TENALIGN: Joint Tensor Alignment and Coupled Factorization

Yunshu Wu*, Uday Singh Saini*, Jia Chen† and Evangelos E. Papalexakis*
*Computer Science and Engineering, †Electrical and Computer Engineering
University of California Riverside
Email: ywu380@ucr.edu, usain001@ucr.edu, jiac@ucr.edu, epapalex@cs.ucr.edu

***Abstract*—Multimodal datasets represented as tensors oftentimes share some of their modes. However, even though there may exist a one-to-one (or perhaps partial) correspondence between the coupled modes, such correspondence/alignment may not be given, especially when integrating datasets from disparate sources. This is a very important problem, broadly termed as entity alignment or matching, and subsets of the problem such as graph matching have been extremely popular in the recent years. In order to solve this problem, current work computes the alignment based on existing embeddings of the data. This can be problematic if our end goal is the joint analysis of the two datasets into the same latent factor space: the embeddings computed separately per dataset may yield a suboptimal alignment, and if such an alignment is used to subsequently compute the joint latent factors, the computation will similarly be plagued by compounding errors incurred by the imperfect alignment. In this work, we are the first to define and solve the problem of joint tensor alignment and factorization into a shared latent space. By posing this as a unified problem and solving for both tasks simultaneously, we observe that the both alignment and factorization tasks benefit each other resulting in superior performance compared to two-stage approaches. We extensively evaluate our proposed method TENALIGN and conduct a thorough sensitivity and ablation analysis. We demonstrate that TENALIGN significantly outperforms baseline approaches where embedding and matching happen separately.**

***Index Terms*—Tensors, Alignment, Tensor decomposition**

I. INTRODUCTION

Tensors are a natural and powerful way to represent different modalities of data inherently capturing rich and meaningful information [1], [2]. Building upon tensor algebra, tensor decomposition has admitted elegant theoretical analysis and has been applied in a broad spectrum of real-world scenarios such as speech signal processing [3], biomedical data analytics [4], fake news detection [5], and channel estimation in wireless communication systems [6], to name a few.

Oftentimes, different datasets share one or more modes resulting in matrices/tensors with so-called *coupled* modes, i.e., there is a one-to-one correspondence between the entities of that coupled mode across datasets. For example, consider two different datasets: one that records movie viewing history for users over time, resulting in a (user, movie, time) tensor, and another one which captures user social media activity, resulting for instance in a (user, hashtag) tensor. In this example, the "user" mode is coupled. We can jointly factorize such coupled datasets [7] by essentially forcing the latent factors that correspond to the coupled mode to be shared across factorizations, and doing so enables the joint analysis of all datasets and entities involved. Beyond this motivating example, such coupled factorizations, with a popular example of Coupled Matrix Tensor Factorization (CMTF) has been widely and successfully applied in a number of different real-world problems [7], [8], [9].

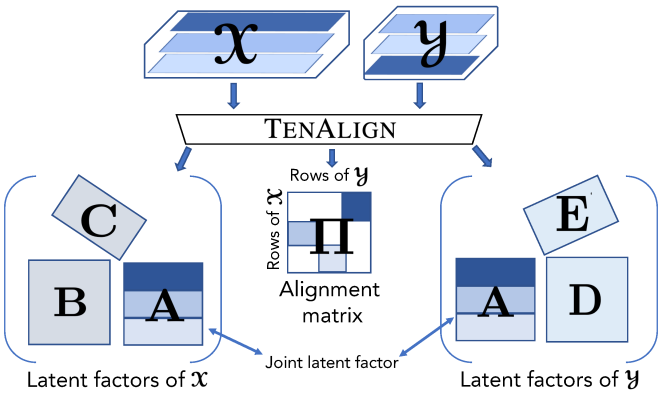


Fig. 1: Overview of the proposed method TENALIGN for joint tensor alingment and coupled factorization.

Virtually all existing works in the coupled factorization and literature [7], [10], [11], assume that the correspondence between indices in the coupled mode, whether complete or partial, is *known* and has been resolved prior to the factorization. However, when datasets are being integrated from disparate sources, it is very likely that such index correspondence is *unknown*, and one has to first *align* those indices, i.e., identify the right correspondence of coupled indices across the datasets of interest. Recently, a variant of CMTF was proposed which allows for *known* linear couplings across factors [12], and if we have knowledge of the alignment, we can encode it as a permutation matrix $\mathbf{\Pi}$ and perform the factorization by assuming that the two coupled factors are related through this linear transformation. However, if such an alignment matrix is imperfect, the quality of the computed factors is going to be low, since the error incurred by the imperfect transformation will compound as the algorithm iterates to a solution.

In order to learn such an alignment across indices, we would have to perform some form of entity alignment or matching [13], [14]. In the literature, there is a particularly prolific area of research which focuses on a specific instance of this

problem, where the two datasets of interest are graphs and the alignment pertains to the nodes across those graphs [15], [16], [17], [18], [19], [20], [21], [22]. In general, the existing state-of-the-art graph matching frameworks take two separate steps: (1) learning entity embeddings, and (2) searching for the entity matching between two sets of learnt embeddings. For example, REGAL [22] learns embeddings by xNetMF in the first step, which makes use of both attribute and structure information of entities, and then in the second step, conducts soft matching. We will be using REGAL as our running example graph matching in this paper, because of the fact that its embedding stage is the closest conceptually to our envisioned factorization, however, our proposed method can benefit from any current or future advances in embedding alignment, since, as we demonstrate, we can absorb such methods within our proposed method.

Currently, if we wish to jointly analyze two datasets which require alignment, we would have to first compute separate embeddings per dataset (e.g., single tensor factorizations for each given tensor) and then perform the alignment as a subsequent step, much like state-of-the-art graph alignment works [22]. However, doing so, suffers from a number of significant drawbacks: (a) The aligned embeddings will not going to be in the same latent factor space since they have been computed independently, and (b) the quality of the alignment will be suboptimal, especially if the two datasets share their latent structure. It is important to note here that drawback (a) could potentially be addressed by conducting a coupled factorization post-alignment, however, as we mention above and demonstrate in the experiments, because of the lower-quality computed alignment, this coupled factorization will not be able to recover the latent factors accurately.

In this paper, we jointly tackle the two tasks of alignment and coupled factorization. To the best of our knowledge, this work is the first to do so, and as we demonstrate in our experimental evaluation, solving those two tasks jointly is highly beneficial for both tasks, and substantially outperforms approaches where embedding and alignment happen separately in sequence. Our contributions are summarized as follows:

- **Novel Problem**: We are the first to formulate the problem of joint tensor alignment and factorization, and in this paper we explore two novel expressions of this problem and we discuss the trade-offs between the two formulations.
- **Flexible Algorithms**: We introduce TENALIGN which solves the two proposed formulations by deriving custom optimization algorithms. Furthermore, TENALIGN is *extremely flexible and can subsume existing and future advances in computing alignment matrices as parts of the optimization procedure*. We demonstrate such flexibility by leveraging REGAL's alignment solver [22] within TENALIGN as an example.
- **Extensive Experiments**: We perform extensive experimental evaluation of our proposed method using synthetic and real data and comparing against baselines where the two tasks are solved independently. Furthermore, we conduct detailed sensitivity analysis and ablation study.
- **Reproducibility**: We provide detailed derivations and descriptions for our proposed algorithms and our experimental setup, and we demonstrate our results on public data. Furthermore, we make our implementation publicly available upon publication at https://github.com/yunshuwu/TenAlign in order to promote reproducibility and extensions of our work.

II. PROBLEM FORMULATION

We provide the necessary notation and definitions for our work. Table I provides a summary of the notation used.

Notation	Description
$\mathcal{X}$, $\mathbf{X}$, $\mathbf{x}$, x	Tensor, Matrix, vector, scalar
$\|\mathbf{X}\|_F$	Frobenius norm of matrix $\mathbf{X}$
$\|\mathbf{x}\|_p$	ℓ_p-norm of vector $\mathbf{x}$
$\mathbf{A}(i,:)$	the i-th row of matrix $\mathbf{A}$
vec()	Vectorization operator
$\circ$	Outer product
$\odot$	Khatri-Rao product (column-wise Kronecker product)
$*$	Element-wise multiplication
$\times_n$	n-mode product
$\mathbf{X}_{(n)}$	n-mode matricization of tensor $\mathcal{X}$
$\mathbf{X}^{-1}$	Inverse of matrix $\mathbf{X}$
$\mathbf{X}^\intercal$	Transpose of matrix $\mathbf{X}$
$H(\mathbf{p})$	Entropy of probability distribution $\mathbf{p}$

TABLE I: Notations used in matrix and tensor algebra.

Tensors are higher-order extensions of matrices, and they are a natural way to express multimodal real world data. The Canonical Polyadic Decomposition (CPD), also named as PARAFAC or CANDECOMP [23], one of the most commonly used tensor methods, models a tensor $\mathcal{T}$ as the sum of R outer products where R is a pre-defined decomposition rank. Taking a three-mode tensor as an example (higher order CPD can be readily generalized), CPD minimizes the distance between the input tensor $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$ and the approximation of it in the squared norm, i.e., $\mathcal{T} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ where $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, and $\mathbf{c}_r \in \mathbb{R}^K$ are the r-th column of the factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, respectively.

A matrix and a tensor can be multiplied together using the n-mode product, denoted by $\mathcal{X} \times_1 \mathbf{M}$ as an example of a 1-mode product, where the slices along the n-th mode of the tensor are multiplied by matrix $\mathbf{M}$ resulting in slices of a new tensor. In the interest of space, we refer the reader to existing comprehensive surveys [1], [2] for a detailed overview of notation and definitions.

Problem Introduction Consider two tensor datasets $\mathcal{X}$ and $\mathcal{Y}$, where for instance $\mathcal{X}$ captures (user,video,time) YouTube data and $\mathcal{Y}$ captures (user,product,time) Amazon data. Further assume that there is a complete matching $\mathbf{\Pi}$ (for current problem it is assumed to be complete and we defer partial matching for future work) which is unknown to us. In optimization terms, we can express the joint alignment and coupled factorization below, by jointly learning the mapping $\mathbf{\Pi}$ and the latent factors, while forcing the rows of $\mathcal{X}$ and $\mathcal{Y}$ to be expressed by the same latent factors:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D},\mathbf{E},\mathbf{\Pi}} \|\mathcal{X} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_F^2 + \|\mathcal{Y} \times_1 \mathbf{\Pi} - [\![\mathbf{A}, \mathbf{D}, \mathbf{E}]\!]\|_F^2 \tag{1}$$

where $\mathbf{\Pi}$ is a permutation matrix and $\mathcal{Y} \times_1 \mathbf{\Pi}$ effectively permutes the data pertaining to each row of the tensor. The above formulation simultaneously learns a joint embedding (where $\mathbf{A}$ embeds the rows to a latent space) while enforcing that transformation on the data. We assume that both tensors $\mathcal{X}$ and $\mathcal{Y}$ admit a joint CPD structure, which allows us to consider that $\mathbf{\Pi}$ when multiplying the tensor in the first mode equivalently only multiplies the factor of that mode [2].

What does the ideal $\mathbf{\Pi}$ look like? In order to define precisely our problem, we should describe what the ideal alignment matrix $\mathbf{\Pi}$ should look like. Strictly speaking, $\mathbf{\Pi}$, is a permutation matrix, which is a square matrix whose each row and column have exactly one "1" and their rest of their entries equal to "0". Every "1" in that matrix is essentially capturing the corresponding of the i-th row of tensor $\mathcal{X}$ and the j-th row of tensor $\mathcal{Y}$.

The permutation matrix $\mathbf{\Pi}$ has the following properties:

- $\mathbf{\Pi}$ is an orthogonal matrix, i.e., $\mathbf{\Pi}^\intercal = \mathbf{\Pi}^{-1}$.
- $\mathbf{\Pi}_{ij} \in \{0, 1\}, \forall i \in I, j \in J$, element is either 0 or 1

To identify the constraints for $\mathbf{\Pi}$, we have:

Proposition 2.1: Given an orthogonal matrix $\mathbf{\Pi}$ with all elements non-negative, then $\mathbf{\Pi}$ is a permutation matrix.

Proof. Let us prove this by contradiction. Assume each row has more than one positive elements, by pigeonhole principle, there must be one component (column) which is positive at least in two columns. Let this component be $\mathbf{a}$, where at least $\mathbf{a}_i > 0$ and $\mathbf{a}_j > 0$. By the assumption, there will be another component $\mathbf{b}$ which is positive at row i, then the dot product of $\mathbf{a}$ and $\mathbf{b}$ is not going to be zero:

$$\mathbf{a} \cdot \mathbf{b} \geq \mathbf{a}_i \mathbf{b}_i > 0$$

which contradicts the orthogonality. ■

Thus, we can form the alignment matrix $\mathbf{\Pi}$ to be an orthogonal matrix with all elements being non-negative decimals.

III. Proposed Method

Factorizing tensors and solving directly for a permutation matrix $\mathbf{\Pi}$ given the $\mathbf{\Pi}$ structure are computationally very hard, therefore we investigate two linear programming relaxations of the optimization problem.

Proposed relaxations Here for each of these two relaxations we have a separate formulation, where $\mathcal{L}_1$ is a harder constrained problem which forces factors into a same space:

$$\mathcal{L}_1 = \|\mathcal{X} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_F^2 + \|\mathcal{Y} \times_1 \mathbf{\Pi} - [\![\mathbf{A}, \mathbf{D}, \mathbf{E}]\!]\|_F^2 \quad (2)$$

and $\mathcal{L}_2$ is a relaxed problem which softly makes factors of tensors $\mathcal{X}$ and $\mathcal{Y}$ to share some common components, meaning that the space of two tensors are intersect but not the same:

$$\begin{aligned}\mathcal{L}_2 = \|\mathcal{X} - [\![\mathbf{A}_\mathbf{x}, \mathbf{B}, \mathbf{C}]\!]\|_F^2 + \|\mathcal{Y} - [\![\mathbf{A}_\mathbf{y}, \mathbf{D}, \mathbf{E}]\!]\|_F^2 \\ +\lambda \|\mathbf{A}_\mathbf{x} - \mathbf{\Pi}\mathbf{A}_\mathbf{y}\|_F^2\end{aligned} \quad (3)$$

For both relaxations, the permutation matrix $\mathbf{\Pi}$ should satisfy the following constraints:

$\mathbf{\Pi}_{ij} \in \{0, 1\}, \forall i, j$, and $\mathbf{\Pi} \cdot \mathbf{1} = \mathbf{1}$ (row sum equal to 1), and $\mathbf{1}^\intercal \cdot \mathbf{\Pi} = \mathbf{1}^\intercal$ (column sum equal to 1), and $\mathbf{\Pi}\mathbf{\Pi}^\intercal = \mathbf{\Pi}^\intercal\mathbf{\Pi} = \mathbf{I}$ (orthogonality constraint), and $H(\mathbf{\Pi}(i, :)) < b, \forall j$ (row-wise entropy constraint)

By bounding the entropy of each row of $\mathbf{\Pi}$. This way we are forcing the entropy of each row to be small which, if we view each row as a probability distribution that describes the most likely alignments, we would like that distribution to be as far from uniform as possible and as close to a deterministic distribution as possible (with a single 1 in that row).

Trade-offs between formulations $\mathcal{L}_1$ and $\mathcal{L}_2$ Conceptually, formulation $\mathcal{L}_1$ is defining a harder optimization problem because the permutation matrix $\mathbf{\Pi}$ is directly affecting tensor $\mathcal{Y}$, compared to $\mathcal{L}_2$ where the permutation matrix is softly enforced as a regularization term.

In terms of the type of latent factors that each formulation learns, conceptually, $\mathcal{L}_1$ is directly enforcing the coupling between $\mathcal{X}$ and $\mathcal{Y}$ by forcing them to be expressed by the same factor matrix $\mathbf{A}$, which is how typically CMTF-style approaches are expressed. On the other hand, $\mathcal{L}_2$ is learning two different factor matrices $\mathbf{A}_\mathbf{x}$ and $\mathbf{A}_\mathbf{y}$ which are softly required to be similar under the alignment transformation. This implies that $\mathcal{L}_2$ may not always give us the exact same set of latent factors, thereby violating the typical CMTF modeling.

However, regarding this last remark, there are cases where the two tensors may share part of their latent factors [11] (in fact, the chemical datasets analyzed in our experiments fall under this category), and this formulation may be able to more flexibly capture shared and unshared latent factors.

To solve these two optimization problems, we compute the gradient and let it be zero to update the corresponding factor matrix. Next, let's first discuss the calculations of the gradient for the first formulation $\mathcal{L}_1$ and then followed by the calculations of the gradient for the second formulation $\mathcal{L}_2$.

A. Solution for TenAlign-$\mathcal{L}_1$

For formula $\mathcal{L}_1$, We can rewrite it by two sub-functions f_{11} and f_{12}:

$$\mathcal{L}_1 = \underbrace{\|\mathcal{X} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]\|_F^2}_{f_{11}(\mathbf{A},\mathbf{B},\mathbf{C})} + \underbrace{\|\mathcal{Y} \times_1 \mathbf{\Pi} - [\![\mathbf{A}, \mathbf{D}, \mathbf{E}]\!]\|_F^2}_{f_{12}(\mathbf{A},\mathbf{D},\mathbf{E},\mathbf{\Pi})} \quad (4)$$

Let $\mathbf{Y}'_{(1)}$, $\mathbf{Y}'_{(2)}$ and $\mathbf{Y}'_{(3)}$ to be the one-mode matricization, two-mode matricization and three-mode matricization of $\mathcal{Y} \times_1 \mathbf{\Pi}$, respectively.

The first order partial derivative of $\mathcal{L}_1$ respect to each factor matrix is as follows:

1) For partial derivative with respect to A:

$$\begin{aligned}\frac{\partial \mathcal{L}_1}{\partial \mathbf{A}} &= \frac{\partial f_{11}}{\partial \mathbf{A}} + \frac{\partial f_{12}}{\partial \mathbf{A}} \\ &= \frac{\partial}{\partial \mathbf{A}} \|\mathbf{X}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^\intercal\|_F^2 + \frac{\partial}{\partial \mathbf{A}} \|\mathbf{Y}'_{(1)} - \mathbf{A}(\mathbf{E} \odot \mathbf{D})^\intercal\|_F^2 \\ &= \frac{\partial}{\partial \mathbf{A}} \operatorname{Tr}\left((\mathbf{X}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^\intercal)(\mathbf{X}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^\intercal)^\intercal\right) \\ &+ \frac{\partial}{\partial \mathbf{A}} \operatorname{Tr}\left((\mathbf{Y}'_{(1)} - \mathbf{A}(\mathbf{E} \odot \mathbf{D})^\intercal)(\mathbf{Y}'_{(1)} - \mathbf{A}(\mathbf{E} \odot \mathbf{D})^\intercal)^\intercal\right) \\ &= -2\mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B}) + 2\mathbf{A}(\mathbf{C}^\intercal\mathbf{C} * \mathbf{B}^\intercal\mathbf{B}) \\ &\quad -2\mathbf{Y}'_{(1)}(\mathbf{E} \odot \mathbf{D}) + 2\mathbf{A}(\mathbf{E}^\intercal\mathbf{E} * \mathbf{D}^\intercal\mathbf{D})\end{aligned}$$

Where in the first equality above, we obtain this representation for the 1-mode matricization of the two tensors according to the CPD model [2].

Let $\frac{\partial \mathcal{L}_1}{\partial \mathbf{A}}$ be zero, then we have the update for matrix $\mathbf{A}$:

$$\mathbf{A} = (\mathbf{X}_{(1)}(\mathbf{C}\odot\mathbf{B})+\mathbf{Y}'_{(1)}(\mathbf{E}\odot\mathbf{D}))\cdot(\mathbf{C}^\intercal\mathbf{C}*\mathbf{B}^\intercal\mathbf{B}+\mathbf{E}^\intercal\mathbf{E}*\mathbf{D}^\intercal\mathbf{D})^{-1}$$

2) For partial derivative with respect to B:

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{B}} = \frac{\partial f_{11}}{\partial \mathbf{B}} = \frac{\partial}{\partial \mathbf{B}} \left\|\mathbf{X}_{(2)} - \mathbf{B}(\mathbf{C}\odot\mathbf{A})^\intercal\right\|_F^2$$
$$= \frac{\partial}{\partial \mathbf{B}} \operatorname{Tr}\left((\mathbf{X}_{(2)} - \mathbf{B}(\mathbf{C}\odot\mathbf{A})^\intercal)(\mathbf{X}_{(2)} - \mathbf{B}(\mathbf{C}\odot\mathbf{A})^\intercal)^\intercal\right)$$
$$= -2\mathbf{X}_{(2)}(\mathbf{C}\odot\mathbf{A}) + 2\mathbf{B}(\mathbf{C}^\intercal\mathbf{C} * \mathbf{A}^\intercal\mathbf{A})$$

Let $\frac{\partial \mathcal{L}_1}{\partial \mathbf{B}}$ be zero, then we have the update rule for matrix $\mathbf{B}$:

$$\mathbf{B} = \mathbf{X}_{(2)}(\mathbf{C}\odot\mathbf{A})\cdot(\mathbf{C}^\intercal\mathbf{C} * \mathbf{A}^\intercal\mathbf{A})^{-1}$$

3) For partial derivative with respect to C:

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{C}} = \frac{\partial f_{11}}{\partial \mathbf{C}} = \frac{\partial}{\partial \mathbf{C}} \left\|\mathbf{X}_{(3)} - \mathbf{C}(\mathbf{B}\odot\mathbf{A})^\intercal\right\|_F^2$$
$$= \frac{\partial}{\partial \mathbf{C}} \operatorname{Tr}\left((\mathbf{X}_{(3)} - \mathbf{C}(\mathbf{B}\odot\mathbf{A})^\intercal)(\mathbf{X}_{(3)} - \mathbf{C}(\mathbf{B}\odot\mathbf{A})^\intercal)^\intercal\right)$$
$$= -2\mathbf{X}_{(3)}(\mathbf{B}\odot\mathbf{A}) + 2\mathbf{C}(\mathbf{B}^\intercal\mathbf{B} * \mathbf{A}^\intercal\mathbf{A})$$

Let $\frac{\partial \mathcal{L}_1}{\partial \mathbf{C}}$ be zero, then we have the update rule for matrix $\mathbf{C}$:

$$\mathbf{C} = \mathbf{X}_{(3)}(\mathbf{B}\odot\mathbf{A})\cdot(\mathbf{B}^\intercal\mathbf{B} * \mathbf{A}^\intercal\mathbf{A})^{-1}$$

4) For partial derivative with respect to D:

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{D}} = \frac{\partial f_{12}}{\partial \mathbf{D}} = \frac{\partial}{\partial \mathbf{D}} \|\mathbf{Y}'_{(2)} - \mathbf{D}(\mathbf{E}\odot\mathbf{A})^\intercal\|_F^2$$
$$= \frac{\partial}{\partial \mathbf{D}} \operatorname{Tr}\left((\mathbf{Y}'_{(2)} - \mathbf{D}(\mathbf{E}\odot\mathbf{A})^\intercal)(\mathbf{Y}'_{(2)} - \mathbf{D}(\mathbf{E}\odot\mathbf{A})^\intercal)^\intercal\right)$$
$$= -2\mathbf{Y}'_{(2)}(\mathbf{E}\odot\mathbf{A}) + 2\mathbf{D}(\mathbf{E}^\intercal\mathbf{E} * \mathbf{A}^\intercal\mathbf{A})$$

Let $\frac{\partial \mathcal{L}_1}{\partial \mathbf{D}}$ be zero, then we have the update rule for matrix $\mathbf{D}$:

$$\mathbf{D} = \mathbf{Y}'_{(2)}(\mathbf{E}\odot\mathbf{A})\cdot(\mathbf{E}^\intercal\mathbf{E} * \mathbf{A}^\intercal\mathbf{A})^{-1}$$

5) For partial derivative with respect to E:

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{E}} = \frac{\partial f_{12}}{\partial \mathbf{E}} = \frac{\partial}{\partial \mathbf{E}} \|\mathbf{Y}'_{(3)} - \mathbf{E}(\mathbf{D}\odot\mathbf{A})^\intercal\|_F^2$$
$$= \frac{\partial}{\partial \mathbf{E}} \operatorname{Tr}\left((\mathbf{Y}'_{(3)} - \mathbf{E}(\mathbf{D}\odot\mathbf{A})^\intercal)(\mathbf{Y}'_{(3)} - \mathbf{E}(\mathbf{D}\odot\mathbf{A})^\intercal)^\intercal\right)$$
$$= -2\mathbf{Y}'_{(3)}(\mathbf{D}\odot\mathbf{A}) + 2\mathbf{E}(\mathbf{D}^\intercal\mathbf{D} * \mathbf{A}^\intercal\mathbf{A})$$

Let $\frac{\partial \mathcal{L}_1}{\partial \mathbf{E}}$ be zero, then we have the update rule for matrix $\mathbf{E}$:

$$\mathbf{E} = \mathbf{Y}'_{(3)}(\mathbf{D}\odot\mathbf{A})\cdot(\mathbf{D}^\intercal\mathbf{D} * \mathbf{A}^\intercal\mathbf{A})^{-1}$$

6) For partial derivative with respect to Π :

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{\Pi}} = \frac{\partial f_{12}}{\partial \mathbf{\Pi}} = \frac{\partial}{\partial \mathbf{\Pi}} \|\mathbf{\Pi}\mathbf{Y}_{(1)} - \mathbf{A}(\mathbf{E}\odot\mathbf{D})^\intercal\|_F^2$$
$$= \frac{\partial}{\partial \mathbf{\Pi}} \operatorname{Tr}\left((\mathbf{\Pi}\mathbf{Y}_{(1)} - \mathbf{A}(\mathbf{E}\odot\mathbf{D})^\intercal)(\mathbf{\Pi}\mathbf{Y}_{(1)} - \mathbf{A}(\mathbf{E}\odot\mathbf{D})^\intercal)^\intercal\right)$$
$$= 2\mathbf{\Pi}\cdot\mathbf{Y}_{(1)}\mathbf{Y}_{(1)}^\intercal - 2(\mathbf{Y}_{(1)}(\mathbf{E}\odot\mathbf{D})\mathbf{A}^\intercal)^\intercal$$

Then we need to solve $\frac{\partial \mathcal{L}_1}{\partial \mathbf{\Pi}} = 0$ for $\mathbf{\Pi}$ with the following constraints:

$$\begin{aligned} \mathbf{\Pi}\cdot\mathbf{Y}_{(1)}\mathbf{Y}_{(1)}^\intercal &= (\mathbf{Y}_{(1)}(\mathbf{E}\odot\mathbf{D})\mathbf{A}^\intercal)^\intercal \\ s.t.\ \mathbf{\Pi}\mathbf{1} = \mathbf{1},\ \mathbf{1}^\intercal\mathbf{\Pi} &= \mathbf{1}^\intercal, \\ \mathbf{\Pi}_{ij}\in\{0,1\},\ \mathbf{\Pi}\mathbf{\Pi}^\intercal &= \mathbf{\Pi}^\intercal\mathbf{\Pi} = \mathbf{I} \end{aligned} \tag{5}$$

We find out that the orthogonality constraint of $\mathbf{\Pi}$ is too strong such that it cannot directly be applied, thus we further relax Equation 6 into the following problem:

$$\begin{aligned} \min_{\mathbf{\Pi}} &\|\mathbf{\Pi}\cdot\mathbf{Y}_{(1)}\mathbf{Y}_{(1)}^\intercal - (\mathbf{Y}_{(1)}(\mathbf{E}\odot\mathbf{D})\mathbf{A}^\intercal)^\intercal\|_F^2 \\ &+\gamma_1\|\mathbf{\Pi}\mathbf{\Pi}^\intercal - \mathbf{I}\|_F^2 + \gamma_2\|\mathbf{\Pi}^\intercal\mathbf{\Pi} - \mathbf{I}\|_F^2 \\ s.t.\ &\mathbf{\Pi}\mathbf{1} = \mathbf{1},\ \mathbf{1}^\intercal\mathbf{\Pi} = \mathbf{1}^\intercal,\ H\left(\mathbf{\Pi}(i,:)\right) < b, \forall j \\ &0 \le \mathbf{\Pi}_{ij} \le 1,\ \forall i\in I, j\in J \end{aligned} \tag{6}$$

We use the fmincon [1] solver in Matlab to solve the above problem.

Algorithm 1 describes the overview of our proposed TENALIGN-$\mathcal{L}_1$.

Algorithm 1 Alternating Least Squares Algorithm for TENALIGN-$\mathcal{L}_1$

Input: Tensors $\mathcal{X}$, $\mathcal{Y}$
Output: Factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}$ and $\mathbf{\Pi}$
while not "converged" **do**
$\mathbf{A} = (\mathbf{X}_{(1)}(\mathbf{C}\odot\mathbf{B}) + \mathbf{Y}'_{(1)}(\mathbf{E}\odot\mathbf{D}))\cdot(\mathbf{C}^\intercal\mathbf{C} * \mathbf{B}^\intercal\mathbf{B} + \mathbf{E}^\intercal\mathbf{E} * \mathbf{D}^\intercal\mathbf{D})^{-1}$
$\mathbf{B} = \mathbf{X}_{(2)}(\mathbf{C}\odot\mathbf{A})\cdot(\mathbf{C}^\intercal\mathbf{C} * \mathbf{A}^\intercal\mathbf{A})^{-1}$
$\mathbf{C} = \mathbf{X}_{(3)}(\mathbf{B}\odot\mathbf{A})\cdot(\mathbf{B}^\intercal\mathbf{B} * \mathbf{A}^\intercal\mathbf{A})^{-1}$
$\mathbf{D} = \mathbf{Y}'_{(2)}(\mathbf{E}\odot\mathbf{A})\cdot(\mathbf{E}^\intercal\mathbf{E} * \mathbf{A}^\intercal\mathbf{A})^{-1}$
$\mathbf{E} = \mathbf{Y}'_{(3)}(\mathbf{D}\odot\mathbf{A})\cdot(\mathbf{D}^\intercal\mathbf{D} * \mathbf{A}^\intercal\mathbf{A})^{-1}$
Solve $\mathbf{\Pi}$ in Eq. 6 using fmincon solver in Matlab
end while=0

B. Solution for TENALIGN-$\mathcal{L}_2$

For formula $\mathcal{L}_2$, We can rewrite it by three sub-functions f_{21}, f_{22} and f_{23}:

$$\begin{aligned} \mathcal{L}_2 = &\underbrace{\|\mathcal{X} - [\![\mathbf{A_x}, \mathbf{B}, \mathbf{C}]\!]\|_F^2}_{f_{21}(\mathbf{A_x},\mathbf{B},\mathbf{C})} + \underbrace{\|\mathcal{Y} - [\![\mathbf{A_y}, \mathbf{D}, \mathbf{E}]\!]\|_F^2}_{f_{22}(\mathbf{A_y},\mathbf{D},\mathbf{E})} \\ &+\lambda\underbrace{\|\mathbf{A_x} - \mathbf{\Pi}\mathbf{A_y}\|_F^2}_{f_{23}(\mathbf{A_x},\mathbf{A_y},\mathbf{\Pi})} \end{aligned} \tag{7}$$

1) For partial derivative with respect to A_x :

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{A_x}} = \frac{\partial f_{21}}{\partial \mathbf{A_x}} + \frac{\partial f_{23}}{\partial \mathbf{A_x}}$$
$$= \frac{\partial}{\partial \mathbf{A_x}} \left\|\mathbf{X}_{(1)} - \mathbf{A_x}(\mathbf{C}\odot\mathbf{B})^\intercal\right\|_F^2 + \lambda\frac{\partial}{\partial \mathbf{A_x}} \|\mathbf{A_x} - \mathbf{\Pi}\mathbf{A_y}\|_F^2$$
$$= \frac{\partial}{\partial \mathbf{A_x}} \operatorname{Tr}\left((\mathbf{X}_{(1)} - \mathbf{A_x}(\mathbf{C}\odot\mathbf{B})^\intercal)(\mathbf{X}_{(1)} - \mathbf{A_x}(\mathbf{C}\odot\mathbf{B})^\intercal)^\intercal\right)$$
$$+\frac{\partial}{\partial \mathbf{A_x}} \operatorname{Tr}\left((\mathbf{A_x} - \mathbf{\Pi}\mathbf{A_y})(\mathbf{A_x} - \mathbf{\Pi}\mathbf{A_y})^\intercal\right)$$
$$= -2\mathbf{X}_{(1)}(\mathbf{C}\odot\mathbf{B}) + 2\mathbf{A_x}(\mathbf{C}^\intercal\mathbf{C} * \mathbf{B}^\intercal\mathbf{B}) + 2\lambda(\mathbf{A_x} - \mathbf{\Pi}\mathbf{A_y})$$

Let $\frac{\partial \mathcal{L}_2}{\partial \mathbf{A_x}}$ be zero, then we have the update rule for matrix $\mathbf{A_x}$:

$$\mathbf{A_x} = (\mathbf{X}_{(1)}(\mathbf{C}\odot\mathbf{B}) + \lambda\mathbf{\Pi}\mathbf{A_y})\cdot(\mathbf{C}^\intercal\mathbf{C} * \mathbf{B}^\intercal\mathbf{B} + \lambda\mathbf{I})^{-1}$$

[1] https://www.mathworks.com/help/optim/ug/fmincon.html

2) For partial derivative with respect to B:

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{B}} = \frac{\partial f_{21}}{\partial \mathbf{B}} = \frac{\partial}{\partial \mathbf{B}} \left\| \mathbf{X}_{(2)} - \mathbf{B}(\mathbf{C} \odot \mathbf{A_x})^\intercal \right\|_F^2$$
$$= \frac{\partial}{\partial \mathbf{B}} \operatorname{Tr}\left((\mathbf{X}_{(2)} - \mathbf{B}(\mathbf{C} \odot \mathbf{A_x})^\intercal)(\mathbf{X}_{(2)} - \mathbf{B}(\mathbf{C} \odot \mathbf{A_x})^\intercal)^\intercal\right)$$
$$= -2\mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A_x}) + 2\mathbf{B}(\mathbf{C}^\intercal\mathbf{C} * \mathbf{A_x}^\intercal\mathbf{A_x})$$

Let $\frac{\partial \mathcal{L}_2}{\partial \mathbf{B}}$ be zero, then we have the update rule for matrix $\mathbf{B}$:

$$\mathbf{B} = \mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A_x}) \cdot (\mathbf{C}^\intercal\mathbf{C} * \mathbf{A_x}^\intercal\mathbf{A_x})^{-1}$$

3) For partial derivative with respect to C:

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{C}} = \frac{\partial f_{21}}{\partial \mathbf{C}} = \frac{\partial}{\partial \mathbf{C}} \left\| \mathbf{X}_{(3)} - \mathbf{C}(\mathbf{B} \odot \mathbf{A_x})^\intercal \right\|_F^2$$
$$= \frac{\partial}{\partial \mathbf{C}} \operatorname{Tr}\left((\mathbf{X}_{(3)} - \mathbf{C}(\mathbf{B} \odot \mathbf{A_x})^\intercal)(\mathbf{X}_{(3)} - \mathbf{C}(\mathbf{B} \odot \mathbf{A_x})^\intercal)^\intercal\right)$$
$$= -2\mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A_x}) + 2\mathbf{C}(\mathbf{B}^\intercal\mathbf{B} * \mathbf{A_x}^\intercal\mathbf{A_x})$$

Let $\frac{\partial \mathcal{L}_2}{\partial \mathbf{C}}$ be zero, then we have the update rule for matrix $\mathbf{C}$:

$$\mathbf{C} = \mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A_x}) \cdot (\mathbf{B}^\intercal\mathbf{B} * \mathbf{A_x}^\intercal\mathbf{A_x})^{-1}$$

4) For partial derivative with respect to A_y :

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{A_y}} = \frac{\partial f_{22}}{\partial \mathbf{A_y}} + \frac{\partial f_{23}}{\partial \mathbf{A_y}}$$
$$= \frac{\partial}{\partial \mathbf{A_y}} \left\| \mathbf{Y}_{(1)} - \mathbf{A_y}(\mathbf{E} \odot \mathbf{D})^\intercal \right\|_F^2 + \lambda \frac{\partial}{\partial \mathbf{A_y}} \left\| \mathbf{A_x} - \mathbf{\Pi A_y} \right\|_F^2$$
$$= \frac{\partial}{\partial \mathbf{A_y}} \operatorname{Tr}\left((\mathbf{Y}_{(1)} - \mathbf{A_y}(\mathbf{E} \odot \mathbf{D})^\intercal)(\mathbf{Y}_{(1)} - \mathbf{A_y}(\mathbf{E} \odot \mathbf{D})^\intercal)^\intercal\right)$$
$$+ \frac{\partial}{\partial \mathbf{A_y}} \operatorname{Tr}\left((\mathbf{A_x} - \mathbf{\Pi A_y})(\mathbf{A_x} - \mathbf{\Pi A_y})^\intercal\right)$$
$$= -2\mathbf{Y}_{(1)}(\mathbf{E} \odot \mathbf{D}) + 2\mathbf{A_y}(\mathbf{E}^\intercal\mathbf{E} * \mathbf{D}^\intercal\mathbf{D})$$
$$+ 2\lambda(-\mathbf{\Pi}^\intercal\mathbf{A_x} + \mathbf{\Pi A_y})$$

Because we cannot get $\mathbf{A_y}$ directly from $\frac{\partial \mathcal{L}_2}{\partial \mathbf{A_y}}$, update the factor matrix $\mathbf{A_y}$ by gradient descent.

5) For partial derivative with respect to D:

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{D}} = \frac{\partial f_{22}}{\partial \mathbf{D}} = \frac{\partial}{\partial \mathbf{D}} \left\| \mathbf{Y}_{(2)} - \mathbf{D}(\mathbf{E} \odot \mathbf{A_y})^\intercal \right\|_F^2$$
$$= \frac{\partial}{\partial \mathbf{D}} \operatorname{Tr}\left((\mathbf{Y}_{(2)} - \mathbf{D}(\mathbf{E} \odot \mathbf{A_y})^\intercal)(\mathbf{Y}_{(2)} - \mathbf{D}(\mathbf{E} \odot \mathbf{A_y})^\intercal)^\intercal\right)$$
$$= -2\mathbf{Y}_{(2)}(\mathbf{E} \odot \mathbf{A_y}) + 2\mathbf{D}(\mathbf{E}^\intercal\mathbf{E} * \mathbf{A_y}^\intercal\mathbf{A_y})$$

Let $\frac{\partial \mathcal{L}_2}{\partial \mathbf{D}}$ be zero, then we have the update rule for matrix $\mathbf{D}$:

$$\mathbf{D} = \mathbf{Y}'_{(2)}(\mathbf{E} \odot \mathbf{A}) \cdot (\mathbf{E}^\intercal\mathbf{E} * \mathbf{A}^\intercal\mathbf{A})^{-1}$$

6) For partial derivative with respect to E:

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{E}} = \frac{\partial f_{22}}{\partial \mathbf{E}} = \frac{\partial}{\partial \mathbf{E}} \left\| \mathbf{Y}_{(3)} - \mathbf{E}(\mathbf{D} \odot \mathbf{A_y})^\intercal \right\|_F^2$$
$$= \frac{\partial}{\partial \mathbf{E}} \operatorname{Tr}\left((\mathbf{Y}_{(3)} - \mathbf{E}(\mathbf{D} \odot \mathbf{A_y})^\intercal)(\mathbf{Y}_{(3)} - \mathbf{E}(\mathbf{D} \odot \mathbf{A_y})^\intercal)^\intercal\right)$$
$$= -2\mathbf{Y}_{(3)}(\mathbf{D} \odot \mathbf{A_y}) + 2\mathbf{E}(\mathbf{D}^\intercal\mathbf{D} * \mathbf{A_y}^\intercal\mathbf{A_y})$$

Let $\frac{\partial \mathcal{L}_2}{\partial \mathbf{E}}$ be zero, then we have the update rule for matrix $\mathbf{E}$:

$$\mathbf{E} = \mathbf{Y}_{(3)}(\mathbf{D} \odot \mathbf{A_y}) \cdot (\mathbf{D}^\intercal\mathbf{D} * \mathbf{A_y}^\intercal\mathbf{A_y})^{-1}$$

7) For partial derivative with respect to Π : To update the value of $\mathbf{\Pi}$, we need to solve $\|\mathbf{A_x} - \mathbf{\Pi A_y}\|_F^2 = 0$ for $\mathbf{\Pi}$ with the following constraints in Equation 8, which softly forces the factor matrices of $\mathcal{X}$ and $\mathcal{Y}$ to be in the same space.

$$\begin{aligned} \|\mathbf{A_x} - \mathbf{\Pi A_y}\|_F^2 &= 0 \\ s.t.\ \mathbf{\Pi 1} = \mathbf{1},\ \mathbf{1}^\intercal\mathbf{\Pi} &= \mathbf{1}^\intercal, \\ \mathbf{\Pi}_{ij} \in \{0,1\},\ \mathbf{\Pi\Pi}^\intercal &= \mathbf{\Pi}^\intercal\mathbf{\Pi} = \mathbf{I} \end{aligned} \tag{8}$$

For the same reason with Algorithm 1 for $\mathcal{L}_1$, we relax Equation 9 into the following problem and also solve it by linear programming:

$$\begin{aligned} \min_{\mathbf{\Pi}} \|\mathbf{A_x} - \mathbf{\Pi A_y}\|_F^2 + \gamma_1\|\mathbf{\Pi\Pi}^\intercal - \mathbf{I}\|_F^2 + \gamma_2\|\mathbf{\Pi}^\intercal\mathbf{\Pi} - \mathbf{I}\|_F^2 \\ s.t.\ \mathbf{\Pi 1} = \mathbf{1},\ \mathbf{1}^\intercal\mathbf{\Pi} = \mathbf{1}^\intercal,\ H\left(\mathbf{\Pi}(i,:)\right) < b, \forall j \\ 0 \le \mathbf{\Pi}_{ij} \le 1,\ \forall i \in I, j \in J \end{aligned} \tag{9}$$

We use the fmincon [2] solver in Matlab to solve the above problem.

The algorithm for TENALIGN-$\mathcal{L}_2$ is described in Algorithm 2 below.

Algorithm 2 Alternating Least Squares Algorithm for TENALIGN-$\mathcal{L}_2$

Input: Tensors $\mathcal{X}$, $\mathcal{Y}$
Output: Factor matrices $\mathbf{A_x}, \mathbf{B}, \mathbf{C}, \mathbf{A_y}, \mathbf{D}, \mathbf{E}$ and $\mathbf{\Pi}$

```
while not "converged" do
    A_x = (X_(1)(C ⊙ B) + λΠA_y) · (C^⊺C * B^⊺B + λI)^{-1}
    B = X_(2)(C ⊙ A_x) · (C^⊺C * A_x^⊺A_x)^{-1}
    C = X_(3)(B ⊙ A_x) · (B^⊺B * A_x^⊺A_x)^{-1}
    while ||A_y^{i+1} − A_y^i||_F^2 ≥ threshold do
        A_y^{i+1} = A_y^i − α · ∂L_2/∂A_y
    end while
    D = Y'_(2)(E ⊙ A) · (E^⊺E * A^⊺A)^{-1}
    E = Y_(3)(D ⊙ A_y) · (D^⊺D * A_y^⊺A_y)^{-1}
    Solve Π in Eq. 9 using fmincon solver in Matlab
end while=0
```

For factor matrix updates in Alg.1 and Alg.2 we are using the MTTKRP (Matricized Tensor Times Khatri-Rao Product) operation to speed up the computations.

IV. EXPERIMENTAL EVALUATION

Here, we evaluate our proposed algorithms from the perspectives of *accuracy*, *sensitivity analysis*, *ablation study*, and *sanity check* by answering the following questions:

1) Accuracy: Can our algorithms outperform the baseline alignment methods in terms of latent factor matching and alignment accuracy?
2) Sensitivity analysis: How sensitive are our algorithms to important hyperparameters?
3) Ablation study: How is the performance affected by different constraints on $\mathbf{\Pi}$?
4) Sanity check: Here we check if each component of our our algorithms help with the final solution.

[2]https://www.mathworks.com/help/optim/ug/fmincon.html

A. Experiment Setup and Datasets

We implement TENALIGN in Matlab and our implementation is publicly available[3]. In order to fully test our algorithms, we use real-world coupled data tensors, the chemistry datasets [11] and a large number of synthetic data tensors generated by [12]. For all datasets, before delivering the input tensors, we do column-wise normalization and tensor normalization.

1) Synthetic Data: For each experiment with synthetic dataset, we run it for 100 times where each time with random generated datasets. Tensors $\mathcal{X}$ and $\mathcal{Y}$ are built following CPD model from already known factors, which are also used as the true factors when measuring the factor accuracy, or called factor matching score. Specifically, the two synthetic datasets by [12] where both datasets' tensors are first-mode coupled and contain 100 pairs of tensors which are randomly drawn from a normal distribution. The synthetic dataset 1 (namely Syn 1) contains a tensor $\mathcal{T}_1$ of size $10 \times 30 \times 40$ coupled with a tensor $\mathcal{T}_2$ of size $10 \times 70 \times 10$ and they are created by ground-truth factor matrices with rank $R = 4$. The synthetic dataset 2 (namely Syn 2) contains a tensor $\mathcal{T}_3$ of size $15 \times 30 \times 40$ coupled with a tensor $\mathcal{T}_4$ of size $15 \times 70 \times 10$ and they are created by ground-truth factor matrices with rank $R = 5$.

2) Real Data: We use real-world public chemistry datasets (namely Chem) [11] which have know joint CPD structure and are publicly available[4]. This dataset includes three datasets coupled in the first mode, which consist of chemical samples observed using different measurement techniques: two tensors EEM of size $28 \times 13324 \times 8$ and NMR, and one matrix LC-MS of size 28×168. Note that NMR contains five chemical information, LC-MS contains four, and EEM only contains three, this means that this coupled matrix tensor datasets doesn't have all components shared. Because TENALIGN assumes all components shared, we only use tensor EEM and matrix LC-MS which exhibit that structure [11].

3) Metrics: We measure TENALIGN-$\mathcal{L}_1$ and TENALIGN-$\mathcal{L}_2$ with three metrics: raw accuracy measurement for alignment defined in Eq. 10, factor matching score, and clustering accuracy measurement for alignment.

$$\text{Raw accuracy} = \frac{\text{No. of correctly aligned entities}}{\text{No. of entities}} \tag{10}$$

For factor accuracy measurement, we use factor matching score (FMS). There the learned factors $\hat{\mathbf{C}}_{\mathbf{i},\mathbf{d}}$ are tested by how well can they match with the true factors $\mathbf{C}_{\mathbf{i},\mathbf{d}}$ by FMS function defined below [12]:

$$\text{FMS} = \prod_{i=1}^{N} \frac{1}{R_i} \sum_{r=1}^{R_i} \left(\prod_{d=1}^{D_i} \frac{\langle \mathbf{C}_{\mathbf{i},\mathbf{d}}(:,r), \hat{\mathbf{C}}_{\mathbf{i},\mathbf{d}}(:,r) \rangle}{\|\mathbf{C}_{\mathbf{i},\mathbf{d}}(:,r)\|_2 \|\hat{\mathbf{C}}_{\mathbf{i},\mathbf{d}}(:,r)\|_2} \right) \tag{11}$$

where N is the number of tensors we have in the model, and each tensor $\mathcal{T}_i$ of order $D_i \geq 2$ has R_i components.

Typically, state-of-the-art graph alignment methods measure alignment accuracy via a "soft"-alignment approach, where they identify a list of potential candidates for a given node, and if that list contains the correct matching node, this is counted as a successful match [22]. The rationale behind existing works for this is that due to node/entity similarities and correlations, strict matching may be impossible and, thus, measuring the accuracy in a strict manner may not paint a fair picture of how the alignment algorithm actually performs.

In our case, we propose a similar in spirit clustering-based accuracy measurement. We first group entities in the first mode into k clusters, and assume that two entities can be regarded as the same one if they are from the same cluster. Consider the property of the row permutation matrix $\mathbf{\Pi}_{row}$, if $\mathbf{\Pi}_{row}(i,j) = p$ where $0 \leq p \leq 1$, this means that the j-th row will be moved to i-th position with probability p. For each row i of the learned alignment matrix, compare the true matrix $\mathbf{\Pi}(j,i)$ and the learned matrix $\hat{\mathbf{\Pi}}(k,i)$. If $j = k$, row i-th is correct aligned. If $j \neq k$, only if the destinations row j and row k are in the same cluster, then it is counted as correct alignment. Numerically, our clustering accuracy measurement metric is defined below

$$\mathbf{\Pi} \text{ accuracy} = \frac{\text{No. correctly aligned entities based on clustering}}{\text{No. of entities}}$$

It's worth to mention that when we don't perform clustering, i.e., each entity is in the cluster of itself, $\mathbf{\Pi}$ accuracy = Raw accuracy. Without loss of generality, we use $\mathbf{\Pi}$ accuracy to capture alignment performance as Raw accuracy is its special case. In our experiments, due to space limitations, we present (a) results for the best performing number of clusters for each given baseline and dataset (where in some cases, the best performance was observed without the need for clustering), and (b) for a given dataset and algorithm combination we demonstrate the behavior of the accuracy measured for different numbers of clusters k.

B. TENALIGN *Variants*

We are testing three variants of our method: TENALIGN-$\mathcal{L}_1$, TENALIGN-$\mathcal{L}_2$, and TENALIGN-$\mathcal{L}_2$-REGAL. For the latter, we substitute our alignment method in formulation $\mathcal{L}_2$ with REGAL's alignment algorithm. This is meant to (a) demonstrate the flexibility of TENALIGN, which can incorporate any existing state-of-the-art alignment method, and (b) compare the performance of this state-of-the-art alignment component against our proposed relaxed optimization scheme.

C. Baseline Methods

Since this work is the first to perform joint tensor alignment and coupled factorization, we were not able to identify published baseline methods to compare against, however, we are comparing against the following schemes which conceptually represent different baseline methods.

- CPD-REGAL: First conduct separate CPD factorizations to $\mathcal{X}$ and $\mathcal{Y}$ and we subsequently apply REGAL's alignment [22] to the row embeddings (factor matrices) computed by the CPD. This is not an iterative algorithm but a two-step process which closely mimics state-of-the-art approaches. This is meant to test the quality of the alignment obtained via this two-step process.

[3]https://github.com/yunshuwu/TenAlign

[4]http://www.models.life.ku.dk/joda/prototype

- CPD-REGAL-LCMTF: This baseline obtains the alignment matrix the exact same way as CPD-REGAL and subsequently uses that alignment matrix within a very recent Linearly Coupled Matrix Tensor Factorization (LCMTF) [12] algorithm. This is meant to test the quality of the factors obtained if we were to use the "traditional" two-step process for alignment.

D. Hyperparameter Selection

To show fair results, for each experiment we run 100 times and deliver the results. For all the algorithms, without specification when applicable, by default we set $R = 4$ and $\lambda = 5$ when testing Syn 1, $R = 5$, $k = 3$ and $\lambda = 5$ on Syn 2, and $R = 3$, $k = 4$, and $\lambda = 10$ on Chem. We set $\gamma_1 = \gamma_2 = \gamma_3 = 0.01$.

E. Alignment and Factor Accuracy

We study the alignment and factor accuracy with experiments on one real-world coupled chemistry datasets [11] and another two synthetic datasets. In Fig. 4, we plot the estimated two-dimensional probability density function (2d-pdf) to visualize the joint distribution of FMS and $\mathbf{\Pi}$ accuracy using *seaborn.kdeplot* in Python which represents the two metrices using a continuous probability density curve in two dimensions. In Table II, we shows the means and standard deviation respect to $\mathbf{\Pi}$ accuracy and FMS. The results in Fig. 4 and Table II show that our algorithms outperform the competing alternatives in terms of higher FSM and $\mathbf{\Pi}$ accuracy.

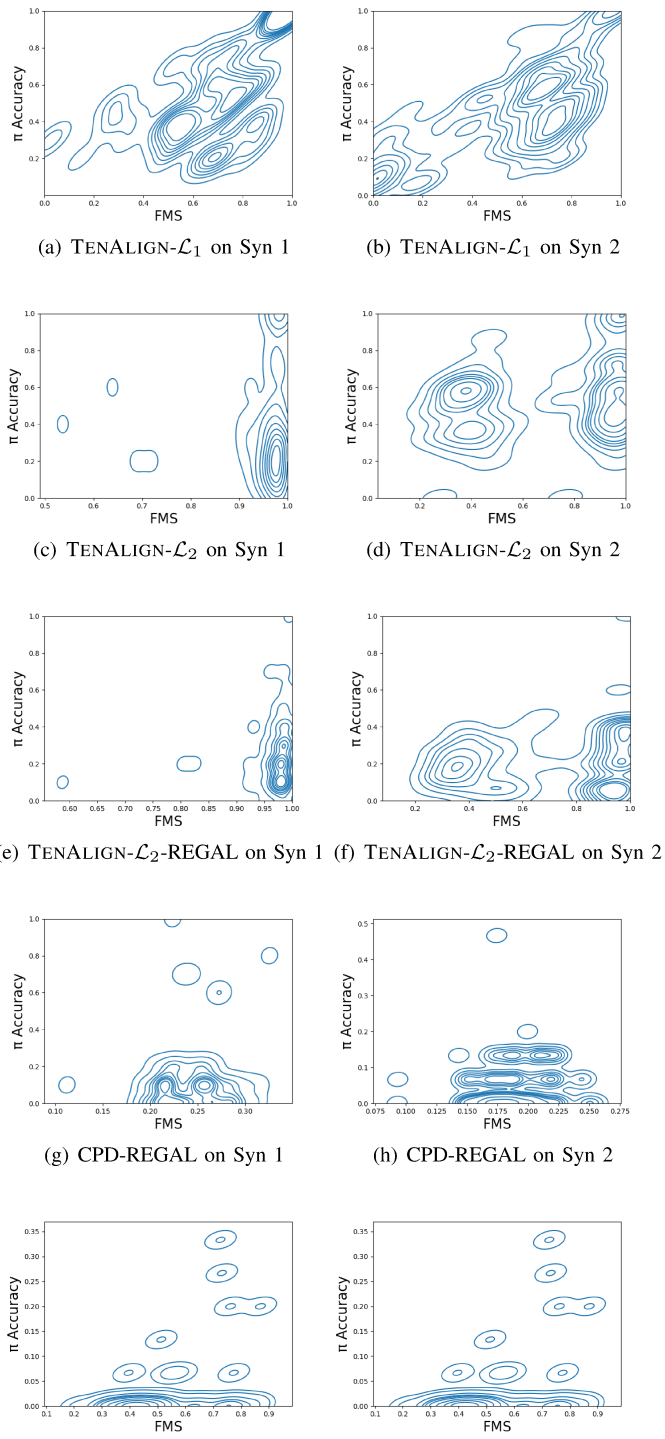
Dataset	Method	$\mathbf{\Pi}$ accuracy	FMS
Syn 1	TENALIGN-$\mathcal{L}_1$	$\mathbf{.616 \pm .299}$	$.735 \pm .238$
	TENALIGN-$\mathcal{L}_2$	$.3450 \pm .2876$	$.9597 \pm .0679$
	TENALIGN-$\mathcal{L}_2$-REGAL	$.2290 \pm .1622$	$\mathbf{.9708 \pm .0471}$
	CPD-REGAL	$.1060 \pm .1769$	$.2439 \pm .0343$
	CPD-REGAL-LCMTF	$.0490 \pm .0785$	$.5658 \pm .1811$
Syn 2	TENALIGN-$\mathcal{L}_1$ (no k)	$.4540 \pm .2426$	$.5681 \pm .2848$
	TENALIGN-$\mathcal{L}_2$	$\mathbf{.5280 \pm .1973}$	$.7440 \pm .2840$
	TENALIGN-$\mathcal{L}_2$-REGAL	$.2080 \pm .1552$	$\mathbf{.7470 \pm .2707}$
	CPD-REGAL	$.0387 \pm .0503$	$.1937 \pm .0249$
	CPD-REGAL-LCMTF	$.0140 \pm .0530$	$.5144 \pm .1614$
Chem	TENALIGN-$\mathcal{L}_1$	$\mathbf{.4578 \pm .2087}$	N/A
	TENALIGN-$\mathcal{L}_2$	$.2678 \pm .0738$	N/A
	TENALIGN-$\mathcal{L}_2$-REGAL	$.1411 \pm .0681$	A/A
	CPD-REGAL	$.0250 \pm .0447$	N/A
	CPD-REGAL-LCMTF	$.0250 \pm .0447$	N/A

TABLE II: FMS and $\mathbf{\Pi}$ accuracy for different methods on different datasets in the format of $a \pm b$ where a is mean and b is standard derivation from 100 independent tests; FMS values on dataset Chem are not available (N/A) due to the lack of ground truth latent factors; the best performance for each dataset per metric is in bold.

F. Sensitivity Analysis

We evaluate the sensitivity of our TENALIGN $\mathcal{L}_1$ using different initialization methods and the important hyperparameter λ in $\mathcal{L}_2$ which determines how close are spaces of factors of $\mathcal{X}$ and $\mathcal{Y}$.

1) $\mathbf{\Pi}$ Initialization:: For initlaization, we show the results of TENALIGN-$\mathcal{L}_1$ to give a hint of how initialization methods (see below) influence the performance. From the results in Fig. 4, we can show that FMS and $\mathbf{\Pi}$ accuracy are always strongly correlated and the performance of our proposed TENALIGN-$\mathcal{L}_1$ is not sensitive to the initialization.



(a) TENALIGN-$\mathcal{L}_1$ on Syn 1 (b) TENALIGN-$\mathcal{L}_1$ on Syn 2

(c) TENALIGN-$\mathcal{L}_2$ on Syn 1 (d) TENALIGN-$\mathcal{L}_2$ on Syn 2

(e) TENALIGN-$\mathcal{L}_2$-REGAL on Syn 1 (f) TENALIGN-$\mathcal{L}_2$-REGAL on Syn 2

(g) CPD-REGAL on Syn 1 (h) CPD-REGAL on Syn 2

(i) CPD-REGAL-LCMTF on Syn 1 (j) CPD-REGAL-LCMTF on Syn 2

Fig. 2: Joint distribution of FMS and $\mathbf{\Pi}$ accuracy among 100 Monte Carlo tests from four algorithms on two datasets; our proposed methods (top three rows) achieve higher FMS and higher $\mathbf{\Pi}$ accuracy statistically than competing alternatives.

Different $\mathbf{\Pi}$ initialization methods:

- Init. 1: Use random decimal matrices to initialize factor matrices and $\mathbf{\Pi}$.

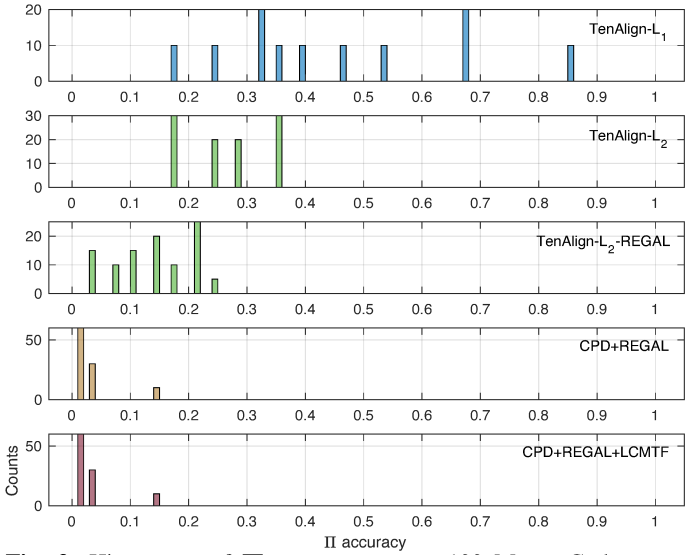


Fig. 3: Histograms of $\mathbf{\Pi}$ accuracy among 100 Monte Carlo tests from four algorithms on dataset Chem; the panels from the top to the bottom show the results of the algorithms TENALIGN-$\mathcal{L}_1$, TENALIGN-$\mathcal{L}_2$, TENALIGN-$\mathcal{L}_2$-REGAL, CPD-REGAL and CPD-REGAL-LCMTF accordingly,; our methods outperform the state-of-the-art.

- Init. 2: Use random decimal matrices to initialize factor matrices and random permutaiton matrix for $\mathbf{\Pi}$.
- Init. 3: Use random decimal matrices initialize factor matrices and $\mathbf{\Pi}$ is initialized by first taking the first slice of both tensor $\mathcal{X}$ and $\mathcal{Y}$, and then using linear least squared solver to align them.
- Init. 4: Use CPD factors of tensor $\mathcal{X}$ and $\mathcal{Y}$ as the initial factor matrices and the same way as Init. 3 for $\mathbf{\Pi}$.

2) Hyperparameter λ of TENALIGN-$\mathcal{L}_2$*:* The hyperparameter λ in TENALIGN-$\mathcal{L}_2$ is designed to force factors of both tensors $\mathcal{X}$ and $\mathcal{Y}$ into two closely intersected spaces. The larger the λ, the closer the two spaces are. In Fig. 7, we can show that our TENALIGN-$\mathcal{L}_2$ performs well in terms of various metrics with proper λ.

3) Clustering-based accuracy sensitivity: Figure 8 shows the behavior of the clustering-based accuracy for varying number of clusters k TENALIGN-$\mathcal{L}_1$ on Syn 1. We observe that the measured accuracy is relatively stable as the number of clusters changes.

G. Ablation Study: Effectiveness of Constraints

Here we analyze the effect of each constraint imposed on the $\mathbf{\Pi}$ matrix in order to determine whether all constraints are necessary for the recovery of a highly-accurate alignment. Figure 6 shows results for TENALIGN-$\mathcal{L}_1$ on dataset Syn 1 when different combinations of constraints are active. We observe that as we add more constraints, the performance increases, since the algorithm is more likely to identify an alignment with high accuracy.

H. Sanity Check of $\mathcal{L}_1$

Here, we verify if every term of formulation $\mathcal{L}_1$ is necessary, by answering the following two questions:

1) Will the learned permutation matrix $\mathbf{\Pi}_e$ be of high quality if we use ground-truth factors as input factors?
2) Will the learned factors be good if we use ground-truth permutation matrix as input matrix?

In our experiments, a third-order tensor of size $10 \times 30 \times 40$ is coupled in the first mode with another third-order tensor of size $10 \times 70 \times 10$, where both tensors have rank $R = 4$ and follow normal distribution. We find out that given ground-truth factors as input factors, the raw accuracy of the learned permutation matrix $\mathbf{\Pi}_e$ is 100%. Also, given true permutation matrix as input $\mathbf{\Pi}$ matrix, factor matching score between learned and true factors is about 0.99. We, thus, conclude, that every part of the TENALIGN $\mathcal{L}_1$ is essential.

V. Related Work

A. Coupled Tensor Factorization

Coupled tensor factorization [7], [24], [25] typically refers to the factorization of two or more datasets which form tensors and/or matrices into a set of latent factors that are common for modes of the datasets that are "coupled", i.e., they have 1-1 or partial correspondence. There have been variants of the traditional formulation which account for some unshared latent factors across coupled datasets [10], [11].

The closest work to our work is the recently proposed Linearly Coupled Matrix Tensor Factorization (LCMTF) [12] which assumes that two datasets are coupled via a *known* linear transformation. However, as we demonstrate in our experiments, if we use a fixed alignment matrix (which is most likely going to be imperfect, unless we have access to the optimal permutation matrix, which is not realistic), this method will fail to recover the true factors, which further motivates our joint alignment and coupled factorization approach.

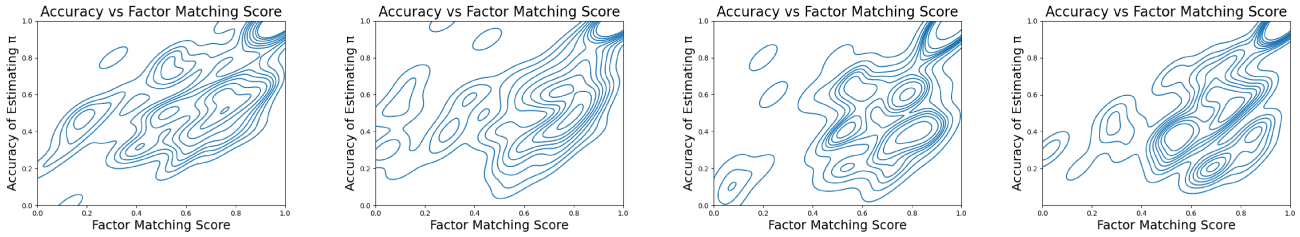
B. Alignment Methods

There is a strong interest in the community for the special case of the entity alignment problem, where entities are nodes in a graph, as evidenced by a rich number of publications in the recent years [15], [16], [17], [18], [19], [26], [27], [20], [21], [22]. Beyond graph matching, there exist recent approaches for general entity alignment [13], [14], [28], [29], [30].

In general, our proposed TENALIGN framework is *synergistic with and not competing against* the aforementioned lines of work: any advances in performing better alignment of entity embeddings directly benefits TENALIGN, since we can readily substitute our relaxed alignment subproblem (as we demonstrate experimentally by borrowing REGAL's matching function) and can subsume and absorb any benefits conferred by the improved alignment method.

VI. Conclusions & Future Work

In this work we introduce TENALIGN, the first joint tensor alignment and coupled factorization framework. We propose



(a) Init.1 (b) Init. 2 (c) Init.3 (d) Init.4

Fig. 4: Joint distribution of FMS and $\mathbf{\Pi}$ accuracy of TENALIGN-$\mathcal{L}_1$ on Syn 1 with different initialization strategies; the results show that FMS and $\mathbf{\Pi}$ accuracy are always strongly correlated and the performance of our proposed TENALIGN-$\mathcal{L}_1$ are not sensitive to the initialization.

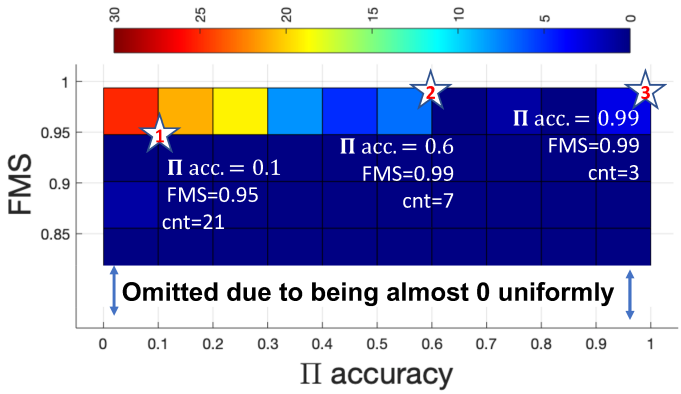


Fig. 5: Number of tests out of 100 w.r.t $\mathbf{\Pi}$ accuracy and FMS using our TENALIGN-$\mathcal{L}_2$ with $\lambda = 5$ on dataset Syn 2; the three starts indicate cases when $\mathbf{\Pi}$ is low, median, and high while FMS is constantly high.

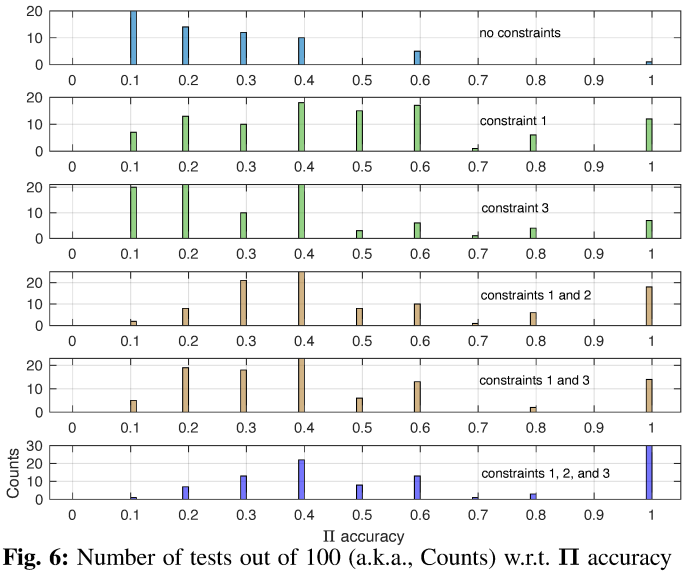


Fig. 6: Number of tests out of 100 (a.k.a., Counts) w.r.t. $\mathbf{\Pi}$ accuracy under different combinations of constraints using our TENALIGN-$\mathcal{L}_1$ on dataset Syn 1; the more constraints are used the better performance in terms of higher counts of perfect $\mathbf{\Pi}$ accuracy we achieve; all the three constraints for our algorithm are essential.

two optimization formulations and we explore their trade-offs, while also comparing our propose joint scheme with

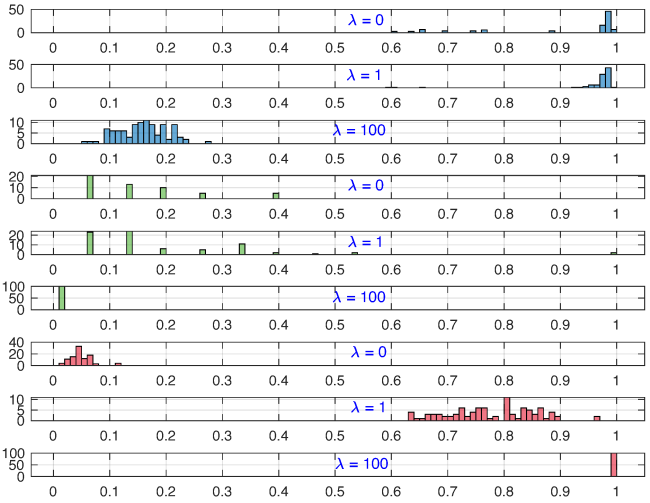


Fig. 7: Sensitivity analysis of the latent factor matching coefficient λ in TENALIGN-$\mathcal{L}_2$ using dataset Syn 2; the top 3 panels are FMS for $\lambda = 0, 1, 100$ respectively; the middle 3 panels are $\mathbf{\Pi}$ accuracy for different λs; the bottom 3 panels are component overlapping rate for various λs; the results show that our TENALIGN-$\mathcal{L}_2$ performs well in terms of various metrics with proper λ.

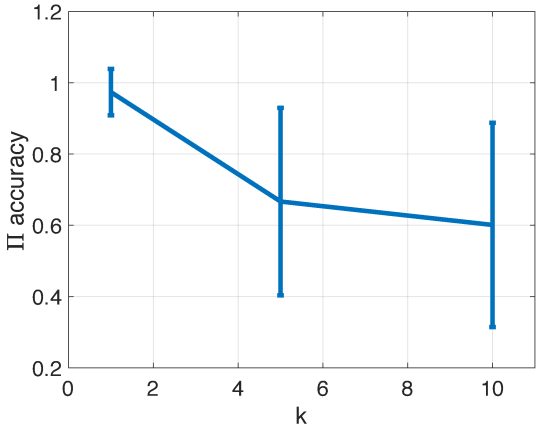


Fig. 8: Sensitivity analysis on k using TENALIGN-$\mathcal{L}_2$ on Syn 2.

alterntatives from the state of the art. We demonstrate that our joint alignment and factorization substantially outperforms multi-step approaches where we embed, align, and coupled factorize in the end, and where each step is independent.

In future work we will focus on exploring algorithmic

improvements in the alignment subproblem, especially as it pertains to scalability, but also in terms of obtaining a better solution to the optimization problem. At the same time we will explore variations of the coupled factorization paradigm where e.g., there are shared and individual latent factors.

VII. Acknowledgements

Research was supported by the National Science Foundation under CAREER grant no. IIS 2046086 and CREST Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (MECIS) grant no. 2112650, a UCR Regents Faculty Fellowship, and a CISCO Faculty Research Award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

References

[1] T. Kolda and B. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, 2009.

[2] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Signal Processing Magazine*.

[3] D. Nion, K. N. Mokios, N. D. Sidiropoulos, and A. Potamianos, "Batch and adaptive parafac-based blind separation of convolutive speech mixtures," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1193–1207, 2009.

[4] G. Zhou, Q. Zhao, Y. Zhang, T. Adalı, S. Xie, and A. Cichocki, "Linked component analysis from matrices to high-order tensors: Applications to biomedical data," *Proceedings of the IEEE*, vol. 104, no. 2, pp. 310–331, 2016.

[5] G. B. Guacho, S. Abdali, N. Shah, and E. E. Papalexakis, "Semi-supervised content-based detection of misinformation via tensor embeddings," in *2018 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM)*. IEEE, 2018, pp. 322–325.

[6] Z. Zhou, J. Fang, L. Yang, H. Li, Z. Chen, and R. S. Blum, "Low-rank tensor decomposition-aided channel estimation for millimeter wave mimo-ofdm systems," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 7, pp. 1524–1538, 2017.

[7] E. Acar, T. G. Kolda, and D. M. Dunlavy, "All-at-once optimization for coupled matrix and tensor factorizations," *arXiv preprint arXiv:1105.3422*, 2011.

[8] E. Acar, M. A. Rasmussen, F. Savorani, T. Næs, and R. Bro, "Understanding data fusion within the framework of coupled matrix and tensor factorizations," *Chemometrics and Intelligent Laboratory Systems*, vol. 129, pp. 53–63, 2013.

[9] E. E. Papalexakis, C. Faloutsos, T. M. Mitchell, P. P. Talukdar, N. D. Sidiropoulos, and B. Murphy, "Turbo-smt: Accelerating coupled sparse matrix-tensor factorizations by 200x," in *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 2014, pp. 118–126.

[10] E. Acar, A. J. Lawaetz, M. Rasmussen, and R. Bro, "Structure-revealing data fusion model with applications in metabolomics," in *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*. IEEE, 2013, pp. 6023–6026.

[11] E. Acar, E. E. Papalexakis, M. A. Rasmussen, A. J. Lawaetz, M. Nilsson, and R. Bro, "Structure-revealing data fusion," *BMC bioinformatics*, vol. 15, no. 1, p. 239, 2014.

[12] E. A. Carla Schenker, Jérémy Cohen, "An optimization framework for regularized linearly coupled matrix-tensor factorization," in *EUSIPCO 2020 - 28th European Signal Processing Conference, Jan 2021, Virtual, Netherlands*. EUSIPCO, 2020, pp. 1–5.

[13] Y. L. Jin Wang and W. Hirota, "Machamp: a generalized entity matching benchmark," in *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia. ACM, New York, NY, USA*. ACM, 2021, pp. 4633–4642.

[14] Y. S. A. D. Yuliang Li, Jinfeng Li and W.-C. Tan, "Deep entity matching with pre-trained language models," in *PVLDB, 14(1): XXX-XXX, 2021*. PVLDB, 2021, pp. 4633–4642.

[15] Y. Yan, L. Liu, Y. Ban, B. Jing, and H. Tong, "Dynamic knowledge graph alignment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4564–4572.

[16] T. Derr, H. Karimi, X. Liu, J. Xu, and J. Tang, "Deep adversarial network alignment," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 352–361.

[17] Y. Yan, S. Zhang, and H. Tong, "Bright: A bridging algorithm for network alignment," in *Proceedings of the Web Conference 2021*, 2021, pp. 3907–3917.

[18] M. Heimann, X. Chen, F. Vahedian, and D. Koutra, "Refining network alignment to improve matched neighborhood consistency," in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 2021, pp. 172–180.

[19] S. Zhang and H. Tong, "Network alignment: recent advances and future directions," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 3521–3522.

[20] K. K. Qin, F. D. Salim, Y. Ren, W. Shao, M. Heimann, and D. Koutra, "G-crewe: Graph compression with embedding for network alignment," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1255–1264.

[21] S. Zhang, H. Tong, R. Maciejewski, and T. Eliassi-Rad, "Multilevel network alignment," in *The World Wide Web Conference*, 2019, pp. 2344–2354.

[22] M. Heimann, H. Shen, T. Safavi, and D. Koutra, "REGAL: representation learning-based graph alignment," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. ACM, 2018, pp. 117–126.

[23] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.

[24] B. Jeon, I. Jeon, L. Sael, and U. Kang, "Scout: Scalable coupled matrix-tensor factorization-algorithm and discoveries," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 811–822.

[25] K. Yılmaz, A. Cemgil, and U. Simsekli, "Generalised coupled tensor factorisation," *Advances in neural information processing systems*, vol. 24, 2011.

[26] S. Zhang, H. Tong, Y. Xia, L. Xiong, and J. Xu, "Nettrans: Neural cross-network transformation," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 986–996.

[27] Q. Zhu, H. Wei, B. Sisman, D. Zheng, C. Faloutsos, X. L. Dong, and J. Han, "Collective multi-type entity alignment between knowledge graphs," in *Proceedings of The Web Conference 2020*, 2020, pp. 2241–2252.

[28] D. Zhang, Y. Nie, S. Wu, Y. Shen, and K.-L. Tan, "Multi-context attention for entity matching," in *Proceedings of The Web Conference 2020*, 2020, pp. 2634–2640.

[29] M. Berrendorf, E. Faerman, and V. Tresp, "Active learning for entity alignment," in *European Conference on Information Retrieval*. Springer, 2021, pp. 48–62.

[30] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra, "Deep learning for entity matching: A design space exploration," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 19–34.