=1}^n q_{i_l,k}^{(l)} + \mathcal{Y}_{i_n, i_{n-1}, \ldots, i_1}^0)$
21: **end for**

---

rank-k approximation to A in both spectral and Frobenius norm. We have

$$||\hat{A} - A||_2 < ||A_k - A||_2 \text{ and } ||\hat{A} - A||_F < ||A_k - A||_F, \quad (21)$$

where $||.||_2$ is spectral norm and $||.||_F$ is Frobenius norm.

Proof: Let $\sigma_1 > \sigma_2 > \cdots > \sigma_N$ be the sorted eigenvalues of matrix S. Since $\sigma_i \in [-1, 1], i = 1, \ldots N$ and $\alpha \in (0, 1)$, by Eckart-Young-Mirsky theorem, the eigenvalues of A_k are $\{\frac{1}{1-\alpha\sigma_i}|i = 1, \ldots, k\}$ and the perturbations are

$$||A_k - A||_2 = \frac{1}{1 - \alpha\sigma_{k+1}}, \quad (22)$$

$$||A_k - A||_F = \sqrt{\sum_{i=k+1}^{N} (\frac{1}{1 - \alpha\sigma_i})^2}. \quad (23)$$

Using $\{\sigma_i|i = 1, \ldots, k\}$ as eigenvalues and their corresponding eigenvectors of S to construct a rank-k matrix L and define $B = (I - \alpha L)^{-1}$. We have $||\hat{A} - A||_2 \leq ||B - A||_2$ and $||\hat{A} - A||_F \leq ||B - A||_F$ by definition of $\hat{A}$. Also, it is not hard to show $||B - A||_2 < ||A_k - A||_2$ and $||B - A||_F < ||A_k - A||_F$ by the facts that every $\sigma_i \in [-1, 1]$ and $\alpha \in (0, 1)$. Thus, inequalities (21) hold. (End of Proof)

7 EXPERIMENTS

In the experiments, LowrankTLP was applied for multi-relational learning in simulation and three real datasets: DBLP dataset of scientific publication records, segmented CT Scan images and protein-protein interaction (PPI) networks. LowrankTLP was compared with four baselines ApproxLink [16], TOP [12], TD (tensor decomposition) and GraphTD [13] in the simulation, DBLP data and CT Scan images data. For global alignment of multiple PPI networks

LowrankTLP was compared with IsoRankN [10] and BEAMS [1] which were developed specifically for global PPI network alignment.

7.1 Simulations

Synthetic graphs were generated to evaluate the performance and the scalability. We started with a graph of density 0.1 and size I to generate n distinct graphs by randomly permuting 10% of edges from the common "ancestor" graph so that they share similar structures that can be utilized for alignment. The inputs are the n graphs and a sparse n−way tensor $\mathcal{Y}_0$ with $I/2$ (half) of its diagonal entries set to 1s. We set the other $I/2$ diagonal entries and $I/2$ randomly sampled off-diagonal entries to 0.9 and treat them as positive and negative test samples, respectively. The outputs are scores of the I test entries after label propagation, which can be used to distinguish the positive and negative classes based on the assumption that the vertices indexed by the diagonal entries of the tensor should have high similarities since they come from the same "ancestor" graph and this information should be captured by the TPG.
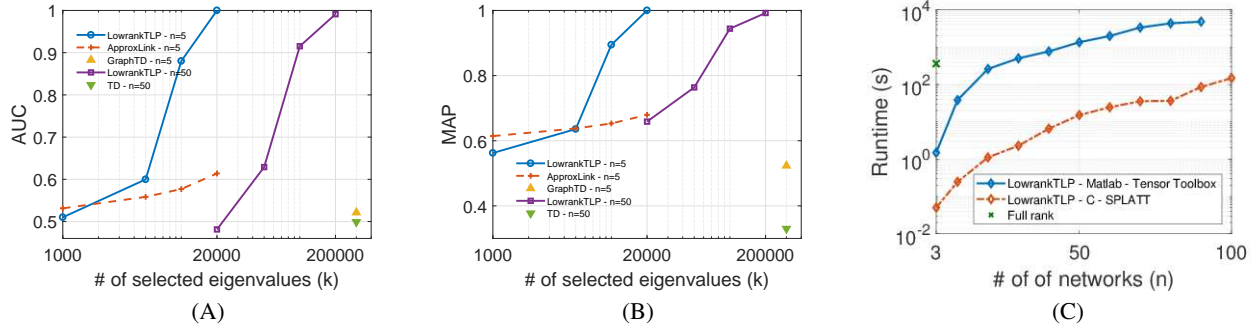
We compared LowrankTLP with ApproxLink, TD and GraphTD using the same sparse tensor $\mathcal{Y}_0$ as input. Rank $\lceil \sqrt[n]{k} \rceil$ approximation is applied to each individual graph for ApproxLink to guarantee their TPG has at least rank k. For fair comparisons, the best parameters are chosen for all the baselines, and fixed $\alpha = 0.1$ is used for LowrankTLP. The area under the curve (AUC) and mean average precision (MAP) are the evaluation metrics.

Figure 3 (A) & (B) show that LowrankTLP clearly outperforms all the baselines to align five graphs at moderate rank $k \geq 3000$. We also observe that ApproxLink performs better than GraphTD which implies that label-propagation-based methods are more robust to sparse input than tensor-completion methods due to the utilization of manifold learning. When $n = 50$, LowrankTLP achieves a high AUC and MAP when $k \geq 100,000$, whereas ApproxLink and GraphTD are not applicable to such a large number of graphs.
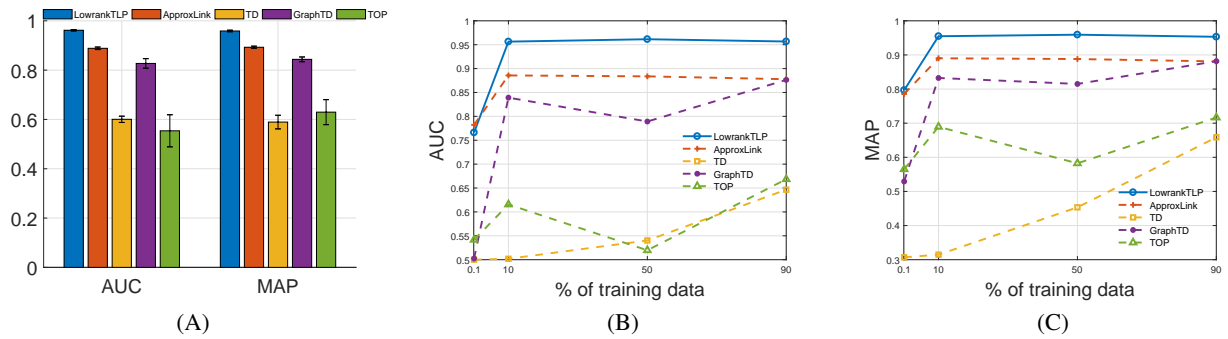
We further compared our MATLAB LowrankTLP implementation using Tensor Toolbox [2] version 2.6 with a parallel shared-memory version developed in C with SPLATT library [18], version 1.1.1 for sparse tensor operations in Figure 3 (C). The parallelization by SPLATT results in speedups of over an order of magnitude compared to the Tensor Toolbox. This parallel version is able to align 100 graphs of size 1000 each in 148s while the MATLAB version is not able to perform the alignment using a server with Intel(R) Xeon(R) CPU E5-2450 with 32 cores 2.10GHz, 2 CPUs and 196GB of RAM.

7.2 Predicting relations in scientific publications

We downloaded the DBLP dataset of scientific publication records from AMiner (Extraction and Mining of Academic Social Networks) [21]. The dataset contains 2,092,356 papers, 8,024,869 citations, 1,712,433 authors and 4,258,946 collaborations between authors, and a total of 264,025 distinct venues in which the papers were published. We built three graphs: Authors × Author ($W^{(1)}$), Paper × Paper ($W^{(2)}$) and Venue × Venue ($W^{(3)}$). In $W^{(1)}$ the edge weight is the count of papers that both authors have co-authored; in $W^{(2)}$, the edge weight is the number of times both papers were cited by another paper; and in $W^{(3)}$, the edge weight is calculated using

(A)　　　　　　　　　　　　　　　(B)　　　　　　　　　　　　　　　(C)

Figure 3: Simulation results. (A) & (B) Each experiment was repeated five times and the average is shown. (C) Runtime of Lowrank-TLP. The rank $k = \frac{n*10^4}{5}$ is chosen by the empirical results to achieve AUC ≈ 0.9.

(A)　　　　　　　　　　　　　　　(B)　　　　　　　　　　　　　　　(C)

Figure 4: DBLP results. (A) The performance of 5-fold cross-validation. The average and standard deviation across the 5 folds are shown. (B) & (C) The performance of using various percentages of training data.

Jaccard similarity between the vectors of the two venues whose dimensions are a bag of citations. After removing the vertices with low degree in each graph, we finally obtain $W^{(1)}$ with 13,823 vertices and 266,222 edges; $W^{(2)}$ with 11,372 vertices and 4,309,772 edges; and $W^{(3)}$ with 10,167 vertices and 46,557,116 edges. We also built 12,066 triples in the form (Paper, Author, Venue) as positive multi-relations, given by the natural relationship that a paper is written by an author, and published in a specific venue. These triples are stored in a sparse Tensor as input.

We performed 5-fold cross-validation to select the parameters for LowrankTLP and the baselines with the 12,066 positive triples together with the same number of randomly sampled triples as negative samples. Figure 4 (A) shows that LowrankTLP outperforms the baselines in every fold. We also randomly sampled 0.1%, 10%, 50% and 90% of positive triples as training data to test the rest of the positive tuples together with the same number of randomly sampled negative triples. Using the optimal parameters chosen from the previous 5-fold cross-validation, we tested the performance of all the methods on various percentages of training/test data. Figure 4 (B) & (C) show that LowrankTLP clearly outperforms all the baselines. Interestingly, both LowrankTLP and ApproxLink can achieve AUC ≈ 0.76 and MAP ≈ 0.8 when there are only 0.1% of training data while the prediction of the tensor-completion methods TD and GraphTD are nearly random, which further confirms that label propagation based methods are more stable to sparse input than tensor-completion methods.
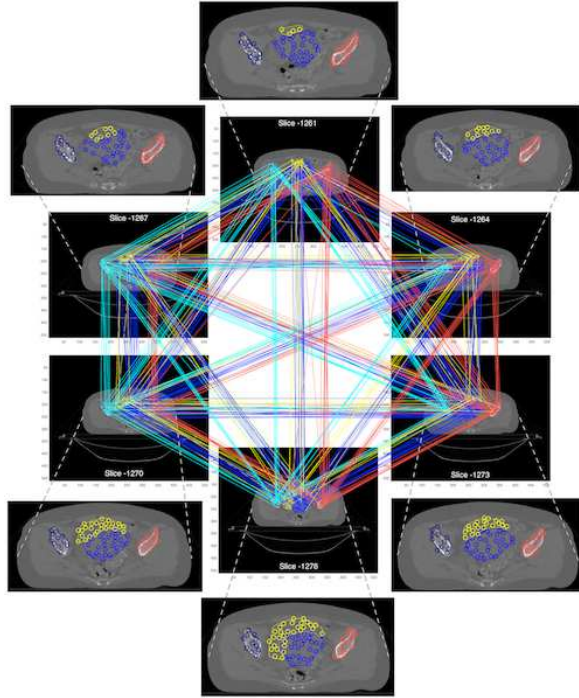
7.3 Alignment of CT Scan images

We obtained a dataset of 134 CT Scan images of an anonymized female patient. The scans were acquired on a Philips Brilliance Big Bore CT Scanner, and each slice is 512×512 pixels with a slice thickness of 3mm. We used a subset of 26 slices which contain the same set of four segmented regions (features). We built a graph for each slice by sampling from each region a number of circles where the number is proportional to the size of the region. To build a graph for each image, we calculated the similarity between the circles' spots using the following function: $s(x_i, x_j) = e^{-\frac{||x_i - x_j||_2^2}{\sigma}}$ if $\phi(x_i) \neq \phi(x_j)$ and otherwise 1, where x_i and x_j are the coordinates of the two circles, $\phi(x)$ represents the region where the circle x is located and $\sigma = 10$ is the width of RBF function. The pairwise similarity between the circles in two different images was obtained by the color density difference between the spots, calculated using a RBF function with $\sigma = 10^{-2}$. The initial tensor $\mathcal{Y}_0$ was then generated in CP-form using these cross-images spots similarity matrices. For example, to align 7 images, $\binom{7}{2} = 21$ pairwise matrices are computed.

Our objective is to align the sampled spots in the same features across the images. A set of candidate tuples of spots across the graphs were picked as the multi-relations to be learned. The candidate tuples of spots were selected if the color densities between each pair of the spots in the tuple are all above a threshold. The alignment accuracy was measured by the top-1 match of each spot.

	LowrankTLP				TD	GraphTD
	rank 10	rank 100	rank 1000	rank 10000		
4 images	0.59	0.91	0.91	0.91	0.61	0.73
5 images	0.67	0.80	0.89	0.89	0.66	0.75
6 images	0.78	0.78	0.84	0.84	0.69	0.78
7 images	0.75	0.73	0.80	0.83	0.72	0.76

Table 1: Performance of image alignment. The accuracy is measured as the % of accurately aligned spots in the first image.
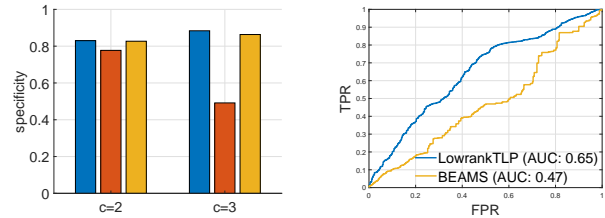
Figure 5: Results of CT Scan image alignment. Each segmented region in the images is represented by a different color in the alignment.

Specifically, for each spot in the first graph, we take the subtensor of dimension $(n-1)$ associated with the entry in the first dimension to find the entry of the highest score in the subtensor. Then, we check if the features of the aligned spots from all the other images in the maximum entry are the same as the spot in the first dimension.
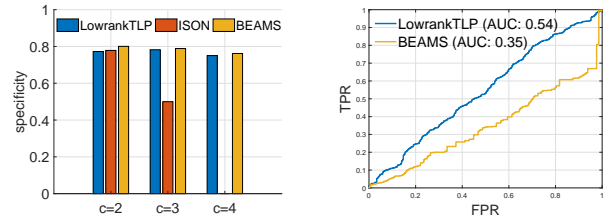
Table (1) shows the results using the evaluation. LowrankTLP was compared with TD and GraphTD. Note that ApproxLink and TOP cannot take pairwise similarities as input and are thus inapplicable to this dataset. LowrankTLP achieved much higher accuracy than the baselines TD and GraphTD for rank ≥ 100 in almost all the cases. It is also interesting that, with a rank of only 10,000, LowrankTLP was able to align 7 images with an accuracy of 0.83 meaning 83% of the spots in the first graph was perfectly matched with a spot of the same feature in all other six images. An example of six aligned images is shown in Figure 5. It is clear that the aligned spots are consistent across the images.

7.4 Alignment of PPI Networks

We downloaded the IsoBase dataset [10, 15, 17], containing protein-protein interactions (PPI) networks for five species: *H. sapiens* (HS), *D. melanogaster* (DM), *S. cerevisiae* (SC), *C. elegans* (CE) and *M.*

(A) Alignment of HS/DM/SC networks

(B) Alignment of HS/DM/SC/CE networks

Figure 6: Results of PPI network alignment. In both (A) and (B), the figure on the left shows the specificity of the detected clusters containing different number of species, and the figure on the right shows the AUC curves between true positives and false positives in the predicted candidate entries only.

musculus (MM). The *M. musculus* network only contains 776 interactions and was dropped from the analysis. After removing proteins with no association in the PPI networks, there are 10,403, 7,396, 5,524 and 2,995 proteins and 109,822, 49,991, 165,588 and 9,711 interactions in the HS, DM, SC and CE PPI networks, respectively.

The dataset also contains cross-species protein sequence similarity as BLAST Bit-values for all the pairs of species. Similar to the CT Scan experiment, we generated the input tensor $\mathcal{Y}_0$ in CP-form by using the pairwise sequence similarity of all pairs of the species. This input option is very suitable for PPI network alignment since the pairwise matrices can be naturally obtained using BLAST. In addition, the annotations of the proteins with 37,463 gene ontology (GO) terms below level five of GO are also provided. For the evaluation, we generated candidate tuples of proteins with high sequence similarity between all pairs of proteins in the tuple. These candidate tuples can then be classified as true multi-relations if all the annotated proteins in the tuple share at one common GO term, and otherwise false multi-relations. The experiments were performed using three species (HS, DM and SC) and four species by adding CE. Around 3M tuples were generated among three species and about 163M among four species.

Similar to the post-processing in BEAMS [1], after applying LowrankTLP to generate the prediction scores for all the candidate tuples, the tuples are sorted for a greedy merge as protein clusters for standard evaluation of PPI network alignment. A cluster of size n is defined as a set of proteins with at least one protein from each of the n species. The greedy merge scans the tuples and adds the tuple that only contains proteins not seen yet as a new cluster. Otherwise, the proteins that are already in some other clusters are removed from the tuple, and the remaining proteins are added as a smaller cluster.

In the evaluation, the specificity is defined as the ratio between the number of consistent clusters and the number of annotated clusters, where an annotated cluster is a cluster in which at least two proteins are associated to a GO term, and a consistent cluster is the one in which all of its annotated proteins share at least one GO term. In the left plot in Figure 6 (A), for both clusters of size 2 and 3, LowrankTLP performs better than both BEAMS and IsorankN in the alignment of three networks. The left plot in Figure 6 (B) shows that LowrankTLP performed similarly or slightly worse than BEAMS in every cluster size in the alignment of four networks. IsorankN was not able to detect any cluster of size 4.

To further compare LowrankTLP with BEAMS, we compared the detailed ranking of the annotated clusters reported by BEAMS. We classified these clusters as positive multi-relations or false multi-relations, and report the AUC by their rankings in the right plots in Figure 6. In both three-network alignment and four-network alignment, LowrankTLP ranked the true multi-relations above the false multi-relations with AUC larger than 0.5. Note that since we only check the very top of the predictions (those predicted as true multi-relations), the AUC is less than 0.5 for BEAMs results.

8 CONCLUSION

In this study, we introduced a new algorithm LowrankTLP to improve the scalability and performance of label propagation on tensor product graphs for multi-relational learning. We demonstrated that LowrankTLP well approximates label propagation on the full tensor product graph to achieve both the better scalability and performance. We also demonstrated that LowrankTLP, capable of taking either a sparse tensor or a CP-form tensor as input, is a flexible approach to meet the requirements of multi-relational learning problems in a wide range of applications. In all the experiments, we observed that it does not require a huge rank to achieve a good prediction performance even if the size of a tensor product graph is exponential of the size of the individual graphs. This observation supports that the direct and efficient analysis of the entire spectral of the tensor product graph is a better approach. In the future, we will analyze the spectral of the tensor product graphs to develop a strategy of estimating a good rank k for better application of LowrankTLP to multi-relational learning.

A USEFUL LEMMAS

LEMMA A.1. *If A, B, C and D are matrices of such size that one can form the matrix products AC and BD, then $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$.*

LEMMA A.2. *If matrices A, B, C and D are of such size that one can form the operation $(A \odot B), (C \odot D), (A^T C)$ and $(B^T D)$, then equality $(A \odot B)^T (C \odot D) = (A^T C) * (B^T D)$ holds.*

LEMMA A.3. *Given two matrices A and B, the equation $(A \otimes B)^T = A^T \otimes B^T$ holds.*

LEMMA A.4. *Let $\lambda_1, \ldots, \lambda_n$ be eigenvalues of A with corresponding eigenvectors $x_1, \ldots, x_n$, and let $\mu_1, \ldots, \mu_m$ be eigenvalues of B with corresponding eigenvectors $y_1, \ldots, y_m$. Then the eigenvalues and eigenvectors of $A \otimes B$ are $\lambda_i \mu_j$ and $x_i \otimes y_j$, $i = 1, \ldots, n, j = 1, \ldots, m$.*

REFERENCES

[1] Ferhat Alkan and Cesim Erten. 2013. BEAMS: backbone extraction and merge strategy for the global many-to-many alignment of multiple PPI networks. *Bioinformatics* (2013).

[2] Brett W. Bader, Tamara G. Kolda, et al. 2015. MATLAB Tensor Toolbox Version 2.6. Available online. (February 2015). http://www.sandia.gov/~tgkolda/TensorToolbox/

[3] Xiaofei He Jiawei Han Cai, Deng and Thomas S. Huang. 2011. Graph regularized nonnegative matrix factorization for data representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2011), 1548–1560.

[4] Xiaofei He Xiaoyun Wu Cai, Deng and Jiawei Han. 2008. Non-negative matrix factorization on manifold.. In *ICDM*. 63–72.

[5] Chris Ding, Xiaofeng He, and Horst D Simon. 2005. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM, 606–610.

[6] Olivier Duchenne, Francis Bach, In-So Kweon, and Jean Ponce. 2011. A tensor-based algorithm for high-order graph matching. *IEEE transactions on pattern analysis and machine intelligence* 33, 12 (2011), 2383–2395.

[7] Carl Eckart and Gale Young. 1936. The approximation of one matrix by another of lower rank. *Psychometrika* 1, 3 (1936), 211–218.

[8] Nicholas J Higham. 2002. *Accuracy and stability of numerical algorithms*. SIAM.

[9] Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review* 51, 3 (2009), 455–500.

[10] Chung-Shou Liao, Kanghao Lu, Michael Baym, Rohit Singh, and Bonnie Berger. 2009. IsoRankN: spectral methods for global alignment of multiple protein networks. *Bioinformatics* 25, 12 (2009), i253–i258.

[11] Hanxiao Liu and Yiming Yang. 2015. Bipartite Edge Prediction via Transductive Learning over Product Graphs.. In *ICML*. 1880–1888.

[12] Hanxiao Liu and Yiming Yang. 2016. Cross-Graph Learning of Multi-Relational Associations. In *ICML*. 2235–2243.

[13] Kohei Hayashi Ryota Tomioka Narita, Atsuhiro and Hisashi Kashima. 2011. Tensor factorization using auxiliary information.. In *ECML-PKDD*.

[14] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *IEEE proceeding* (2015).

[15] Daniel Park, Rohit Singh, Michael Baym, Chung-Shou Liao, and Bonnie Berger. 2010. IsoBase: a database of functionally related proteins across PPI networks. *Nucleic acids research* 39, suppl_1 (2010), D295–D300.

[16] Rudy Raymond and Hisashi Kashima. 2010. Fast and scalable algorithms for semi-supervised link prediction on static and dynamic graphs. *Machine Learning and Knowledge Discovery in Databases* (2010), 131–147.

[17] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2008. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences* 105, 35 (2008), 12763–12768.

[18] Shaden Smith and George Karypis. 2016. SPLATT: The Surprisingly ParalleL spArse Tensor Toolkit. http://cs.umn.edu/~splatt/. (2016).

[19] Shaden Smith, Niranjay Ravindran, Nicholas D Sidiropoulos, and George Karypis. 2015. SPLATT: Efficient and parallel sparse tensor-matrix multiplication. *29th IEEE International Parallel & Distributed Processing Symposium* (2015).

[20] Martin Szummer and Tommi Jaakkola. 2001. Partially labeled classification with Markov random walks. In *NIPS*, Vol. 14.

[21] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 990–998.

[22] Maoqiang Xie, Taehyun Hwang, and Rui Kuang. 2012. Prioritizing disease genes by bi-random walk. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 292–303.

[23] Ruochen Xu, Yiming Yang, Hanxiao Liu, and Andrew Hsi. 2016. Cross-lingual Text Classification via Model Translation with Limited Dictionaries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 95–104.

[24] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with local and global consistency.. In *NIPS*, Vol. 16. 321–328.

[25] Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. (2002).