Sylvester Tensor Equation for Multi-Way Association

Boxin Du, Lihui Liu, Hanghang Tong
Department of Computer Science, University of Illinois at Urbana-Champaign
{boxindu2,lihuil2,htong}@illinois.edu

ABSTRACT

How can we identify the same or similar users from a collection of social network platforms (e.g., Facebook, Twitter, LinkedIn, etc.)? Which restaurant shall we recommend to a given user at the right time at the right location? Given a disease, which genes *and* drugs are most relevant? Multi-way association, which identifies strongly correlated node sets from multiple input networks, is the key to answering these questions. Despite its importance, very few multi-way association methods exist due to its high complexity. In this paper, we formulate multi-way association as a convex optimization problem, whose optimal solution can be obtained by a Sylvester tensor equation. Furthermore, we propose two fast algorithms to solve the Sylvester tensor equation, with a *linear* time and space complexity. We further provide theoretic analysis in terms of the sensitivity of the Sylvester tensor equation solution. Empirical evaluations demonstrate the efficacy of the proposed method.

CCS CONCEPTS

• **Information systems** → **Data mining**; • **Theory of computation** → *Graph algorithms analysis.*

KEYWORDS

multi-way association; Sylvester tensor equation; multi-network mining; graph mining

ACM Reference Format:
Boxin Du, Lihui Liu, Hanghang Tong. 2021. Sylvester Tensor Equation for Multi-Way Association. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM,, 11 pages. https://doi.org/10.1145/3447548.3467336

1 INTRODUCTION

Multiple networks are ubiquitous in many important applications. For example, how to link identical or similar users from multiple social networks (i.e., collective network alignment) [18]? How to simultaneously recommend items, activities as well as locations to a user (i.e., high-order recommendation) [11, 19]? In bioinformatics, how to discover relevant drugs and genes for a specific disease [2]? In team management, how to optimally assign team members with the right skills to the right teams for relevant tasks [15, 38]? The key to answering these questions lies in *multi-way association*, which identifies strongly correlated nodes from multiple networks.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467336

For pair-wise association (e.g., network alignment [6, 13, 16, 32]), it links *node-pairs* across two input networks. On the contrary, multi-way association aims to discover the collective association w.r.t. to *a set of nodes*. Fig. 1 presents an illustrative example. Given three social networks, a multi-way association (e.g., the red dashed box on the left of Fig. 1) is a set of three users (i.e. nodes) from each of the three input social networks, who are either identical or similar with each other. We can represent all the multi-way associations in the form of a 3^{rd}-order tensor $\mathcal{X}$ (e.g., the right-most part in Fig. 1), and the weights of the tensor entries measure the strength of the corresponding multi-way associations. For a given user, each entry of the corresponding *slice* of tensor $\mathcal{X}$ with a high weight indicates *a pair of* users from the other two social networks who are both strongly associated with the given user. Likewise, given an activity network, a social network and a location network, strong multi-way associations inferred from them could indicate that the corresponding activities, users, and locations are associated with each other.

Compared with the pair-wise association, much fewer methods exist for multi-way association due to a number of challenges. First (*C1. Problem Formulation*), pair-wise cross-network association is often formulated as an optimization problem [13, 16, 32]. For example, soft network alignment [16, 25, 32] can be formulated as a convex optimization problem based on the alignment consistency principle. However, it is not clear if such a consistency principle would generalize to multi-way association. Second (*C2. Algorithms*), even if we can formulate multi-way association from the optimization perspective, it is still highly challenging to solve it in terms of its optimality and sensitivity. Third (*C3. Scalability*), the solution space of multi-way association is significantly larger than pair-wise association. To see this, suppose there are K input networks, each with n nodes. There could be as many as n^K multi-way associations. Therefore, another major hurdle is to scale-up the multi-way association algorithm to large networks. In this paper, we address these three challenges, and our main contributions are summarized as follows.

- **Formulation**. We formulate the multi-way association problem as an optimization problem and show that it can be solved optimally by a Sylvester tensor equation.
- **New Algorithms**. We propose two fast algorithms (SyTE-Fast-P and SyTE-Fast-A) to solve the Sylvester tensor equation on plain networks and attributed networks respectively, with a linear complexity in both time and space.
- **Proofs and Analysis**. We provide theoretic analysis of the proposed algorithms in terms of optimality, sensitivity and complexity.
- **Empirical Evaluations**. We conduct extensive empirical evaluations on a diverse set of real networks which demonstrate the efficacy of the proposed methods.

The rest of the paper is organized as follows. Section 2 gives the formal definition of the multi-way association problem. Section 3 presents the optimization formulation of the multi-way association problem and a basic algorithm. Section 4 introduces an accelerated algorithms for solving the optimization problem on plain networks, together with some analysis. Section 5 proposes an accelerated algorithm to solve the optimization problem on attributed networks. The experimental results are presented in Section 7. The related works are reviewed in Section 8.

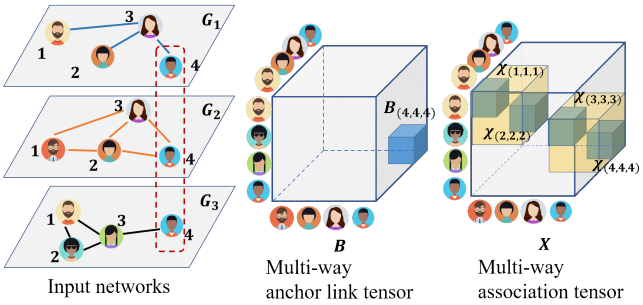


Figure 1: An illustrative example of multi-way association for collective social network alignment. Left (from top to bottom): three input social networks, such as LinkedIn, Facebook and Twitter. The red dashed rectangle indicates a prior anchor link ($\mathcal{B}$ in the middle). Right: solution tensor $\mathcal{X}$ of the corresponding Sylvester tensor equation gives the inferred multi-way association; the highlighted cubes in $\mathcal{X}$ denote four strong multi-way associations. Best viewed in color. See the details in Sections 2-3.

2 PROBLEM DEFINITION

We first briefly introduce the notations used in the paper. We use bold uppercase letters to represent matrices, bold lowercase letters to represent vectors, bold calligraphic letters to represent tensors, lowercase or uppercase letters in regular font for scalars, and calligraphic letters for multi-index (e.g. $\mathfrak{L}$). Specifically, each network G is represented by three matrices, including the (weighted) adjacency matrix $\mathbf{A}$, the edge attribute matrix $\mathbf{E}$ and the node attribute matrix $\mathbf{N}$. The i-th element of a vector $\mathbf{x}$ is denoted as $\mathbf{x}_i$, the element (i, j) of a matrix $\mathbf{A}$ is denoted as $\mathbf{A}(i, j)$, and the element $(i_1, i_2, ..., i_k)$ of a tensor $\mathcal{X}$ is denoted as $\mathcal{X}(i_1, i_2, ..., i_k)$. We focus on categorical edge and node attributes in this paper. $\mathbf{E}$ has the same dimension as its corresponding adjacency matrix $\mathbf{A}$, and $\mathbf{E}^p(i, j) = 1$ if the edge (i, j) has edge attribute p. $\mathbf{N}$ is a diagonal matrix, and $\mathbf{N}^q(i, i) = 1$ if node i has node attribute q. We use similar notations for tensors as described in [12]. For example, $\times_k$ denotes the k-mode product of tensors, and $\mathcal{X}_{(k)}$ denotes the mode-k unfolding. We define vec$(\cdot)$ and reshape$(\cdot, n_1, ..., n_K)$ as vectorizing operation and tensorizing operator. In general, we need to specify orders of vectorization and tensorization in tensor. By default, we let vec$(\cdot)$ and reshape$(\cdot, n_1, ..., n_K)$ be vectorizing and tensorizing operations along the *1-st mode*.[1] For example, vec$(\mathcal{X})$ = vec$(\mathcal{X}_{(1)})$, i.e., column-wise vectorization of the mode-1 unfolding of tensor $\mathcal{X}$. reshape$(\mathbf{x}, n_1, ..., n_K) = \mathcal{X}$ tensorizes vector $\mathbf{x}$ to a tensor of dimension $n_1 \times n_2 \times ... \times n_K$ by stacking the vectors of length n_1 as the mode-1 fibers of $\mathcal{X}$. For brevity, we assume that the K input networks share comparable numbers of nodes (i.e., $O(n_1) = ... = O(n_K) = O(n)$) and comparable numbers of edges (i.e., $O(m_1) = ... = O(m_K) = O(m)$). With these notations, we formally define the multi-way association tensor (Definition 1), the multi-way anchor link tensor (Definition 2), and the multi-way association inference problem (Problem 1) as follows.

[1] vec$_k(\cdot)$ will be used to denote vectorization along the k-th mode when necessary.

Table 1: Table of Symbols

Symbols	Definitions and Descriptions
$G = \{\mathbf{A}, \mathbf{E}, \mathbf{N}\}$	An attributed network
$\mathbf{A}, \tilde{\mathbf{A}}$	Original/normalized adjacency matrix
$\mathbf{E}$	Edge attribute matrix
$\mathbf{N}$	Node attribute matrix
$\mathcal{B}$	Multi-way anchor link tensor
$\mathcal{X}$	Multi-way association tensor
$\mathbf{x}, \mathbf{b}$	Vectorized tensor $\mathcal{X}$ and $\mathcal{B}$
K	Number of input networks
P, Q	Number of edge/node attributes
n_i, m_i	# of nodes/edges of $G_i (i = 1, ..., K)$
s	Rank of multi-way anchor link tensor
r	Rank used in eigen-decomposition
vec$(\cdot)$	Vectorization operator along 1-st mode
reshape$(\cdot, n_1, ..., n_K)$	Tensorization operator along 1-st mode
$\bigotimes_{i=1}^{K}$	Kronecker products from index 1 to K

Definition 1. *Multi-way association Tensor*
Given a set of K networks G_k $(k = 1, ..., K)$. A multi-way association is the association of a set of nodes $(i_1, i_2, ..., i_K)$, where $i_1 \in G_1, i_2 \in G_2, ..., i_K \in G_K$. The multi-way association tensor $\mathcal{X}$ is of size $n_K \times n_{K-1} \times ... \times n_1$. Each entry $\mathcal{X}(i_k, i_{k-1}, ..., i_1)$ indicates to what extent the corresponding nodes $(i_k, i_{k-1}, ..., i_1)$ are associated with each other.

Definition 2. *Multi-way anchor link tensor*
Given a set of K networks G_k $(k = 1, ..., K)$. The multi-way association prior tensor $\mathcal{B}$ is of size $n_K \times n_{K-1} \times ... \times n_1$. If the corresponding nodes in $(i_K, i_{K-1}, ..., i_1)$ are known to be associated with each other a priori, we call $(i_K, i_{K-1}, ..., i_1)$ an anchor link.

For the example in Fig. 1, only one multi-way anchor link (marked by the red rectangle) is known a priori, and thus there is only one non-zero entry in $\mathcal{B}$.[2] Each positive entry in $\mathcal{X}$ represents a multi-way association between a set of three users, one from each of the three input social networks. By the multi-way association inference algorithms that will be introduced in the subsequent sections, we might find that there are four positive entries in tensor $\mathcal{X}$ (e.g., $\mathcal{X}(1, 1, 1)$, $\mathcal{X}(2, 2, 2)$, $\mathcal{X}(3, 3, 3)$ and $\mathcal{X}(4, 4, 4)$). Each of these four entries indicates a set of three users who are either identical or similar with each other.

Problem 1. *Multi-way Association Inference*
Given: *A set of K networks $\{G_k \ (k = 1, ..., K)\}$, and a multi-way anchor link tensor $\mathcal{B}$ with dimension $n_K \times n_{K-1} \times ... \times n_1$;*
Output: *The multi-way association tensor $\mathcal{X}$, whose entries measure the strength of the association of the corresponding node sets.*

[2] For clarity, we assume that a sparse anchor link tensor $\mathcal{B}$ is given for the rest of this paper. Nonetheless, such an anchor link tensor is optional, and our proposed formulation, algorithms and analysis can be naturally adapted in the absence of $\mathcal{B}$. For example, if no multi-way association is available a prior, we can set $\mathcal{B}$ as a uniform tensor, i.e. each entry is equal to 1.

3 FORMULATION AND BASIC ALGORITHM

3.1 SyTE Formulation

We first formulate Problem 1 from the optimization perspective. The key idea is to generalize the *consistency* principle which was originally designed for pair-wise network alignment [32]. Given two sets of nodes $(i_K, ..., i_1)$ and $(j_K, ..., j_1)$. Intuitively, the consistency principle requires that the multi-way association of $(i_K, ..., i_1)$ be consistent with that of $(j_K, ..., j_1)$ (i.e., $\mathcal{X}(i_K, ..., i_1) \approx \mathcal{X}(j_K, ..., j_1)$), if the two node sets are consistent in terms of the following three aspects. This includes (1) *topology consistency*, meaning that the two node sets are strongly connected with each other (i.e., large $\mathbf{A}_k(i_k, j_k)$ $(k = 1, .., K)$); (2) *node attribute consistency*, meaning that nodes in each of the two sets share the same attributes respectively (i.e., $\mathbf{N}_1(i_1, i_1) = \cdots = \mathbf{N}_K(i_K, i_K)$ and $\mathbf{N}_1(j_1, j_1) = \cdots = \mathbf{N}_K(j_K, j_K)$); and (3) *edge attribute consistency*, meaning that the two node sets are connected with each other by the same edge attribute (i.e., $\mathbf{E}_1(i_1, j_1) = \cdots = \mathbf{E}_K(i_K, j_K)$). Formally, this leads to the following objective function.

$$J(\mathcal{X}) = \sum_{\substack{i_1,...,i_K \\ j_1,...,j_K}} [\beta(\frac{\mathcal{X}(i_K, ..., i_1)}{\sqrt{d(i_1, ..., i_K)}} - \frac{\mathcal{X}(j_K, ..., j_1)}{\sqrt{d(j_1, ..., j_K)}})^2 \underbrace{t(\mathbf{A}_1 ... \mathbf{A}_K)}_{\text{Topology consistency}} \times \underbrace{f(i_k) \times f(j_k)}_{\text{Node attribute consistency}} \times \underbrace{g(i_k, j_k)}_{\text{Edge attribute consistency}} + \underbrace{(1 - 2\beta)(\mathcal{X}(i_K, ..., i_1) - \mathcal{B}(i_K, ..., i_1))^2}_{\text{Anchor link regularizer}}] \quad (1)$$

where $\beta \in (0, 0.5)$ is a weighting parameter, and functions $f(\cdot)$ and $g(\cdot)$ denote the node and edge attribute consistency terms. The topology consistency term for all networks is $t(\mathbf{A}_1 ... \mathbf{A}_K) = \mathbf{A}_1(i_1, j_1) \cdots \mathbf{A}_K(i_K, j_K)$. Function $d(\cdot)$ denotes node-set normalization. They are defined as $f(i_k) = \mathbb{1}(\mathbf{N}_1(i_1, i_1) = \cdots = \mathbf{N}_K(i_K, i_K))$, $g(i_k, j_k) = \mathbb{1}(\mathbf{E}_1(i_1, j_1) = \cdots = \mathbf{E}_K(i_K, j_K))$, where $\mathbb{1}()$ is the indicator function. $d(i_1, ..., i_K) = \sum_{j_1,...,j_K} \mathbf{A}_1(i_1, j_1) \cdots \mathbf{A}_K(i_K, j_K)$, if $f(i_k) = f(j_k) = g(i_k, j_k) = 1$; otherwise $d(i_1, ..., i_K) = 1$. By using the notation defined in Section 2, the consistency terms $f(\cdot)$ and $g(\cdot)$ can be re-written as $f(i_k) = \sum_{p=1}^{P} \mathbf{N}_1^p(i_1, i_1) \cdots \mathbf{N}_K^p(i_K, i_K)$, and $g(i_k, j_k) = \sum_{q=1}^{Q} \mathbf{E}_1^q(i_1, j_1) \cdots \mathbf{E}_K^q(i_K, j_K)$.

By vectorizing the multi-way association tensor, the objective function in Eq. (1) can be re-written in a more concise form. First, let the tensor indices $i_K, i_{K-1}, ...i_1$ and $j_K, j_{K-1}, ..., j_1$ become vector indices u and v respectively, where $u = \sum_{k=1}^{K-1} \prod_{j=k+1}^{K} n_j(i_k - 1) + i_K$, and $v = \sum_{k=1}^{K-1} \prod_{j=k+1}^{K} n_j(j_k - 1) + j_K$. Then, note that $d(\cdot)$ in Eq. (1) actually calculates the degree matrix of a Kronecker graph formed by $\mathbf{A}_1, ..., \mathbf{A}_K$ after filtered by the node and edge attributes. Specifically, let: $\mathbf{W} = \sum_{i,j,k} \mathbf{N}_1^i(\mathbf{E}_1^k \odot \mathbf{A}_1)\mathbf{N}_1^j \otimes \cdots \otimes \mathbf{N}_K^i(\mathbf{E}_K^k \odot \mathbf{A}_K)\mathbf{N}_K^j = \mathbf{N}(\mathbf{E} \odot (\mathbf{A}_1 \otimes \ldots \otimes \mathbf{A}_K))\mathbf{N}$ where $\mathbf{N} = \sum_{i=1}^{P} \mathbf{N}_1^i \otimes \cdots \otimes \mathbf{N}_K^i$, $\mathbf{E} = \sum_{k=1}^{Q} \mathbf{E}_1^k \otimes \cdots \otimes \mathbf{E}_K^k$, and $\odot$ is the element-wise product. Let $\mathbf{D}$ be the diagonal degree matrix of $\mathbf{W}$ such that $\mathbf{D}(u, u) = \sum_v \mathbf{W}(u, v)$, and let $\mathbf{x} = \text{vec}(\mathcal{X})$, $\mathbf{b} = \text{vec}(\mathcal{B})$. Plugging all these into Eq. (1), we have the following objective function to minimize w.r.t. multi-way association vector $\mathbf{x}$.

$$\arg\min_{\mathbf{x}} J(\mathbf{x}) = \alpha \mathbf{x}^\top (\mathbf{I} - \widetilde{\mathbf{W}})\mathbf{x} + (1 - \alpha)||\mathbf{x} - \mathbf{b}||_2^2 \quad (2)$$

where $\widetilde{\mathbf{W}} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$. The first term in Eq. (2) is equivalent to the first term of summation in Eq. (1) for consistency principles, and the second term is to encode the prior knowledge of anchor links. The weighting parameter is reset as $0 < \alpha < 1$ which balances the consistency objective and prior knowledge of anchor links. It can be shown that $\alpha = 2\beta$.

3.2 Basic Algorithm

In the appendix, we show that Eq. (2) is a convex problem. Thus, its fixed point solution gives the optimal solution of Eq. (2). By taking the derivative of $J(\mathbf{x})$ in Eq. (2) w.r.t. $\mathbf{x}$ and setting it to zero, we have

$$(\mathbf{I} - \alpha\widetilde{\mathbf{W}})\mathbf{x} - (1 - \alpha)\mathbf{b} = \mathbf{0} \Rightarrow \mathbf{x} = \alpha\widetilde{\mathbf{W}}\mathbf{x} + (1 - \alpha)\mathbf{b} \quad (3)$$

By grouping the Kronecker products of K normalized adjacency matrices in $\widetilde{\mathbf{W}}$ into two parts $\mathbf{A}_f = \mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_k$, $\mathbf{A}_s = \mathbf{A}_{k+1} \otimes \cdots \otimes \mathbf{A}_K$, $k \in (1, K)$, the fixed point iteration can be seen as the procedure on two input networks with adjacency matrices $\mathbf{A}_f$ and $\mathbf{A}_s$ respectively. We transform the Kronecker products in $\widetilde{\mathbf{W}}$ to matrix products $(\mathbf{A}_f \otimes \mathbf{A}_s)\mathbf{v} = \mathbf{A}_s\mathbf{V}\mathbf{A}_f^\top$, where $\mathbf{v} = \text{vec}(\mathbf{V}) = \mathbf{N}\mathbf{D}^{-1/2}\mathbf{x}$. Thus the fixed point method (referred to as Basic Algorithm) can be applied to obtain the solution $\mathbf{x}$. The details of Basic Algorithm, together with its convergence, optimality and complexity analysis are given in the appendix. In a nutshell, we can show that the Basic Algorithm (summarized in Alg. 3 in the appendix) converges to the closed-form solution of Eq. (3) with a polynomial time complexity w.r.t. the number of nodes and edges of the input networks. Note that Basic Algorithm is not the major focus of this paper because of its high complexity.

3.3 Sylvester Tensor Equation

In order to speed-up and scale-up the computation, we reformulate Eq. (3) as follows. Note that Eq. (3) is a linear system w.r.t. vector variable $\mathbf{x}$. Considering the definition of $\mathbf{W}$, the linear system in Eq. (3) can be re-written as the following *Sylvester tensor equation* by the property of tensor mode product and Kronecker product (i.e., $\mathcal{X} \times_1 \tilde{\mathbf{A}}_K \times_2 \cdots \times_K \tilde{\mathbf{A}}_1 \Leftrightarrow (\tilde{\mathbf{A}}_1 \otimes \cdots \otimes \tilde{\mathbf{A}}_K)\text{vec}(\mathcal{X})$ [12]).

$$\mathcal{X} - \alpha \sum_{o,p,q} \mathcal{X} \times_1 \tilde{\mathbf{A}}_K^{(o,p,q)} \times_2 \cdots \times_K \tilde{\mathbf{A}}_1^{(o,p,q)} - (1 - \alpha)\mathcal{B} = \mathbf{0} \quad (4)$$

where $\tilde{\mathbf{A}}_i^{(o,p,q)} = (\mathbf{D}_i^{-1/2}\mathbf{N}_i^p)(\mathbf{E}_i^o \odot \mathbf{A}_i)(\mathbf{D}_i^{-1/2}\mathbf{N}_i^q)$. When neither node nor edge attributes are available (i.e., plain networks), Eq. (4) degenerates into the following Sylvester tensor equation.

$$\mathcal{X} - \alpha\mathcal{X} \times_1 \tilde{\mathbf{A}}_K \times_2 \cdots \times_K \tilde{\mathbf{A}}_1 - (1 - \alpha)\mathcal{B} = \mathbf{0} \quad (5)$$

where $\tilde{\mathbf{A}}_i = (\mathbf{D}_i^{-1/2})\mathbf{A}_i(\mathbf{D}_i^{-1/2})$.

In the next two sections, we will present two fast algorithms to solve Eq. (5) and Eq. (4), respectively.

4 SYTE-FAST-P FOR PLAIN NETWORKS

In this section, we present the proposed fast algorithm, SyTE-Fast-P for solving Eq. (5) (i.e., plain networks).

4.1 Intuitions and Key Ideas

In order to solve Eq. (5) efficiently, the key idea is to decompose its corresponding linear system into a series of subsystems, and then to solve each of them by a tensorized Krylov subspace method. To

simplify the description, we absorb the scalar α and $1-\alpha$ of Eq. (3) into the matrix $\widetilde{\mathbf{W}}$ and vector $\mathbf{b}$ respectively to have the following concise equation.

$$(\mathbf{I} - \tilde{\mathbf{A}}_1 \otimes \cdots \otimes \tilde{\mathbf{A}}_K)\mathbf{x} = \mathbf{b} \tag{6}$$

where we let $\mathbf{b} \coloneqq -(1-\alpha)\mathbf{b}$, and $\tilde{\mathbf{A}}_i \coloneqq \alpha^{1/K}\tilde{\mathbf{A}}_i, \forall i \in [1, K]$. The coefficient matrix of this linear system is strictly diagonally dominant and positive definite. Therefore, the system is solvable with a unique solution [22]. However, the coefficient matrix in this linear system contains the Kronecker product. A direct solver such as Krylov subspace method would cost at least $O(N^2)$ where $N = n^K >> n$ is the dimension of the linear system in Eq. (6), due to the construction of the orthonormal basis of the Krylov subspace [7]. To address this issue, the key idea is to use a tensorized Krylov subspace, and let the solution tensor reside in the tensorized Krylov subspace, so as to avoid explicit calculation of the orthonormal basis. In this way, the solution can be represented in a Tucker decomposition form without the explicit calculation, which will reduce the time complexity from $O(N^2)$ to $O(c_1 m + c_2 l^K)$ and reduce the space complexity from $O(N + M)$ to $O(c_3 n + Km + l^{2K})$. Here, c_1, c_2, c_3, l are small constants, and K can also be regarded as a constant because it is often much smaller than both n and m (see details below).

4.2 SyTE-Fast-P Algorithm

Firstly, notice that tensor $\boldsymbol{\mathcal{B}}$ contains s non-zero entries (i.e., s known anchor links). We represent $\boldsymbol{\mathcal{B}}$ as the summation of the outer product of the indicator vectors $\boldsymbol{\mathcal{B}} = \sum_{i=1}^{s} \mathbf{b}_K^{(i)} \circ \cdots \circ \mathbf{b}_1^{(i)}$ where $\mathbf{b}_j^{(i)}$ is an indicator vector of the i−th non-zero entry in $\boldsymbol{\mathcal{B}}$ corresponding to mode j. Tensor $\boldsymbol{\mathcal{B}}$ can be further re-written as its vector form $\mathbf{b} = \sum_{i=1}^{s} \mathbf{b}_1^{(i)} \otimes \cdots \otimes \mathbf{b}_K^{(i)}$. For the example in Fig. 1, the anchor link tensor $\boldsymbol{\mathcal{B}}$ only contains one non-zero entry at (4, 4, 4). It can be shown that $\boldsymbol{\mathcal{B}} = [0,0,0,1]^\top \circ [0,0,0,1]^\top \circ [0,0,0,1]^\top$, and $\mathbf{b} = [0,0,0,1]^\top \otimes [0,0,0,1]^\top \otimes [0,0,0,1]^\top$. Then, Eq. (6) is decomposed into the following subsystems.

$$(\mathbf{I} - \tilde{\mathbf{A}}_1 \otimes \cdots \otimes \tilde{\mathbf{A}}_K)\mathbf{x}_i = \bigotimes_{j=1}^{K} \mathbf{b}_j^{(i)} \ (i = 1, ..., s) \tag{7}$$

Secondly, instead of directly constructing the Krylov subspace $\mathcal{K}_L(\mathbf{A}_\times, \mathbf{b})$ where $\mathbf{A}_\times$ denotes $\mathbf{I} - \tilde{\mathbf{A}}_1 \otimes \cdots \otimes \tilde{\mathbf{A}}_K$, and L is the dimension of the Krylov subspace, we construct the Krylov subspaces for $\tilde{\mathbf{A}}_1, ..., \tilde{\mathbf{A}}_K$ separately as $\mathcal{K}_l(\tilde{\mathbf{A}}_1, \mathbf{b}_1), ..., \mathcal{K}_l(\tilde{\mathbf{A}}_K, \mathbf{b}_K)$. Then, we use the Kronecker product of these K Krylov subspaces as the new subspace. Using the same notation as [14], we define the tensorized Krylov subspace as follows.

$$\mathcal{K}_{\mathfrak{L}}^{\otimes}(\mathbf{A}_\times, \mathbf{b}) \coloneqq span(\mathcal{K}_{l_1}(\tilde{\mathbf{A}}_1, \mathbf{b}_1) \otimes \cdots \otimes \mathcal{K}_{l_K}(\tilde{\mathbf{A}}_K, \mathbf{b}_K)) \tag{8}$$

where $\mathfrak{L} = (l_1, ..., l_K)$ is a multi-index. For each Krylov subspace $\mathcal{K}_{l_i}(\tilde{\mathbf{A}}_i, \mathbf{b}_i)$, we use the standard Arnoldi method[3] to obtain the orthonormal basis represented in $\mathbf{U}$ with orthonormal columns. Specifically, the Arnoldi method gives the Hessenberg matrix $\tilde{\mathbf{H}}_i = \mathbf{U}_{l_i+1}^\top \tilde{\mathbf{A}}_i \mathbf{U}_{l_i}$ of size $(l_i + 1) \times l_i$. Note that l_i is often much smaller than n_i (i.e., $l_i \ll n_i$). We can prove that (1) $\otimes_{i=1}^{K} \mathbf{U}_{l_i}$ forms the orthonormal basis of $\mathcal{K}_{\mathfrak{L}}^{\otimes}(\mathbf{A}_\times, \mathbf{b})$, and (2) the original Krylov subspace $\mathcal{K}_L(\mathbf{A}_\times, \mathbf{b})$ is contained in the tensorized Krylov subspace $\mathcal{K}_{\mathfrak{L}}^{\otimes}(\mathbf{A}_\times, \mathbf{b})$ (see details in the appendix).

[3]Lanczos algorithm could also be adopted here alternatively.

Thirdly, we can further prove that by using a tensorized Krylov subspace based generalized minimal residual method, each subsystem $(\mathbf{I} - \tilde{\mathbf{A}}_1 \otimes \cdots \otimes \tilde{\mathbf{A}}_K)\mathbf{x}_i = \otimes_{j=1}^{K} \mathbf{b}_j^{(i)}$ in Eq. (7) can be solved by the following linear system, whose scale is much smaller than the original linear system:

$$(\bigotimes_{i=1}^{K} \mathbf{I}_{l_i+1,l_i} - \bigotimes_{i=1}^{K} \tilde{\mathbf{H}}_i)\mathbf{y} = \bigotimes_{i=1}^{K} \mathbf{U}_{l_i+1}^\top \mathbf{r}_0 \tag{9}$$

Note that the coefficient matrix has a Hessenberg-like structure. Thus, it can be solved by the back-substitute method. Putting everything together, the proposed SyTE-Fast-P algorithm is presented in Algorithm 1. We use the same dimension for all the Krylov subspaces for notation simplicity. Note that $\mathbf{x} = \mathbf{x}_1 + \cdots + \mathbf{x}_K$, where each $\mathbf{x}_i = \otimes_{j=1}^{K} \mathbf{U}_{l_j}^{(i)} \mathbf{y}_i$, and $\boldsymbol{\mathcal{X}} = \text{reshape}(\mathbf{x}, n_K, ..., n_1)$.

Algorithm 1 SyTE-Fast-P Algorithm

Input: K normalized adjacency matrices of input networks $\tilde{\mathbf{A}}_1$, ..., $\tilde{\mathbf{A}}_K$, tensor $\boldsymbol{\mathcal{B}}$ or $\mathbf{b} = \text{vec}(\boldsymbol{\mathcal{B}})$ with s known multi-way anchor links, Krylov subspace size $l > 0$;

Output: The solution tensor $\boldsymbol{\mathcal{X}}$ of Eq. (6).

```
Decompose b for Eq. (7) to obtain {b_j^(i)}_{j∈[1,K],i∈[1,s]};
for i = 1, ..., s do
    Initialize x_i as a zero vector;
    for j = 1, ..., K do
        Construct K_l(I_j − Ã_j, b_j);
        Obtain H̃_j^(i), U_{l_j}^(i), and U_{l_j+1}^(i);
    end for
    Solve Eq. (9) to obtain y_i;
end for
Return implicit solution {y_i, {U_{l_j}^(i)}_{j=1}^K}_{i=1}^s.
```

4.3 Proofs and Analysis

We give the following theorem for the complexity of the proposed SyTE-Fast-P algorithm. In the analysis of complexity, for notation simplicity, we assume all input networks share the same number of nodes n and number of edges m.

Theorem 1. ***Complexity of SyTE-Fast-P.*** *The time complexity of Algorithm 1 is $O(sKlm + sl^K)$. The space complexity of Algorithm 1 is $O(Km + l^{2K} + Kln)$.*

Proof. The Arnoldi process takes $O(lm)$ for one iteration, and the number of total iteration is sK. Thus the complexity for Arnoldi process is $O(sKlm)$. Solving s linear systems of Eq. (9) takes $O(sl^K)$, which is linear w.r.t. the size of the linear system, thanks to the back-substitute method. The overall time complexity for SyTE-Fast-P is $O(sKlm + sl^K)$.

For space complexity, storing K adjacency matrices takes $O(Km)$. Storing the Hessenberg matrix for each Eq. (9) in the outer iteration takes $O(l^{2K})$. Storing the orthonomal basis $\mathbf{U}_{l_j}^{(i)}$ for the outer iteration takes $O(Kln)$. Overall, the space complexity is $O(Km+l^{2K}+Kln)$ for SyTE-Fast-P. □

Remark. Since K, l, s are usually much smaller than m, n (K, l, s are treated as *constants*[4] in the big-O notation), Alg. 1 has a much smaller time complexity than the Basic Algorithm (Alg. 3) whose

[4]We assume that the number of input networks K is a small non-variant constant.

time complexity is $O(Qm^{\lfloor K/2 \rfloor} n^{\lfloor K/2 \rfloor} \cdot t_{max} + n^K)$. Furthermore, both the space and time complexities of Alg. 1 is *linear* w.r.t. the size (i.e., the number of nodes and edges) of input networks.

5 SYTE-FAST-A FOR ATTRIBUTED NETWORKS

In this section, we present a fast algorithm to solve Eq. (4).

5.1 Intuitions and Key Ideas

Here, we present a fast solver when the node attributes are available. Notice that SyTE-Fast-P can not be directly applied because the $\mathbf{W}$ matrix in Eq. (2) contains summations of Kronecker products. To address this issue, we have the following key observations.

Firstly, WLOG, assume that nodes in each network are reordered such that the nodes with the same node attributes have adjacent indices. For the example in Fig. 1, assume that nodes $\{1, 2\}$ and $\{3, 4\}$ in $G_i, \forall i \in [1, 3]$ have the same attributes respectively. We observe that the solution tensor of Eq. (4) has a block-diagonal structure, which means that the non-zero entries in the solution tensor $\boldsymbol{\mathcal{X}}$ only exist in the diagonal block tensors. Intuitively, this is because the diagonal blocks in the solution tensor correspond to nodes across networks with the same node attributes, meanwhile the off-diagonal blocks correspond to nodes across networks with different node attributes. This indicates that Eq. (4) could be decomposed into a series of subsystems by node attributes.

Secondly, based on the above observation, we only need to solve the diagonal tensors by block coordinate descent (*BCD*) method. The off-diagonal entries in $\boldsymbol{\mathcal{X}}$ can be set equal to the corresponding entries with the same indices in $\boldsymbol{\mathcal{B}}$. The linear system for the example in Fig. 1 can be decomposed into:

$$\boldsymbol{\mathcal{X}}^{1,1,1} - [\boldsymbol{\mathcal{X}}^{1,1,1} \times_1 \tilde{\mathbf{A}}_1^{11} \dots \times_3 \tilde{\mathbf{A}}_3^{11} + \underbrace{\boldsymbol{\mathcal{X}}^{2,2,2} \times_1 \tilde{\mathbf{A}}_1^{12} \dots \times_3 \tilde{\mathbf{A}}_3^{12}}_{\boldsymbol{\mathcal{C}}_{2,2,2}^{1,1,1}}] = \boldsymbol{\mathcal{B}}^{1,1,1}$$

$$\boldsymbol{\mathcal{X}}^{2,2,2} - [\boldsymbol{\mathcal{X}}^{2,2,2} \times_1 \tilde{\mathbf{A}}_1^{22} \dots \times_3 \tilde{\mathbf{A}}_3^{22} + \underbrace{\boldsymbol{\mathcal{X}}^{1,1,1} \times_1 \tilde{\mathbf{A}}_1^{21} \dots \times_3 \tilde{\mathbf{A}}_3^{21}}_{\boldsymbol{\mathcal{C}}_{1,1,1}^{2,2,2}}] = \boldsymbol{\mathcal{B}}^{2,2,2}$$

with $\boldsymbol{\mathcal{X}}^{ijk} = \boldsymbol{\mathcal{B}}^{ijk}, \forall i \neq j$ or $j \neq k$ or $i \neq k$. $\boldsymbol{\mathcal{X}}^{1,1,1}$ and $\boldsymbol{\mathcal{X}}^{2,2,2}$ denote the $(1, 1, 1)-$th and $(2, 2, 2)-$th tensor block respectively (similar notation for $\boldsymbol{\mathcal{B}}$). $\tilde{\mathbf{A}}^{11} = \mathbf{D}^{-1/2}\mathbf{N}^1\mathbf{A}\mathbf{N}^1\mathbf{D}^{-1/2}$ denotes the normalized adjacency matrix, filtered by the first node attribute. The parameter α, together with $(1 - \alpha)$ on $\boldsymbol{\mathcal{B}}$, is absorbed into tensors for notation simplicity. $\boldsymbol{\mathcal{C}}_{1,1,1}^{2,2,2}$ represents the contribution of block variable $\boldsymbol{\mathcal{X}}^{1,1,1}$ to $\boldsymbol{\mathcal{X}}^{2,2,2}$, and the similar notation applies for $\boldsymbol{\mathcal{C}}_{2,2,2}^{1,1,1}$.

Thirdly, *BCD* requires that each $\boldsymbol{\mathcal{X}}^{i\dots i}$ be computed explicitly (e.g., by the Basic Algorithm), because each block variable is needed to calculate other block variables. Although *BCD* is faster than applying the Basic Algorithm on the whole variable tensor, the computational complexity is still polynomial. To address this issue, the idea is to omit the contribution of diagonal blocks from other subsystems (e.g. $\boldsymbol{\mathcal{C}}_{1,1,1}^{2,2,2}$ and $\boldsymbol{\mathcal{C}}_{2,2,2}^{1,1,1}$) for each subsystem. In this way, we only need one single iteration by Alg. 1 to approximately solve each diagonal block independently.

5.2 SyTE-Fast-A Algorithm

Generally speaking, Eq. (4) with node attributes can be decomposed as: $\boldsymbol{\mathcal{X}}^{i\dots i} - \sum_{j=1} \boldsymbol{\mathcal{X}}^{j\dots j} \times_1 \tilde{\mathbf{A}}_1^{ij} \dots \times_K \tilde{\mathbf{A}}_K^{ij} = \boldsymbol{\mathcal{B}}^{i\dots i}$, where the off-diagonal variant blocks are equal to the corresponding blocks in $\boldsymbol{\mathcal{B}}$. It can be solved by approximated block coordinate descent as follows. By the exact *BCD*, each iteration should solve:

$$\boldsymbol{\mathcal{X}}^{i\dots i} - \boldsymbol{\mathcal{X}}^{i\dots i} \times_1 \tilde{\mathbf{A}}_1^{ii} \dots \times_K \tilde{\mathbf{A}}_K^{ii} = \hat{\boldsymbol{\mathcal{B}}}^{i\dots i} \tag{11}$$

where $\hat{\boldsymbol{\mathcal{B}}}^{i\dots i} = \boldsymbol{\mathcal{B}}^{i\dots i} + \sum_{j\neq i} \boldsymbol{\mathcal{X}}^{j\dots j} \times_1 \tilde{\mathbf{A}}_1^{ij} \dots \times_K \tilde{\mathbf{A}}_K^{ij}$, and it can be viewed as the updated $\boldsymbol{\mathcal{B}}^{i\dots i}$ tensor. If we approximate $\hat{\boldsymbol{\mathcal{B}}}^{i\dots i} = \boldsymbol{\mathcal{B}}^{i\dots i}$, each subsystem can be solved in one single iteration. The SyTE-Fast-A algorithm is summarized in Alg. 2. Similar to Alg. 1, line 6 returns the *implicit* solution of diagonal block variables.

Algorithm 2 SyTE-Fast-A Algorithm

Input: Normalized adjacency matrices $\tilde{\mathbf{A}}_1, \dots, \tilde{\mathbf{A}}_K$; node attribute matrices $\mathbf{N}_1, \dots, \mathbf{N}_K$; Krylov subspace size $l > 0$; tensor $\boldsymbol{\mathcal{B}}$ or $\mathbf{b} = \text{vec}(\boldsymbol{\mathcal{B}})$;

Output: The solution tensor $\boldsymbol{\mathcal{X}}$ of Equation (4).

1: Construct block matrices $\tilde{\mathbf{A}}_1^{ij}, \dots, \tilde{\mathbf{A}}_K^{ij}$, block tensor $\boldsymbol{\mathcal{B}}^{i\dots i}, \forall 1 \leq i, j \leq P$ by the node attribute matrices $\mathbf{N}_1, \dots, \mathbf{N}_K$;
2: Initialize $\boldsymbol{\mathcal{X}}^{i\dots i}, \forall i \in [1, K]$;
3: **for** $p = 1, \dots, P$ **do**
4: Solve Eq. (11) by Algorithm 1 to obtain $\{\mathbf{y}_i, \{\mathbf{U}_{l_j}^i\}_{j=1}^K\}_{i=1}^s$;
5: **end for**
6: Return implicit solution $\{\{\mathbf{y}_i^p, \{\mathbf{U}_{l_j}^{i,p}\}_{j=1}^K\}_{i=1}^s\}_{p=1}^P$.

5.3 Proofs and Analysis

Next, we provide the complexity analysis of the proposed algorithm. Let m_i, n_i and s_i ($i \in [1, P]$) be the number of edges/nodes in $\tilde{\mathbf{A}}_k^{ii}$ ($k \in [1, K]$), and the number of non-zero entries in $\boldsymbol{\mathcal{B}}^{i\dots i}$, respectively. For notation simplicity, we assume $m_i, \forall i \in [1, P]$ is the same for each input network. Let l be the subspace size when using Algorithm 1 in solving Eq. (11).

Theorem 2. ***Complexity of SyTE-Fast-A.*** *The time complexity of Algorithm 2 is $O(Km+n+\sum_{i=1}^P (s_i K l m_i + s_i l^K)$. The space complexity of Algorithm 2 is $O(PKm_i + Kln_i + l^{2K})$.*

Proof. Constructing $\tilde{\mathbf{A}}_1^{ii}, \dots, \tilde{\mathbf{A}}_K^{ii}, \forall 1 \leq j \leq P$ takes $O(Km)$. Calculating $\boldsymbol{\mathcal{B}}^{i\dots i}$ takes $O(n)$ because the tensor $\boldsymbol{\mathcal{B}}$ is rank-n. Solving P equations of Eq. (11) by using SyTE-Fast-P takes $O(\sum_{i=1}^P (s_i K l m_i + s_i l^K))$. Overall, the time complexity of SyTE-Fast-A is $O(Km + n + \sum_{i=1}^P (s_i K l m_i + s_i l^K))$.

For space complexity, storing $\tilde{\mathbf{A}}_1^{ii}, \dots, \tilde{\mathbf{A}}_K^{ii}$ takes $O(PKm_i)$. Since in the iteration only one Eq. (11) is solved each time, storing Hessenberg matrices and orthonomal basis takes $O(Kln_i + l^{2K})$. The overall space complexity for SyTE-Fast-A is $O(PKm_i + Kln_i + l^{2K})$. □

Remark. From Theorem 2, note that the number of node attributes has great impact on time and space complexity, since $\sum_{i=1}^P m_i = m$. A larger number of node attribute will lead to smaller complexity for computing each block tensor variable. Also note that the time complexity is much less than the basic method, which is $O(N)$.

6 SENSITIVITY ANALYSIS

In this section, we analyze the sensitivity of the linear system formulated in Eq. (3). To be specific, we aim to understand how the solution of Eq. (3) will be impacted if some edges of the input networks are changed, due to either random noise or adversarial attacks (e.g., edge removal [35]). Given K networks $G_1, \dots, G_K$ and

the budget for edge perturbation in each network $p_1, ..., p_K$. Let $\tilde{\mathbf{A}} = \mathbf{I} - \tilde{\mathbf{A}}_1 \otimes \cdots \otimes \tilde{\mathbf{A}}_K$, we have:

$$(\tilde{\mathbf{A}} + \Delta\tilde{\mathbf{A}})(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} \tag{12}$$

where $\Delta\tilde{\mathbf{A}}$ and $\Delta\mathbf{x}$ are the perturbations to $\tilde{\mathbf{A}}$ and solution $\mathbf{x}$ respectively. We present the following theorem:

Theorem 3. *The relative change after edge perturbation satisfies:*

$$\frac{||\Delta\mathbf{x}||}{||\mathbf{x}||} \leq \frac{\eta}{1-\epsilon}(\prod_{i=1}^{K} m_i - \prod_{i=1}^{K}(m_i - 2p_i))^{1/2} \tag{13}$$

where $||\tilde{\mathbf{A}}_\times||_F = \epsilon < 1$, $\eta = ||\mathbf{D}^{-1/2}||_F^2$. m_i *is the number of edges in network* G_i.

Proof. Based on Eq. (3), we have $\Delta\mathbf{x} \approx -\tilde{\mathbf{A}}^{-1}\Delta\tilde{\mathbf{A}}\mathbf{x}$ by dropping the high-order small term $\Delta\tilde{\mathbf{A}}\Delta\mathbf{x}$. The relative change after edge removal is as follows:

$$\frac{||\Delta\mathbf{x}||}{||\mathbf{x}||} = \frac{||\tilde{\mathbf{A}}^{-1}\Delta\tilde{\mathbf{A}}\mathbf{x}||}{||\mathbf{x}||} \leq \frac{||\tilde{\mathbf{A}}^{-1}||\,||\Delta\tilde{\mathbf{A}}||\,||\mathbf{x}||}{||\mathbf{x}||} = \kappa(\tilde{\mathbf{A}})\frac{||\Delta\tilde{\mathbf{A}}||}{||\tilde{\mathbf{A}}||} \tag{14}$$

where $\kappa(\tilde{\mathbf{A}}) = ||\tilde{\mathbf{A}}||\,||\tilde{\mathbf{A}}^{-1}||$ is the condition number of matrix $\tilde{\mathbf{A}}$. Note that $\tilde{\mathbf{A}}$ is invertible since it is nonsingular. Since $||\tilde{\mathbf{A}}||_\infty < 1$, we have $\tilde{\mathbf{A}}^{-1} = \sum_{j=0}^{\infty}(\tilde{\mathbf{A}}_\times)^j$. Therefore,

$$||\tilde{\mathbf{A}}^{-1}||_F \leq \sum_{j=0}^{\infty}\epsilon^j = \frac{1}{1-\epsilon} \tag{15}$$

$||\Delta\tilde{\mathbf{A}}|| = ||\tilde{\mathbf{A}}_\times - \tilde{\mathbf{A}}'_\times|| = ||\mathbf{D}^{-1/2}(\mathbf{A}_\times - \mathbf{A}'_\times)\mathbf{D}^{-1/2}||$, where $\tilde{\mathbf{A}}'_\times$ and $\mathbf{A}'_\times$ are perturbed $\tilde{\mathbf{A}}_\times$ and $\mathbf{A}_\times$, respectively. Thus we have:

$$||\Delta\tilde{\mathbf{A}}|| \leq ||\mathbf{D}^{-1/2}||_F^2||\tilde{\mathbf{A}}_\times - \tilde{\mathbf{A}}'_\times|| = \eta(\prod_{i=1}^{K} m_i - \prod_{i=1}^{K}(m_i - 2p_i))^{1/2} \tag{16}$$

where $\prod_{i=1}^{K} m_i - \prod_{i=1}^{K}(m_i - 2p_i)$ is the number of non-zero entries in $\Delta\tilde{\mathbf{A}}$. Plugging Eq. (15) (16) into Eq. (14) leads to Eq. (13). □

From Theorem 3, we can see that the relative change of the solution $\mathbf{x}$ is bounded by the norm of the perturbed matrix $\Delta\tilde{\mathbf{A}}$. One implication for adversarial attacking (e.g., removing certain edges to maximally alter the solution $\mathbf{x}$) of this bound is as follows. Intuitively, a good attacking strategy might be to remove the edges in each $\tilde{\mathbf{A}}_i$ with high weights, since this will lead to the largest relative norm change (i.e., a larger upper bound in Theorem 3).

7 EXPERIMENTS

In this section we present the experimental results to answer the following questions:

- **Q1 Effectiveness.** How effective and accurate are the proposed SyTE methods for inferring multi-way association?
- **Q2 Efficiency.** How fast and scalable are the proposed SyTE methods?

7.1 Experimental Setup

Datasets. We use five datasets for evaluations whose statistics is summarized in Table 2 and details can be found in the appendix.

Comparison methods. In total, we evaluate 12 methods, including the proposed SyTE algorithms. For one-to-one multi-network alignment, we compare with *CLF* [30], *FINAL* [32] and *IsoRank* [25]. For multi-network node retrieval, we compare with *REGAL* [10], *CrossMNA* [6], *FINAL* [32], and *IsoRank* [25]. For high-order recommendation, we compare with *nNTF* (non-negative tensor factorization), *NTF* (Neural Tensor Factorization) [27], and *wiZAN-Dual* (Dual-Regularized One-Class Collaborative Filtering) [29]. For scalability study, we compare the proposed fast methods SyTE-*Fast-P* and SyTE-*Fast-A* with two classic Sylvester equation solvers, including *FP* (Fixed Point method) and *CG* (Conjugate Gradient method) [22].

Table 2: Datasets Summary

Dataset Name	Category	# of Nodes	# of Edges
DBLP	Co-authorship	1,013	3,244
Arxiv	Academic network	2,908	3,551
Douban	User relationship	3,384	6,556
Aminer	Academic network	1,274,360	4,756,194
Dataset Name	**# of Users**	**# of Artists**	**# of Tags**
LastFm	15,154	2,982	4,144

Evaluation Tasks. We design the following tasks for evaluations.

Task 1. Multi-network alignment. We first conduct multi-network alignment on three networks, which is different from traditional pair-wise network alignment. We use the following datasets. Three networks from *Arxiv* (two physical domains and one mathematical domain), three networks from *DBLP* (the original *DBLP* network and two permutated *DBLP* networks with 5% randomly added edge noises), and three networks from *Douban* (two *Douban* online networks with following and messaging relation respectively and one offline network).

Task 2. Multi-network node retrieval. In order to compare with some multi-network alignment baseline methods which do not support one-to-one alignment, we design a multi-network node retrieval experiment which is often referred to as 'soft alignment' [6, 10]. Given three networks and a node from one network, the goal of multi-network node retrieval is to return a ranking tuple list such that the similar nodes from other networks would appear in high ranks of the tuple list.

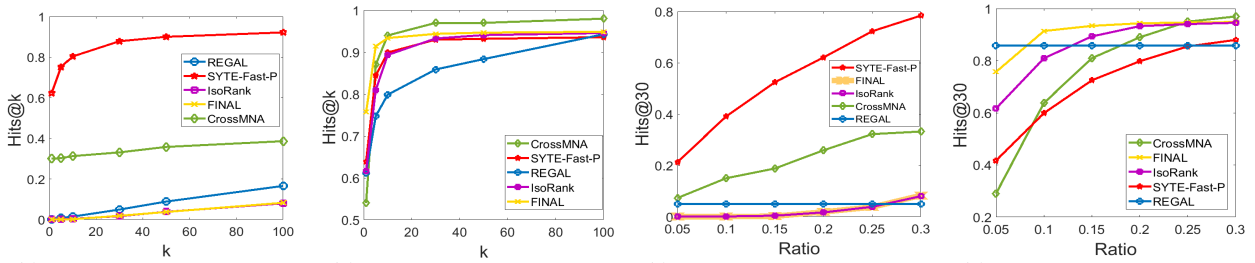
Task 3. High-order recommendation. To further evaluate the effectiveness of multi-way association, we conduct high-order recommendation. Traditional recommendation only recommend items to users, but we conduct high-order recommendation task to recommend tuples with (artist, tag) to users simultaneously on *LastFm*. The original dataset contains user-user interaction network, the (user, artist, tag) tuples reflecting the user's listening behavior, the (artist, tag) tuples reflecting the categories of artists, and (user, artist) tuple reflecting the users' artist preference. The artist-artist network is constructed by the the artists' cosine similarities calculated from (user, artist) tuples. The tag-tag network is constructed by calculating the cosine similarities of tags pairs in the (artist, tag) tuples. Note that the attributes are not used in this task.

The source code is available in this link[5]. The details of datasets, data preprocessing, and baseline methods are in the appendix.

7.2 Effectiveness Results

First, we present the experimental results of the multi-network alignment task. We focus on one-to-one alignment in this experiment. In order to obtain the one-to-one mapping, we implement a high-order greedy match algorithm to convert the multi-way association solution tensor $\boldsymbol{\mathcal{X}}$ to a matching tensor $\boldsymbol{\mathcal{M}}$, in which

[5]https://drive.google.com/drive/folders/1Bu72H7_0TpPFNrefkc6E8BWXoa4wc7f7?usp=sharing

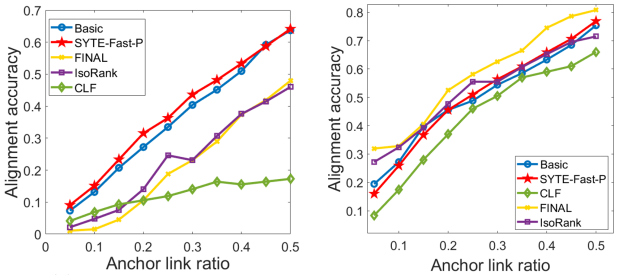


(a) Hits@k vs. k with *high-order* metric. Ratio of known anchor links: 0.3. **(b) Hits@k vs. k with *pair-wise* metric. Ratio of known anchor links: 0.3.** **(c) Hits@30 vs. ratio of known anchor links. *High-order* metric.** **(d) Hits@30 vs. ratio of known anchor links. *Pair-wise* metric.**

Figure 2: Cross-network node retrieval results on *DBLP* dataset. Higher is better. Best viewed in color.[6]

each fiber (e.g. $\mathcal{M}(i_K, i_{K-1}, ..., i_2, :)$) contains at most one non-zero entry to indicate a one-to-one alignment.

We use two metrics to evaluate the effectiveness. First, for a given one-to-one alignment tuple of nodes (e.g. $(u_1, ..., u_K)$, $u_1 \in G_1, ..., u_K \in G_k$), we consider it as a successful alignment iff *all* nodes in the tuple are correct (referred to as the *high-order metric*). For the second metric, for a given one-to-one alignment tuple of nodes, we consider it successful iff *any* pair of nodes (e.g. u_1 and u_2) in the tuple are aligned correctly (referred to as *pair-wise metric*). For both metrics, the alignment accuracy is calculated as $\frac{\#\text{ of correctly aligned node tuples}}{\#\text{ of node tuples in test data}}$, where the test data does not contain any known multi-way anchor links. The results on *Arxiv* without attribute are shown in Fig. 3. We observe that by the high-order metric, both basic algorithm and SyTE-Fast-P algorithm outperform baselines by up to 16.2%. By the pair-wise metric, our proposed methods cannot outperform, but are comparable with baselines. This is consistent with the goal of the proposed SyTE methods which are designed to primarily capture multi-way (i.e., high-order) associations. On the other hand, the baseline methods, being pair-wise approaches, are better suited for pair-wise association inference.

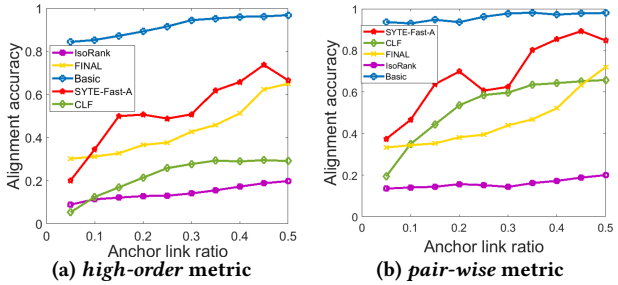


(a) *high-order* metric. **(b) *pair-wise* metric.**

Figure 3: Multi-network alignment results on *Arxiv* dataset (without node attributes). Best viewed in color.

The results of multi-network alignment of attributed networks on *DBLP* dataset are presented in Fig. 4. In most cases, both basic algorithm and SyTE-Fast-A outperform baseline methods. Although there are some performance loss of SyTE-Fast-A compared to the basic algorithm, the alignment accuracy of SyTE-Fast-A is still higher than baseline pair-wise network alignment methods.

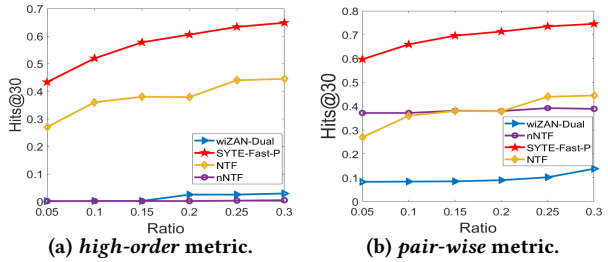
Second, we present the experimental results of the high-order recommendation task. We focus on one-class recommendation



(a) *high-order* metric **(b) *pair-wise* metric**

Figure 4: Multi-network alignment results on *DBLP* (with attributes). Best viewed in color.

in this task [29]. On one hand, for each user in the test data, if the returned top-k tuple list of (artist, tag) contains the ground-truth, we consider it as a successful hit. Similar to multi-network alignment, this is referred to as *high-order* metric. On the other hand, if either artist *or* tag is correctly recommended to a given user, we consider it as a successful recommendation. This is referred to as *pair-wise* metric. We use hits@30 for both metrics. The results are shown in Fig. 5. We can observe that the proposed algorithm SyTE-Fast-P outperforms baselines in terms of both *high-order* and *pair-wise* metrics.



(a) *high-order* metric. **(b) *pair-wise* metric.**

Figure 5: High-order recommendation results on *LastFm* dataset. Best viewed in color.

Next, we present the experimental results of the multi-network node retrieval task. For each query node of a given network, the task retrieves nodes from the other two networks for a top-k list. We study the hits@k vs. k for a fixed ratio of known anchor multi-way associations, and then fix $k = 30$ to study the hits@30 vs.

[6]Note that the curves of *REGAL* in both (c) and (d) are flat because *REGAL* is an unsupervised method.

the ratio of known multi-way anchor links. *DBLP* dataset is used for this task, and the results are shown in Fig. 2. In Fig. 2 (a) and (c), we can see that SyTE-Fast-P outperforms baselines by a large margin (e.g., by 50%+ when $k = 100$). In Fig. 2(b) and (d), the proposed SyTE-Fast-P algorithm does not outperform some pair-wise network alignment methods (e.g., *CrossMNA*, *FINAL*). This is also expected since the pair-wise metric is used for the retrieval task, whereas the proposed SyTE algorithms are primarily designed for the high-order metric (i.e., multi-way association).

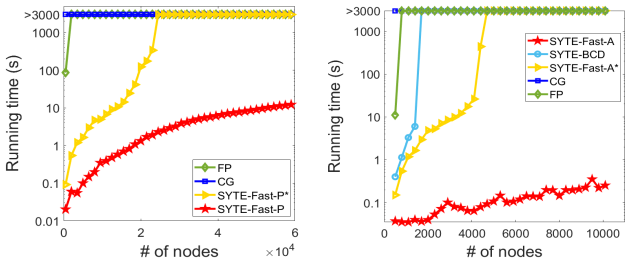
7.3 Scalability Results

In the heart of our proposed algorithm is a Sylvester equation solver. Here, so we compare the proposed methods with two classic Sylvester equation/linear system solvers, i.e., *Fixed Point method* (*FP*) and *Conjugate Gradient method* (*CG*)[7]. We extract subgraphs from the largest dataset *Aminer*, and the results are presented in Fig. 6. We terminate the program if it can not finish in 3,000 seconds. Note that the vertical axes are in *log* scale. From Fig. 6 (a), for plain networks, our proposed method SyTE-Fast-P exhibits a *linear* scalability w.r.t. the number of nodes of the input networks, whereas neither of the two baseline methods (*FP* and *CG*) can finish within 3,000 seconds with more than 1,200 nodes. SyTE-Fast-P* is a variant of SyTE-Fast-P, and it is detailed in the appendix. SyTE-Fast-A* is a variant of SyTE-Fast-A, which uses SyTE-Fast-P* in line 4 of Alg. 2. From Fig. 6 (b), the proposed SyTE-Fast-A scales linearly whereas all other methods scales super-linearly. Note that the blue curve (marked as SyTE-BCD) denotes a variant of SyTE-Fast-A by using the exact block coordinate descent method. As we can see, it can not scale up to large networks.

Additional scalability results w.r.t. the number of input networks are provided in the appendix.

7.4 Parameter Sensitivity

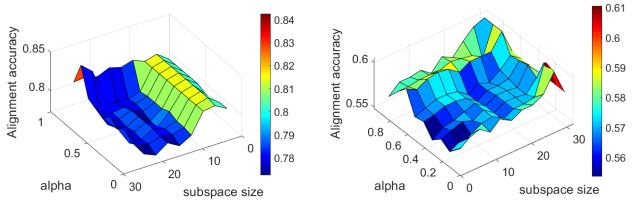
Here, we study the parameter sensitivity of the proposed algorithm SyTE-Fast-P. We use the multi-network alignment task to study the alignment accuracy w.r.t. two key parameters (i.e., α and Krylov subspace dimension l for SyTE-Fast-P). We use three subgraphs extracted from *Douban* dataset. The results are shown in Fig. 7. From Fig. 7, we can see that the performance of Algorithm 1 is stable in a relatively large range of parameter space.



(a) without node attributes. **(b) with node attributes.**

Figure 6: Scalability results and running time comparison on *Aminer* dataset. Notice the *log* scale in the vertical axis.

[7]Baseline methods *FINAL* and *IsoRank* also uses *FP* method. The supervised learning methods (e.g. *CrossMNA*) are difficult to be compared with our numerical method since they require off-line training.



(a) *pair-wise* metric. **(b) *high-order* metric.**

Figure 7: Parameter sensitivity analysis of the proposed Algorithm 1. Best viewed in color.

8 RELATED WORK

A - Multi-network Mining. Multi-network mining, especially those for addressing more than two input networks, has received increasing attention in recent years. For example, Liu et al. propose a multi-relation association learning method (*CGRL*) for joint inference over multiple networks which specifies the internal connections in each type of objects [17]. Chen et al. propose to solve the multi-label learning in multi-networks via a Sylvester equation [3]. Proposed by Zhang et al. *M-NASA* [31] is one of the earliest works on multi-network alignment. Chu et al. propose an embedding algorithm (*CrossMNA*) [6] which leverages inter- and intra-vectors for multi-network alignment. Other embedding based methods for *pair-wise* network alignment/link prediction include *IONE* [18] by Liu et al., *REGAL* by Heimann et al. [10], *CENALP* [9] by Du et al. etc. Multi-network embedding technique also draws significant attention recently. Representative works include [8, 20].

B - Sylvester (Tensor) Equation. Another line of recent work focuses on applying the Sylvester equation/tensor techniques for network mining, which has been shown its effectiveness in a variety of multi-network mining tasks. For example, Du et al. [7] develop a fast Sylvester equation solver for several pair-wise network mining tasks. Zhou et al. [35] propose an adversarial attack method on multi-network mining tasks which is also based on the Sylvester equation. Meanwhile, many algorithms have been developed in the scientific computing community for solving Sylvester equations [4, 14, 24] and Sylvester tensor equations [5]. Recent representative tensor-based approaches for (graph) data mining include [36, 37] for positive-unlabeled recommendation and multi-task crowdsourcing, and [34] for graph clustering.

C - Graph Matching. In the computer vision (CV) domain, graph matching, with the objective of matching anchor points between images, has been extensively studied. Recently there are works on multi-graph matching, such as [23, 26, 28]. Among them, *Tensor-MGM* by Shi et al. [23] proposes a tensor power iteration method for high-order optimization on multi-graph matching, which is remotely related to our work. Although [23] can also take multiple networks as inputs, there are two key differences. First, [23] relies on node/edge features to compute a cross-network node/edge similarity matrix. This is a reasonable design in the CV domain, since such features can be readily extracted from raw images. However, in our method, such a cross-network node similarity matrix/tensor is not mandatory. Second, the objective of methods in [23] focuses on the conformance of the matching result with the known cross-network similarity matrix, which is different from topological/attribute consistency principles in our objective. It is worth mentioning that the formulation of [23] degenerates to multi-dimensional assignment with only categorical node attributes.

9 CONCLUSION

In this paper, we formulate the multi-way association problem as a convex optimization problem, and show that it can be solved optimally by a Sylvester tensor equation. We propose two fast algorithms to solve this Sylvester tensor equation, with a *linear* complexity w.r.t. the size of input networks. On top of that, we provide theoretical analysis on the sensitivity of the Sylvester tensor equation solution. Extensive empirical evaluations demonstrate (1) the effectiveness on a variety of multi-network mining tasks (e.g., multi-network alignment, multi-network node retrieval and high-order recommendation), and (2) the *linear* scalability of the proposed methods.

10 ACKNOWLEDGEMENT

This work is supported by National Science Foundation under grant No. 1947135, and 2003924 by the NSF Program on Fairness in AI in collaboration with Amazon under award No. 1939725, by the United States Air Force and DARPA under contract number FA8750-17-C-0153 [8], and IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network. The content of the information in this document does not necessarily reflect the position or the policy of the Government or Amazon, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2011. 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011). In *Proceedings of the 5th ACM conference on Recommender systems (RecSys 2011)*. ACM, New York, NY, USA.

[2] Chen Chen, Hanghang Tong, Lei Xie, Lei Ying, and Qing He. 2016. FASCINATE: Fast Cross-Layer Dependency Inference on Multi-layered Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 765–774.

[3] Gang Chen, Yangqiu Song, Fei Wang, and Changshui Zhang. 2008. Semi-supervised multi-label learning by solving a sylvester equation. In *Proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM, 410–419.

[4] Minhong Chen and Daniel Kressner. 2019. Recursive blocked algorithms for linear systems with Kronecker product structure. *arXiv preprint arXiv:1905.09539* (2019).

[5] Zhen Chen and LinZhang Lu. 2012. A projection method and Kronecker product preconditioner for solving Sylvester tensor equations. *Science China Mathematics* 55, 6 (2012), 1281–1292.

[6] Xiaokai Chu, Xinxin Fan, Di Yao, Zhihua Zhu, Jianhui Huang, and Jingping Bi. 2019. Cross-Network Embedding for Multi-Network Alignment. In *The World Wide Web Conference*. ACM, 273–284.

[7] Boxin Du and Hanghang Tong. 2018. FASTEN: Fast Sylvester Equation Solver for Graph Mining. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1339–1347.

[8] Boxin Du and Hanghang Tong. 2019. MrMine: Multi-resolution Multi-network Embedding. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 479–488.

[9] Xingbo Du, Junchi Yan, Rui Zhang, and Hongyuan Zha. 2020. Cross-network skip-gram embedding for joint network alignment and link prediction. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[10] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. Regal: Representation learning-based graph alignment. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 117–126.

[11] Meng Jiang, Peng Cui, Fei Wang, Qiang Yang, Wenwu Zhu, and Shiqiang Yang. 2012. Social recommendation across multiple relational domains. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. 1422–1431.

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

[13] Danai Koutra, Hanghang Tong, and David Lubensky. 2013. Big-align: Fast bipartite graph alignment. In *2013 IEEE 13th International Conference on Data Mining*. IEEE, 389–398.

[14] Daniel Kressner and Christine Tobler. 2010. Krylov subspace methods for linear systems with tensor product structure. *SIAM journal on matrix analysis and applications* 31, 4 (2010), 1688–1714.

[15] Liangyue Li, Hanghang Tong, Nan Cao, Kate Ehrlich, Yu-Ru Lin, and Norbou Buchler. 2015. Replacing the irreplaceable: Fast algorithms for team member recommendation. In *Proceedings of the 24th International Conference on World Wide Web*. 636–646.

[16] Zhi Liang, Meng Xu, Maikun Teng, and Liwen Niu. 2006. NetAlign: a web-based tool for comparison of protein interaction networks. *Bioinformatics* 22, 17 (2006), 2175–2177.

[17] Hanxiao Liu and Yiming Yang. 2016. Cross-graph learning of multi-relational associations. *arXiv preprint arXiv:1605.01832* (2016).

[18] Li Liu, William K Cheung, Xin Li, and Lejian Liao. 2016. Aligning Users across Social Networks Using Network Embedding.. In *Ijcai*. 1774–1780.

[19] Federico Monti, Michael Bronstein, and Xavier Bresson. 2017. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*. 3697–3707.

[20] Jingchao Ni, Shiyu Chang, Xiao Liu, Wei Cheng, Haifeng Chen, Dongkuan Xu, and Xiang Zhang. 2018. Co-regularized deep multi-network embedding. In *Proceedings of the 2018 World Wide Web Conference*. 469–478.

[21] Adriana Prado, Marc Plantevit, Céline Robardet, and Jean-Francois Boulicaut. 2012. Mining graph topological patterns: Finding covariations among vertex descriptors. *IEEE Transactions on Knowledge and Data Engineering* 25, 9 (2012), 2090–2104.

[22] Yousef Saad. 2003. *Iterative methods for sparse linear systems*. Vol. 82. siam.

[23] Xinchu Shi, Haibin Ling, Weiming Hu, Junliang Xing, and Yanning Zhang. 2016. Tensor power iteration for multi-graph matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5062–5070.

[24] Valeria Simoncini and Vladimir Druskin. 2009. Convergence analysis of projection methods for the numerical solution of large Lyapunov equations. *SIAM J. Numer. Anal.* 47, 2 (2009), 828–843.

[25] 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.

[26] Runzhong Wang, Junchi Yan, and Xiaokang Yang. 2020. Graduated Assignment for Joint Multi-Graph Matching and Clustering with Application to Unsupervised Graph Matching Network Learning. *Advances in Neural Information Processing Systems* 33 (2020).

[27] Xian Wu, Baoxu Shi, Yuxiao Dong, Chao Huang, and Nitesh Chawla. 2018. Neural tensor factorization. *arXiv preprint arXiv:1802.04416* (2018).

[28] Junchi Yan, Minsu Cho, Hongyuan Zha, Xiaokang Yang, and Stephen M Chu. 2015. Multi-graph matching via affinity optimization with graduated consistency regularization. *IEEE transactions on pattern analysis and machine intelligence* 38, 6 (2015), 1228–1242.

[29] Yuan Yao, Hanghang Tong, Guo Yan, Feng Xu, Xiang Zhang, Boleslaw K. Szymanski, and Jian Lu. 2014. Dual-Regularized One-Class Collaborative Filtering. In *CIKM 2014, Shanghai, China, November 3-7, 2014*. 759–768.

[30] Jiawei Zhang and Philip S. Yu. 2015. Integrated anchor and social link predictions across social networks. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[31] Jiawei Zhang and Philip S. Yu. 2015. Multiple anonymized social networks alignment. In *2015 IEEE International Conference on Data Mining*. IEEE, 599–608.

[32] Si Zhang and Hanghang Tong. 2016. Final: Fast attributed network alignment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1345–1354.

[33] Yutao Zhang, Jie Tang, Zhilin Yang, Jian Pei, and Philip S Yu. 2015. Cosnet: Connecting heterogeneous social networks with local and global consistency. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1485–1494.

[34] Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. 2017. A local algorithm for structure-preserving graph cut. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 655–664.

[35] Qinghai Zhou, Liangyue Li, Nan Cao, Lei Ying, and Hanghang Tong. 2019. Admiring: Adversarial Multi-Network Mining. In *ICDM*.

[36] Yao Zhou, Jianpeng Xu, Jun Wu, Zeinab Taghavi, Evren Korpeoglu, Achan Kannan, and Jingrui He. 2021. PURE: Positive-Unlabeled Recommendation with Generative Adversarial Network. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM.

[37] Yao Zhou, Lei Ying, and Jingrui He. 2019. Multi-task Crowdsourcing via an Optimization Framework. *ACM Trans. Knowl. Discov. Data* 13, 3 (2019), 27:1–27:26. https://doi.org/10.1145/3310227

[38] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*. 912–919.

[8] Distribution Statement "A" (Approved for Public Release, Distribution Unlimited)

APPENDIX

We summarize the appendix as follows:

- **Additional Algorithms:** Basic Algorithm, and SyTE-Fast-P* for plain networks;
- **Proofs and Analysis:** Proofs for the theorems of optimality, complexity, and some mathematical derivations of the algorithms proposed in the paper;
- **Experimental Details:** The additional details of implementation, datasets, pre-processing, and baseline methods;
- **Experimental Results:** The additional parameter sensitivity and scalability results.

A – the Basic Algorithm

The Basic Algorithm (in Section 3) is summarized in Algorithm 3.

Algorithm 3 Basic Algorithm

Input: K Adjacency matrices of input networks $\mathbf{A}_1, ..., \mathbf{A}_K$, anchor link vector $\mathbf{b} = \text{vec}(\mathcal{B})$, node or edge attribute matrix $\mathbf{N}$, $\mathbf{E}$ if available, maximum iteration number t_{max};
Output: The solution vector $\mathbf{x}$ of Eq. (3).

1: Initialize $\mathbf{A}_f$, $\mathbf{A}_s$ by $k \in [1, K-1]$, $\mathbf{x}$, and $t = 1$;
2: Compute $\mathbf{E}_f^q = \otimes_{i=1}^{k} \mathbf{E}_i^q$, $\mathbf{E}_s^q = \otimes_{i=k+1}^{K} \mathbf{E}_i^q, \forall q \in [1, Q]$;
3: **while** $t < t_{max}$ **do**
4: Compute $\mathbf{V} = \text{reshape}(\mathbf{N}\mathbf{D}^{-1/2}\mathbf{x}, n^{K-k}, n^k)$;
5: $\mathbf{x} \leftarrow \alpha\mathbf{D}^{-1/2}\mathbf{N}\text{vec}(\sum_{q=1}^{Q}(\mathbf{E}_s^q \odot \mathbf{A}_s)\mathbf{V}(\mathbf{E}_f^q \odot \mathbf{A}_f)^\top) + (1-\alpha)\mathbf{b}$;
6: Set $t \leftarrow t + 1$;
7: **end while**
8: Return $\mathbf{x}$

Corollary 1. ***Convergence and Optimality of the Basic Algorithm.*** *The Algorithm 3 converges to the closed form solution* $(1-\alpha)(\mathbf{I} - \alpha\widetilde{\mathbf{W}})^{-1}\mathbf{b}$*, which is the global minimum of Eq.* (3).

Proof. See Theorem 1 in [32]. Omitted for brevity. □

B – SyTE-Fast-P* Algorithm

Here, we present another fast algorithm for solving the Sylvester tensor equation of plain network (Eq. (5)). First, since each $\tilde{\mathbf{A}}_i$ in Eq. (6) is diagonalizable (i.e., real symmetric matrix), we take the eigen-decomposition of each $\tilde{\mathbf{A}}_i = \mathbf{Q}_i\mathbf{\Lambda}_i\mathbf{Q}_i^\top$, in which $\mathbf{Q}_i$ is a matrix with each column as an eigenvector. In this case, it can be easily proved that the Eq. (6) can be written as:

$$(\mathbf{I} - \mathbf{\Lambda}_1 \otimes \cdots \otimes \mathbf{\Lambda}_K)\mathbf{y} = \mathbf{c} \tag{17}$$

where $\mathbf{c} = \mathbf{Q}^\top\mathbf{b}$ and $\mathbf{x} = \mathbf{Q}\mathbf{y}$, and $\mathbf{Q} = \mathbf{Q}_1 \otimes \cdots \otimes \mathbf{Q}_K$. Note that Eq. (17) is very easy to solve because the coefficient matrix is diagonal. If we use full eigen-decomposition for each $\tilde{\mathbf{A}}_i$, there would be no approximation error. However, the time complexity of calculating $\mathbf{x}$ from intermediate variable $\mathbf{y}$ would be $O(N^2)$. Since the adjacency matrices are usually sparse and low-rank, we could use rank-r ($r << n$) approximation on the eigen-decomposition of each $\tilde{\mathbf{A}}_i$. Then the linear system in Eq. (17) becomes much smaller ($r^K \times r^K$ instead of $n^K \times n^K$). For notation simplicity, we use the same r for each $\tilde{\mathbf{A}}_i$. The time complexity of calculating $\mathbf{y}$ is $O(r^K)$. Adding the time complexity of eigen-decomposition and calculating $\mathbf{c}$, the overall time complexity is reduced to $O(Krn^2 + r^K + r^K n^K)$. The proposed SyTE-Fast-P* algorithm is summarized in Algorithm 4.

For simplicity, we assume the ranks of eigen-decomposition are the same for each $\tilde{\mathbf{A}}_i$ in line 2. The intermediate solution $\mathbf{y}$ is solved in line 4, and the implicit representation of solution is returned and stored in line 5, which will significantly reduce the space complexity. $\mathbf{x}$ is calculated as $\mathbf{x} = \otimes_{j=1}^{K} \mathbf{Q}_j\mathbf{y}$.

With the proposal of Alg. 4 on plain networks, the SyTE-Fast-A, which uses SyTE-Fast-P in line 4 of Alg. 2, has another variant that instead uses SyTE-Fast-P*. We name it SyTE-Fast-A*, and its scalability is shown in Section 7.

Algorithm 4 SyTE-Fast-P* Algorithm

Input: K Normalized adjacency matrices of input networks $\tilde{\mathbf{A}}_1, ..., \tilde{\mathbf{A}}_K$, multi-way anchor link tensor $\mathcal{B}$, approximation rank r;
Output: The solution tensor $\mathcal{X}$ of Eq. (6).

1: **for** $i = 1, ..., K$ **do**;
2: Conduct top-r eigen-decomposition on $\tilde{\mathbf{A}}_i$ for $\mathbf{Q}_i$, $\mathbf{\Lambda}_i$;
3: **end for**
4: Calculate $\mathbf{c} = \otimes_{j=1}^{K} \mathbf{Q}_j^\top\mathbf{b}$, and solve Eq. (17) to obtain $\mathbf{y}$;
5: Return implicit solution $\{\mathbf{y}, \mathbf{Q}_j\}_{j=1}^{K}$.

C – Proofs and Analysis

We prove that the subsystems decomposed in Eq. (7) can be solved by a much smaller linear system (Eq. (9)) as follows.

Theorem 4. *Subsystem* $(\mathbf{I} - \tilde{\mathbf{A}_1} \otimes \cdots \otimes \tilde{\mathbf{A}_K})\mathbf{x}_i = \otimes_{j=1}^{K} \mathbf{b}_j^{(i)}$ *can be solved by first solving a small-scaled linear system* $(\otimes_{i=1}^{K} \mathbf{I}_{l_i+1,l_i} - \otimes_{i=1}^{K} \tilde{\mathbf{H}}_i)\mathbf{y} = \otimes_{i=1}^{K} \mathbf{U}_{l_i+1}^\top\mathbf{r}_0$.

Proof. For subsystem $(\mathbf{I} - \tilde{\mathbf{A}_1} \otimes \cdots \otimes \tilde{\mathbf{A}_K})\mathbf{x}_i = \otimes_{j=1}^{K} \mathbf{b}_j^{(i)}$, the initial residual vector $\mathbf{r}_0 = \mathbf{b} - (\mathbf{I} - \tilde{\mathbf{A}_1} \otimes \cdots \otimes \tilde{\mathbf{A}_K})\mathbf{x}_0$, given an initial solution vector $\mathbf{x}_0$ (which is typically initialized as zero vector). Let the updated solution vector be $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{z}_0$, in which we let $\mathbf{z}_0 \in \mathcal{K}_{\mathcal{Q}}^{\otimes}$. So $\mathbf{z}_0 = \otimes_{i=1}^{K} \mathbf{U}_{l_i}\mathbf{y}$. The updated residual to be minimized is as follows.

$$\mathbf{r_1} = \mathbf{r_0} - (\mathbf{I} - \bigotimes_{i=1}^{K} \mathbf{A}_i)(\bigotimes_{i=1}^{K} \mathbf{U}_{l_i})\mathbf{y} = \mathbf{r_0} - (\bigotimes_{i=1}^{K} \mathbf{U}_{l_i})\mathbf{y} + (\bigotimes_{i=1}^{K} \mathbf{U}_{l_i+1})(\bigotimes_{i=1}^{K} \tilde{\mathbf{H}}_i)\mathbf{y}$$

Recall that the second equation above is due to the Arnoldi process, which gives $\tilde{\mathbf{H}}_i = \mathbf{U}_{l_i+1}^\top\mathbf{A}_i\mathbf{U}_{l_i}$. Minimizing the norm of the updated residual gives us:

$$\begin{aligned}\min_{\mathbf{y}} ||\mathbf{r_1}||_2^2 &= \min_{\mathbf{y}} ||\bigotimes_{i=1}^{K} \mathbf{U}_{l_i+1}(\bigotimes_{i=1}^{K} \mathbf{U}_{l_i+1}^\top\mathbf{r}_0 - \bigotimes_{i=1}^{K} \mathbf{I}_{l_i+1,l_i}\mathbf{y} + \bigotimes_{i=1}^{K} \tilde{\mathbf{H}}_i\mathbf{y})||_2^2 \\ &= \min_{\mathbf{y}} ||\bigotimes_{i=1}^{K} \mathbf{U}_{l_i+1}^\top\mathbf{r}_0 - \bigotimes_{i=1}^{K} \mathbf{I}_{l_i+1,l_i}\mathbf{y} + \bigotimes_{i=1}^{K} \tilde{\mathbf{H}}_i\mathbf{y}||_2^2 \end{aligned} \tag{18}$$

where the second step is because $\otimes_{i=1}^{K} \mathbf{U}_{l_i+1}$ is a matrix with all columns being orthogonal with each other. In the above equation $\mathbf{I}_{l_i+1,l_i} = [\delta_{i,j}]_{1 \le i \le l_i+1, 1 \le j \le l_i}$, in which $\delta_{i,j}$ is the Kronecker δ-function, which is an identity like matrix with 1 in "diagonal" entries. Solving the minimization problem in Eq. (18) is actually equal to solving a smaller scaled linear system, compared to the original large linear system:

$$(\bigotimes_{i=1}^{K} \mathbf{I}_{l_i+1,l_i} - \bigotimes_{i=1}^{K} \tilde{\mathbf{H}}_i)\mathbf{y} = \bigotimes_{i=1}^{K} \mathbf{U}_{l_i+1}^\top\mathbf{r}_0 \tag{19}$$

Note that $\mathbf{x} = \mathbf{x}_1 + \cdots + \mathbf{x}_K$, where each $\mathbf{x}_i = \otimes_{j=1}^{K} \mathbf{U}_{l_j}^{(i)} \mathbf{y}_i$, and the solution tensor $\mathcal{X} = \text{reshape}(\mathbf{x}, n_K, ..., n_1)$, which does not need to be explicitly calculated in our algorithm. □

We then give the following theorem for the complexity of the proposed SyTE-Fast-P* algorithm. In the analysis of complexity, for notation simplicity, we assume all input networks share the same number of nodes n and number of edges m.

Theorem 5. ***Complexity of SyTE-Fast-P****. The time complexity of Algorithm 4 is $O(Krn^2 + r^K + r^K n^K)$. The space complexity of Algorithm 4 is $O(Km + Knr)$.*

Proof. **SyTE-Fast-P*:** K top-r eigen-decomposition takes $O(Krn^2)$. Solving Eq. (17) takes linear time complexity w.r.t. the linear system size, $O(r^K)$. Calculating $\mathbf{c}$ takes $O(r^K n^K)$. Overall, the time complexity for SyTE-Fast-P* is $O(Krn^2 + r^K + r^k n^K)$.

For space complexity, storing K adjacency matrices takes $O(Km)$, and storing K $\mathbf{Q}$ matrices of eigenvectors takes $O(Knr)$. Overall, the space complexity for SyTE-Fast-P* is $O(Km + Knr)$. □

Since K, r are usually much smaller than m, n (K, r are treated as *constants* in the big-O notation), both Alg. 1 and Alg. 4 have a much smaller time complexity than the Basic Algorithm (Algorithm 3) whose time complexity is $O(Qm^{\lfloor K/2 \rfloor} n^{\lfloor K/2 \rfloor} \cdot t_{max} + n^K)$.

Convexity of objective function. We prove the convexity of the objective function.

Lemma 1. *The objective function in Eq. (2) is convex.*

Proof. Since the gradient of Eq. (2) is $\nabla J(\mathbf{x}) = 2\alpha(\mathbf{I} - \tilde{\mathbf{W}})\mathbf{x} + 2(1-\alpha)(\mathbf{x} - \mathbf{b}) = 2(\mathbf{I} - \alpha\tilde{\mathbf{W}})\mathbf{x} + 2(1-\alpha)\mathbf{b}$. The Hessian matrix of objective function in Eq. (2) is $\mathbf{H}(J(\mathbf{x})) = 2(\mathbf{I} - \alpha\tilde{\mathbf{W}})$. As we have discussed in section 4, this matrix is strictly diagonal dominant when $\alpha \in (0, 1)$ and also positive definite. Eq. (2) is hence convex. □

D – Experimental Details

Hardware and software. All of the datasets are public. All experiments are performed on a machine with Intel(R) Core(TM) i7-9800X CPU with 3.80 GHz and 64.0 GB RAM. The algorithms are programmed with MATLAB R2019a with parallel computing.

Baseline methods. For pairwise network alignment methods in multi-network alignment task (*FINAL, IsoRank, CLF*), the alignment is first conducted on each pair of networks independently, and then the node alignments of each pair of networks are merged together to compare the multi-network alignment performance. The same strategy is used for pairwise methods in multi-network node retrieval task. For high-order recommendation task, *wiZAN-Dual* is conducted on user-user relation network with user-artist network, and user-user network with user-tag network, respectively. The recommendation result for each user is then merged together for high-order recommendation.

Other implementation details. For multi-network alignment, we implement a high-order greedy match algorithm. The multi-way association tensor can be transform to a high-order 0-1 tensor, in which each fiber contains at most one non-zero entry. For multi-network node retrieval task, given one node i_K from network G_K, the ranking list of the nodes from the rest of networks is calculated by sorting the slice of $\mathcal{X}(i_K, :, ... :)$.

Datasets. We use five datasets for evaluations as follows:

DBLP is a co-authorship network. Nodes represents authors while links represents co-authorship relation. The original dataset contains 42,252 nodes and 210,320 edges [21].

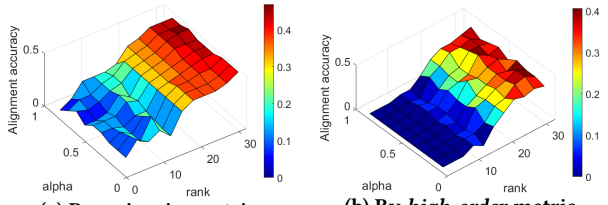
LastFm is a dataset for recommendation. It contains user-user friendship relation, user-artist listening relation, artist-tag categorization relation and artist profile. The original dataset contains 1,982 users, 17,632 artists and 11,946 tags [1].

Douban includes the users' friend relation in the online social network and offline activities, which share overlapping users. It contains 50k users and 5M edges in the original data. [33].

Arxiv[9] is a co-authorship network from two physical and one mathematical domains. Nodes represents authors and links represents co-authorship relation.

Aminer is an academic social network. Undirected edges represent co-authorship relationship. The whole dataset contains 1,274,360 nodes and 4,756,194 edges [33].

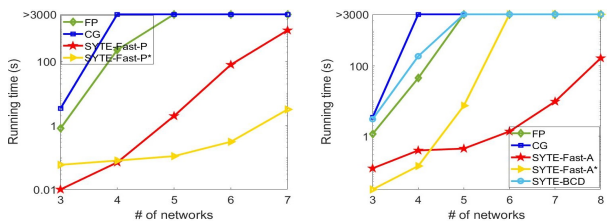
E – Additional Experimental Results



(a) By *pair-wise* metric. **(b) By *high-order* metric.**

Figure 8: Parameter sensitivity of SyTE-Fast-P* on *Arxiv*.

The parameter sensitivity study of SyTE-Fast-P* is presented in Fig. 8. The performance of SyTE-Fast-P* increases with the rank of eigen-decomposition, and α has small impact on the performance since we use uniform multi-way anchor link tensor here. As we can see, compared with Fig. 7, SyTE-Fast-P is relatively more stable w.r.t. the parameters α and subspace size, while the rank of eigen-decomposition has higher impact on the performance of SyTE-Fast-P*. The scalability results on the number of networks are shown in Fig. 9. The number of nodes used for each network is 100, and when the running time is larger than 3,000s, the program is terminated. The vertical axis is in *log* scale. As we can see, the proposed methods show exponential scalability w.r.t. the number of networks (relatively much smaller than the number of nodes) as we analyze in Section 4 and 5.



(a) On plain networks. **(b) On attributed networks.**

Figure 9: Scalability results of running time vs. the number of networks on *DBLP* dataset.

[9] https://comunelab.fbk.eu/data.php