SIAM J. MATRIX ANAL. APPL.
Vol. 44, No. 3, pp. 1006–1031

DOMINANT Z-EIGENPAIRS OF TENSOR KRONECKER PRODUCTS DECOUPLE*

CHARLES COLLEY†, HUDA NASSAR‡, AND DAVID F. GLEICH†

Abstract. Tensor Kronecker products, the natural generalization of the matrix Kronecker product, are independently emerging in multiple research communities. Like their matrix counterpart, the tensor generalization gives structure for implicit multiplication and factorization theorems. We present a theorem that decouples the dominant eigenvectors of tensor Kronecker products, which is a rare generalization from matrix theory to tensor eigenvectors. This theorem implies low-rank structure ought to be present in the iterates of tensor power methods on Kronecker products. We investigate low-rank structure in the network alignment algorithm TAME, a power method heuristic. Using the low-rank structure directly or via a new heuristic embedding approach, we produce new algorithms which are faster while improving or maintaining accuracy, and which scale to problems that cannot be realistically handled with existing techniques.

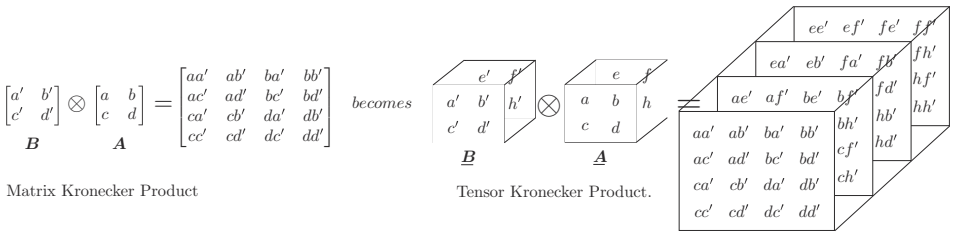
Key words. tensor Kronecker product, tensor eigenvectors, graph matching, network alignments

MSC codes. 68Q25, 15A42, 68R10

DOI. 10.1137/22M1502008

1. Introduction.

Given the ubiquity of the matrix Kronecker product, it is no surprise tensor analogues have been considered where



Existing research around this straightforward generalization of Kronecker products includes random graph models (Akoglu, McGlohon, and Faloutsos, 2008; Eikmeier, Ramani, and Gleich, 2018), image and tensor completion (Phan et al., 2012; Sun, Chen, and So, 2018), generalized CP decompositions (Batselier and Wong, 2017; Phan et al., 2013), and graph alignment (Park, Park, and Hebert, 2013; Mohammadi et al., 2017; Shen et al., 2018). Generalizations of Kronecker product multiplication (Sun et al., 2016; Shao, 2013), folding (Ragnarsson-Torbergsen, 2012, section 4.3.6), and structure inheritance properties (Batselier and Wong, 2017) have been discussed as well. We introduce a new theorem (section 3.2) which shows that the dominant

*Received by the editors June 10, 2022; accepted for publication (in revised form) by J. Nie December 22, 2022; published electronically July 14, 2023.
https://doi.org/10.1137/22M1502008
Funding: The work of the authors was partially supported by National Science Foundation grants IIS-1546488 and CCF-1909528, the NSF Center for Science of Information STC, CCF-0939370, DOE award DE-SC0014543, and the Sloan Foundation. The second author's research was completed at Purdue and Stanford.

†Computer Science, Purdue University, West Lafayette, IN 47907 USA (ccolley@purdue.edu, dgleich@purdue.edu).
‡RelationalAI, Boston, MA 02116 USA (huda.nassar@relational.ai).

z-eigenvector of the tensor $\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}}$ decouples into the dominant eigenvectors of $\underline{\boldsymbol{B}}$ and $\underline{\boldsymbol{A}}$. This result is a simple generalization of the matrix case, which is surprising given the many differences between tensor Z-eigenvectors and matrix eigenvectors.

Differences with the matrix case persist, though. For matrices, we have the stronger result that all eigenvectors of $\boldsymbol{B} \otimes \boldsymbol{A}$ (up to invariant subspaces) are Kronecker products of eigenvectors of $\boldsymbol{A}$ and $\boldsymbol{B}$. This is not true for even a diagonal tensor. Consider the example of the eigenvectors of a $4 \times 4 \times 4$ diagonal tensor with ones on the diagonal $\underline{\boldsymbol{D}}_4$. This tensor can be decomposed into two diagonal tensors, $\underline{\boldsymbol{D}}_4 = \underline{\boldsymbol{D}}_2 \otimes \underline{\boldsymbol{D}}_2$. Using the software associated with Cui, Dai, and Nie (2014) (or see Cartwright and Sturmfels (2013, Example 2.2)), the eigenvalues of $\underline{\boldsymbol{D}}_2$ are ± 1 and $\pm 1/\sqrt{2}$, where the eigenvectors corresponding to 1 are columns of the 2×2 identity matrix $\boldsymbol{I}$, and the eigenvector corresponding to $1/\sqrt{2}$ is $1/\sqrt{2}\begin{bmatrix}1 & 1\end{bmatrix}$. However, the eigenvalues of $\underline{\boldsymbol{D}}_4$ are ± 1, $\pm 1/2$, $\pm 1/\sqrt{3}$, and $\pm 1/\sqrt{2}$. The eigenvector for $\pm 1/\sqrt{3}$ is any permutation of $(1/\sqrt{3})\begin{bmatrix}1 & 1 & 1 & 0\end{bmatrix}^T$. None of these can be decomposed into the Kronecker product of any eigenvectors of $\underline{\boldsymbol{D}}_2$. Our theorem simply says the dominant eigenvector has such a decomposition. The existence of these other eigenvectors is expected as the set of projectively equivalent eigenvectors of symmetric tensors is exponential in the dimension of the tensor (Cartwright and Sturmfels, 2013, Theorem 1.2).

As an application of our new theorem, we discuss how we can apply both existing theory on mixed-products and our new theorem to facilitate the use of large motifs—small repeating subgraphs—in the network alignment algorithm TAME (Mohammadi et al., 2017). Network alignment problems date to the 1950s due to the relationship with the quadratic assignment problem and facility location (Bazaraa and Kirca, 1983; Koopmans and Beckmann, 1957; Lawler, 1963). Methods to address the problem range from relax and round on integer problems (Burkard, Dell'Amico, and Martello, 2012; Lawler, 1963), to Lagrangian relaxation techniques (Klau, 2009), to well-motivated heuristics (Singh, Xu, and Berger, 2008), and many other types of techniques (Patro and Kingsford, 2012; Malod-Dognin and Pržulj, 2015). Recent eigenvector-inspired spectral approaches (Singh, Xu, and Berger, 2008; Kollias, Mohammadi, and Grama, 2011; Feizi et al., 2019; Nassar et al., 2018) have been among the most scalable and versatile. TAME is a graph matching algorithm that uses a tensor Kronecker product and builds upon frameworks which align edges (Blondel et al., 2004; Zass and Shashua, 2008) to align motifs such as triangles. In TAME, the key computation is a tensor power method on $\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}}$, which is only ever manipulated implicitly due to its size. Implicit manipulation addresses the memory complexity but means the computational complexity is quadratic in the number of motifs of the network. Here, our theorem suggests that iterates of the power method should have low rank. We find this to be the case and design the LowRankTAME (see section 5.2) to run TAME using low-rank components. We discuss the limitations of LowRankTAME and how to overcome them with a new algorithm Λ-TAME (see section 4.3) which enforces rank 1 structure for additional scalability.

The new algorithm, LowRankTAME, gives around a 10-fold runtime improvement while producing the same iterates as TAME. The new algorithm Λ-TAME uses the decoupling in Theorem 3.3 to independently processes graphs $\boldsymbol{A}$ and $\boldsymbol{B}$ akin to the NSD algorithm (Kollias, Mohammadi, and Grama, 2011). When Λ-TAME is combined with careful refinement involving nearest neighbor queries, local search, or Klau's algorithm (Klau, 2009), it has faster end-to-end runtimes and better performance (the number of edges and motifs matched increases). Moreover, it scales up to aligning 9-cliques, which is well beyond the capabilities of existing algorithms. In

fact, the discrete operations involving matching and refinement now dominate runtime compared with the linear algebra.

The remainder of our paper formally establishes these results. It begins with a brief overview of our preliminaries for tensors (section 2). From there we move on to our three primary contributions:

(1) a novel extremal Z-eigenbound for tensor Kronecker products (section 3.2),
(2) an exploration of how newly found low-rank structure can accelerate the graph matching algorithm TAME (section 4.2),
(3) a new algorithm Λ-TAME which outperforms TAME in speed and accuracy and allows us to align larger motifs efficiently (section 4.3).

We evaluate our work by exploring larger motifs than previously considered feasible in section 5. We align protein-protein interaction (PPI) networks from the Biogrid repository (Stark et al., 2006) collected in the Local vs. Global Network Alignment collection (LVGNA) (Meng, Striegel, and Milenković, 2016) along with random geometric networks perturbed with partial duplication (Bhan, Galas, and Dewey, 2002; Chung et al., 2003; Hermann and Pfaffelhuber, 2014) and Erdős–Rényi noise models (Feizi et al., 2019, section 3.4). Our code is available (see section 5) and we have attempted to make our results as reproducible as possible by including the experiment driver codes as well.

2. Definitions and preliminaries.

2.1. General matrix and graph notation. We use uppercase bold letters for matrices, $\boldsymbol{A}$, $\boldsymbol{X}$, and lowercase bold letters for vectors, $\mathbf{x}, \mathbf{y}$. We use colons over an index to denote a row or column of a matrix, akin to MATLAB. The vector of all ones of length n is $\mathbb{1}_n$. A graph consists of a vertex set V an edge set E. It can be weighted with a positive edge weight for each edge and an implicit zero weight for each nonedge, or it can be unweighted, in which case edges have an implicit weight of 1 and nonedges have weight 0. All of the graphs we consider are undirected. The adjacency matrix then corresponds with a *symmetric* matrix of edge weights for a fixed order of the vertices.

2.2. Tensor notation and tensor eigenvectors. Tensors are denoted by bold underlined uppercase letters, $\underline{\boldsymbol{A}}$, $\underline{\boldsymbol{T}}$, $\underline{\boldsymbol{S}}$. We use a bold tuple of indices $\mathbf{i} = (i_1, \ldots, i_k)$ to denote each element of the tensor $\underline{A}(\mathbf{i}) = \underline{A}(i_1, \ldots, i_k)$ (Golub and van Loan, 2013, section 12.4.2). The axes of a tensor are called modes. A matrix is a 2-mode tensor, and we consider k-modes in general. We call a tensor symmetric if entries are the same in all permutations of the tensor modes. When the dimension for each mode of a tensor is the same, we call this tensor cubical. We use the shorthand $[n]^k$ to enumerate multi-indices $\mathbf{i}$ across all choices of $1 \ldots n$ in each of the k modes.

Let $\underline{\boldsymbol{A}}$ be a k-mode, cubical tensor of dimension n. We make frequent use of the polynomial

$$\sum_{\mathbf{i} \in [n]^k} \underline{A}(\mathbf{i}) x(i_1) x(i_2) \cdots x(i_k), \text{ written as } \underline{\boldsymbol{A}}\mathbf{x}^k,$$

which generalizes the quadratic $\mathbf{x}^T \boldsymbol{A}\mathbf{x} = \sum_{ij} A_{ij} x_i x_j$ (and we could write it as $\boldsymbol{A}\mathbf{x}^2$). We adopt a functional notation in general. When contracting $p \leq k$ modes (with potentially p different vectors), we express

$$\sum_{\boldsymbol{\ell} \in [n]^p} \underline{A}(\boldsymbol{\ell}, \mathbf{j}) x_1(\ell_1) \cdots x_p(\ell_p) \text{ as } \underline{\boldsymbol{A}}(\mathbf{x}_1, \ldots, \mathbf{x}_p),$$

where $\mathbf{j} \in [n]^{k-p}$ indexes the trailing uncontracted modes. When contracting the same vector in each mode, we can simply write $\underline{\boldsymbol{A}}\mathbf{x}^p$.

We call multiplying by a matrix $\boldsymbol{X} \in \mathbb{R}^{n\times r}$ in a given mode a modal product. The modal product of the first mode produces a nonsymmetric tensor, traditionally written as $\underline{\boldsymbol{A}} \times_1 \boldsymbol{X}$, whose first mode becomes dimension r and other modes remain dimension n. Modal products may be extended to $\underline{\boldsymbol{A}} \times_1 \boldsymbol{X} \times_2 \cdots \times_p \boldsymbol{X}$ for $p \le k$ products. For p matrices of dimension n by r matrices and the indices $\mathbf{i} \in [r]^p$ and $\mathbf{j} \in [n]^p$, we write a general modal product as

$$[\underline{\boldsymbol{A}} \times_1 \boldsymbol{X}_1 \times_2 \cdots \times_p \boldsymbol{X}_p](\mathbf{i},\mathbf{j}) = [\underline{\boldsymbol{A}}\big(\boldsymbol{X}_1(:,i_1), \boldsymbol{X}_2(:,i_2), \ldots, \boldsymbol{X}_p(:,i_p)\big)](\mathbf{j}). \tag{2.1}$$

There are a variety of notions of tensor eigenvectors (Qi and Luo, 2017). We use the Z-eigenvector of Qi (2005) or the ℓ_2 eigenvectors of Lim (2005). A Z-eigenpair of a tensor $\underline{\boldsymbol{A}}$ is a pair $(\lambda, \mathbf{x})$ with λ scalar and $\mathbf{x}$ an n-vector, where

$$\underline{\boldsymbol{A}}\mathbf{x}^{k-1} = \lambda\mathbf{x} \quad \|\mathbf{x}\|_2 = 1. \tag{2.2}$$

Equivalently, a tensor Z-eigenpair is a KKT point of the optimization problem

$$\text{maximize} \quad \underline{\boldsymbol{A}}\mathbf{x}^k \quad \text{subject to} \quad \|\mathbf{x}\|_2^k = 1. \tag{2.3}$$

There is an exponentially increasing number of tensor eigenvectors as the number of modes k grows (Cartwright and Sturmfels, 2013). For symmetric tensors, the eigenvectors can be computed via the Laserre hierarchy and convex programming (Cui, Dai, and Nie, 2014) or Newton methods (Jaffe, Weiss, and Nadler, 2018), although these techniques do not scale to large tensors. The higher-order power method (De Lathauwer et al., 1995), the symmetric shifted higher-order power method (SS-HOPM) (Kolda and Mayo, 2011), its generalizations (Kolda and Mayo, 2014), and dynamical systems (Benson and Gleich, 2019) are among the scalable ways to compute tensor eigenvectors. Although these scalable methods may not have the most satisfactory theoretical guarantees, they are practical and useful.

2.3. Kronecker products of tensors and vectorization. The Kronecker product $\otimes$ between matrices arises from treating the pair of matrices $\boldsymbol{B}$ and $\boldsymbol{A}$ in $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{B}^T$ as a linear operator from $\boldsymbol{X}$ to $\boldsymbol{Y}$. (The order arises from how the matrix $\boldsymbol{X}$ is linearized; see more below.) The way we present the definition of $\boldsymbol{B} \otimes \boldsymbol{A}$ for *matrices* involves a more complicated-seeming *interleaving* of indices from $\boldsymbol{A}$ and $\boldsymbol{B}$, but this will enable a seamless generalization to tensors. Let $i \underline{\perp} i'$ represent a linearization, or vectorization, of the pair i, i' to a single index. For instance, if both i, i' range from 1 to m, then $i \underline{\perp} i'$ represents the linearized index $i + m(i'-1)$ where we have vectorized by the first index. This joint index notation is exactly the vectorization

$$\operatorname{vec}(\boldsymbol{X})[i \underline{\perp} i'] = X(i, i'),$$

where we use the "matrix-to-vector" operator vec, which converts matrix-data into a vector by columns. We extend this definition to an interleaving of index pairs as in $(i,j) \underline{\perp} (i',j') \to (i \underline{\perp} i', j \underline{\perp} j')$. This gives us the matrix Kronecker product

$$(\boldsymbol{B} \otimes \boldsymbol{A})[i \underline{\perp} i', j \underline{\perp} j'] = A(i,j)B(i',j').$$

Using this notation we have for $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{B}^T$, let $\mathbf{y} = \operatorname{vec}(\boldsymbol{Y}), \mathbf{x} = \operatorname{vec}(\boldsymbol{X})$, and

$$y[i \underline{\perp} i'] = \sum_{j \underline{\perp} j'} (\boldsymbol{B} \otimes \boldsymbol{A})[i \underline{\perp} i', j \underline{\perp} j'] x[j \underline{\perp} j'] = \sum_{j \underline{\perp} j'} A(i,j)B(i',j')x[j \underline{\perp} j'].$$

The nice thing about this notation is it gives us a seamless way to generalize to tensors. Given two k-tuples of indices $\mathbf{i}$ and $\mathbf{i}'$ we have $\mathbf{i}_{\underline{\perp}}\mathbf{i}' = (i_1 \underline{\perp} i'_1, \ldots, i_k \underline{\perp} i'_k)$. Then if $\underline{\boldsymbol{A}}$ and $\underline{\boldsymbol{B}}$ are two k-mode cubical tensors, we have the elementwise definition

$$(\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}})[\mathbf{i}_{\underline{\perp}}\mathbf{i}'] = \underline{A}(\mathbf{i})\underline{B}(\mathbf{i}').$$

Equivalently, we can define this in terms of single element tensors. Let $\underline{\boldsymbol{E}}_{\mathbf{i}}$ be a tensor with a 1 in the $\mathbf{i}$ entry and zero elsewhere. Then $\underline{\boldsymbol{A}} = \sum_{\mathbf{i}} \underline{A}(\mathbf{i})\underline{\boldsymbol{E}}_{\mathbf{i}}$ and $\underline{\boldsymbol{B}} = \sum_{\mathbf{i}'} \underline{B}(\mathbf{i}')\underline{\boldsymbol{E}}_{\mathbf{i}'}$, and

$$\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}} = \left(\sum_{\mathbf{i}'} \underline{B}(\mathbf{i}')\underline{\boldsymbol{E}}_{\mathbf{i}'}\right) \otimes \left(\sum_{\mathbf{i}} \underline{A}(\mathbf{i})\underline{\boldsymbol{E}}_{\mathbf{i}}\right) = \sum_{\mathbf{i},\mathbf{i}'} \underline{A}(\mathbf{i})\underline{B}(\mathbf{i}')\underline{\boldsymbol{E}}_{\mathbf{i}'} \otimes \underline{\boldsymbol{E}}_{\mathbf{i}}.$$

Using the elementwise definition above $\underline{\boldsymbol{E}}_{\mathbf{i}'} \otimes \underline{\boldsymbol{E}}_{\mathbf{i}}$ only has a single nonzero in the $\mathbf{i}_{\underline{\perp}}\mathbf{i}'$ entry.

3. The dominant eigenvector of the multilinear Kronecker product. Our primary contribution within this section is the eigenvalue theorem (Theorem 3.3), which establishes a relationship between the dominant eigenpairs of the operands in $\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}}$ and the dominant eigenpair of that tensor. Both the proof of our main theorem and the faster graph matching computations use a number of results about computing the contraction

$$(\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}})\text{vec}(\boldsymbol{X})^{k-1} \text{ when } \boldsymbol{X} \text{ is low-rank.} \tag{3.1}$$

The contraction lemmas we use are known (Ragnarsson-Torbergsen, 2012; Shao, 2013; Sun et al., 2016; Batselier and Wong, 2017). We include our own proofs in the supplement SM1 for completeness and, if needed, to build intuition about our specific notation.

3.1. Existing contraction lemmas in our notation. We begin with a generalization of two core matrix Kronecker contraction theorems that we make use of in our proof and application: $(\boldsymbol{B} \otimes \boldsymbol{A})(\mathbf{y} \otimes \mathbf{x}) = (\boldsymbol{B}\mathbf{y}) \otimes (\boldsymbol{A}\mathbf{x})$ and $(\boldsymbol{B} \otimes \boldsymbol{A})\text{vec}(\boldsymbol{X}) = \text{vec}(\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}^T)$. Each lemma will allow us to compute the contractions in terms of the rank 1 components of $\boldsymbol{X}$. The lemmas are critical to power method algorithms because forming $\boldsymbol{B} \otimes \boldsymbol{A}$ is prohibitively expensive even in the matrix case. When the rank of $\boldsymbol{X}$ and the orders of the tensors are small this is a more effective strategy than implicit contraction with the dense form of $\boldsymbol{X}$.

LEMMA 3.1. *Given two k-mode, cubical tensors $\underline{\boldsymbol{A}}$ and $\underline{\boldsymbol{B}}$ of dimension m and n, respectively, and the $m \times n$ rank 1 matrix $\boldsymbol{X} = \mathbf{u}\mathbf{v}^T$, then for $1 \le p \le k$,*

$$(\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}})\text{vec}(\boldsymbol{X})^p = (\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}})(\mathbf{v} \otimes \mathbf{u})^p = \underline{\boldsymbol{B}}\mathbf{v}^p \otimes \underline{\boldsymbol{A}}\mathbf{u}^p. \tag{3.2}$$

The proof of the lemma can be found using unfolding theorems (Ragnarsson-Torbergsen, 2012, TKP Property 4), or a matrix specific version can be found in Batselier and Wong (2017, equation 3.3). Lemma 3.1 allows us to completely decouple the contractions between the two operands. The mixed product property actually generalizes in its entirety and can be found in Shao (2013, Theorem 3.1) and Sun et al. (2016, Proposition 2.3). The second lemma allows us to work with a general matrix $\boldsymbol{X} = \boldsymbol{Y}\boldsymbol{Z}^T$ where $\boldsymbol{Y}$ and $\boldsymbol{Z}$ have r columns. For the matrix case, we have $\text{vec}(\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}^T) = \text{vec}(\boldsymbol{A}\boldsymbol{Y}\boldsymbol{Z}^T\boldsymbol{B}^T) = (\boldsymbol{B}\boldsymbol{Z} \otimes \boldsymbol{A}\boldsymbol{Y})\text{vec}(\boldsymbol{I})$, where $\boldsymbol{I}$ is the $r \times r$ identity.

LEMMA 3.2. *Given two k-mode, cubical tensors $\underline{\boldsymbol{A}}$ and $\underline{\boldsymbol{B}}$ of dimension m and n, respectively, and the matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ of rank r with the r column decomposition $\boldsymbol{X} = \boldsymbol{Y}\boldsymbol{Z}^T$, then*

$$\begin{aligned}(\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}})vec(\boldsymbol{X})^p &= \sum_{\mathbf{i}=[r]^p} \underline{\boldsymbol{B}}(\boldsymbol{Z}(:,i_1),\ldots,\boldsymbol{Z}(:,i_p)) \otimes \underline{\boldsymbol{A}}(\boldsymbol{Y}(:,i_1),\ldots,\boldsymbol{Y}(:,i_p)) \\ &= ((\underline{\boldsymbol{B}} \times_1 \boldsymbol{Z} \times_2 \cdots \times_p \boldsymbol{Z}) \otimes (\underline{\boldsymbol{A}} \times_1 \boldsymbol{Y} \times_2 \cdots \times_p \boldsymbol{Y}))vec(\boldsymbol{I})^p,\end{aligned}$$

where $\boldsymbol{I}$ is the $r \times r$ identity matrix.

The proof can be found in Ragnarsson-Torbergsen (TKP Property 3). Lemma 3.2 is what we use to compute contractions when the rank of $\boldsymbol{X}$ or the order of the motifs is sufficiently small. We include self-contained proofs of each lemma using our notation in sections SM1.1 and SM1.2.

3.2. Dominant Z-eigenpairs. A useful property of Kronecker products is that the eigenvalues and eigenvectors of $\boldsymbol{B} \otimes \boldsymbol{A}$ decouple into Kronecker products of the eigenvectors of $\boldsymbol{A}$ and $\boldsymbol{B}$, individually. This makes spectral analysis of matrix Kronecker products efficient. We call an eigenpair *dominant* if it is the global maximum of $|\underline{\boldsymbol{A}}\mathbf{x}^k|$ where $\|\mathbf{x}\| = 1$. Here, we show that this decoupling property remains true for the dominant tensor eigenvector of a Kronecker product of tensors.

THEOREM 3.3. *Let $\underline{\boldsymbol{A}}$ be a symmetric, k-mode, m-dimensional tensor and $\underline{\boldsymbol{B}}$ be a symmetric, k-mode, n-dimensional tensor. Suppose that $(\lambda_A^*, \mathbf{u}^*)$ and $(\lambda_B^*, \mathbf{v}^*)$ are any dominant tensor Z-eigenvalues and vectors of $\underline{\boldsymbol{A}}$ and $\underline{\boldsymbol{B}}$, respectively. Then $(\lambda_A^*\lambda_B^*, \mathbf{v}^* \otimes \mathbf{u}^*)$ is a dominant eigenpair of $\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}}$. Moreover, any Kronecker product of Z-eigenvectors of $\underline{\boldsymbol{A}}$ and $\underline{\boldsymbol{B}}$ is a Z-eigenvector of $\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}}$.*

Proof. Let $\mathbf{x} = \text{vec}(\boldsymbol{X})$ be any vector with $\|\mathbf{x}\|_2 = \|\boldsymbol{X}\|_F = 1$ where $\boldsymbol{X}$ is an $m \times n$ matrix. Let $z(i) = \|\boldsymbol{X}(:,i)\|_2$. We have $\|\mathbf{z}\|_2 = 1$ as well. Then we create

$$\boldsymbol{Z} = \text{diag}(z(1),\ldots,z(n)) \text{ and } \boldsymbol{Y} \text{ so that } \boldsymbol{X} = \boldsymbol{Y}\boldsymbol{Z}\ .$$

The ith column of $\boldsymbol{Y}$ is either normalized or entirely 0 (if $z(i) = 0$). Recall that the dominant eigenpair maximizes $|(\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}})\mathbf{x}^k| = |(\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}})\text{vec}(\boldsymbol{X})^k|$. From Lemma 3.2 we have

$$\begin{aligned}|(\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}})\text{vec}(\boldsymbol{X})^k| &= \left|\sum_{\mathbf{i}} \underline{\boldsymbol{A}}(\boldsymbol{Y}(:,i_1),\ldots,\boldsymbol{Y}(:,i_k))\underline{\boldsymbol{B}}(\boldsymbol{Z}(:,i_1),\ldots,\boldsymbol{Z}(:,i_k))\right| \\ &= \left|\sum_{\mathbf{i}} \underline{\boldsymbol{A}}(\boldsymbol{Y}(:,i_1),\ldots,\boldsymbol{Y}(:,i_k)) \prod_{j=1}^{k} z(i_j)\underline{\boldsymbol{B}}(\boldsymbol{I}(:,i_1),\ldots,\boldsymbol{I}(:,i_k))\right|,\end{aligned}$$

where $\boldsymbol{I}(:,j)$ is the jth column of the identity matrix. Now, because $\underline{\boldsymbol{A}}$ is symmetric, we have that

$$|\lambda_A^*| = \text{maximize } |\underline{\boldsymbol{A}}(\mathbf{u}_1,\ldots,\mathbf{u}_k)| \text{ subject to } \|\mathbf{u}_i\| = \{0,1\}.$$

This follows from a result on the best rank 1 approximation of a symmetric tensor (Zhang, Ling, and Qi, 2012, Theorem 2.1), where the result is with $\|\mathbf{u}_i\| = 1$. We can handle cases where $\|\mathbf{u}_i\| = 0$ (which we could have for zero columns of $\boldsymbol{X}$) by simply noting that such an $\boldsymbol{X}$ would make $(\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}})\text{vec}(\boldsymbol{X})^k = 0$, so the maximum will never occur for those. Thus, this gives us an upper-bound on $|\underline{\boldsymbol{A}}(\boldsymbol{Y}(:,i_1),\ldots,\boldsymbol{Y}(:,i_k))|$

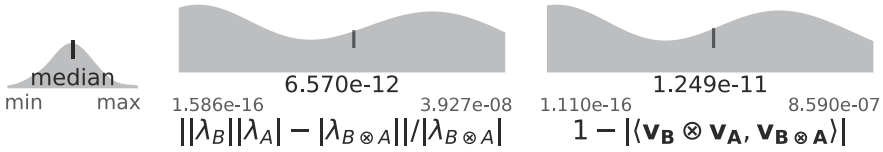


FIG. 3.1. *We computationally verify Theorem* 3.3 *in a handful of random problems. Our synthetic problems are of size* $m, n \in \{2, 3, 4\}$ *and* $k \in \{3, 4, 5\}$. *There are* 30 *trials overall and we show a density plot, median, min, and max over the results. We report the differences of the dominant z-eigenpairs of tensor Kronecker product and its operands. For each tensor we use the largest magnitude* λ *found from each of the methods (Cui, Dai, and Nie,* 2014*; Jaffe, Weiss, and Nadler,* 2018*) and its associated eigenvector. To measure eigenvector similarity, we use* $1-|\langle\cdot,\cdot\rangle|$, *where the absolute value addresses sign discrepancies. We initialize the NCM methods with* 5000 *uniformly drawn points from the unit sphere, and use the default parameters of the methods of Cui, Dai, and Nie. The NCM methods use a tolerance of* 10^{-10} *to measure differences in eigenvalues, whereas the methods of Cui, Dai, and Nie use* 10^{-4}. *We stored the tensors and results for future study by others in our codes.*

$$|(\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}})\mathrm{vec}(\boldsymbol{X})^k| \leq |\lambda_A^*| \cdot \left| \sum_{\mathbf{i}} \prod_{j=1}^{k} z(i_j) \underline{\boldsymbol{B}}(\boldsymbol{I}(:, i_1), \ldots, \boldsymbol{I}(:, i_k)) \right|.$$

Here, $\underline{\boldsymbol{B}}(\boldsymbol{I}(:, i_1), \ldots, \boldsymbol{I}(:, i_k))$ is just $\underline{B}(\mathbf{i})$ and $|\sum_{\mathbf{i}} \prod_{j=1}^{k} z(i_j)\underline{B}(\mathbf{i})| = |\underline{\boldsymbol{B}}\mathbf{z}^k| \leq |\lambda_B^*|$. Putting the pieces together, we have that the dominant Z-eigenvalue of $\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}} \leq |\lambda_A \lambda_B|$.

Now we show that Kronecker products of eigenvectors are also eigenvectors. Let $\mathbf{u}$ and $\mathbf{v}$ be any Z-eigenvectors of $\underline{\boldsymbol{A}}$ and $\underline{\boldsymbol{B}}$, with eigenvalues λ_A and λ_B, respectively; then

$$(\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}})(\mathbf{v} \otimes \mathbf{u})^{k-1} = (\underline{\boldsymbol{B}}\mathbf{u}^{k-1}) \otimes (\underline{\boldsymbol{A}}\mathbf{v}^{k-1}) = \lambda_B \mathbf{u} \otimes \lambda_A \mathbf{v} = (\lambda_A \lambda_B)(\mathbf{v} \otimes \mathbf{u}).$$

Using $\mathbf{u}^*$ and $\mathbf{v}^*$ gives us an eigenvector that achieves the upper-bound $|\lambda_A^* \lambda_B^*|$. ∎

Observations. For nonnegative tensors $\underline{\boldsymbol{A}}$ and $\underline{\boldsymbol{B}}$ in Theorem 3.3, λ_A, λ_B are nonnegative because the components of the best rank 1 approximation are nonnegative (Qi, 2018, Proposition 6). Consequently, the dominant eigenvalue and eigenvectors of the Kronecker product are nonnegative.

We provide additional MATLAB routines (and precomputed results) making use of Cui, Dai, and Nie (2014) and Jaffe, Weiss, and Nadler (2018) as computational verification for Theorem 3.3 with randomized symmetric tensors which are small enough to enumerate the entire spectrum of $\underline{\boldsymbol{A}}$ and $\underline{\boldsymbol{B}}$. As expected, we found no counterexamples with random dense symmetric tensors generated with the tensor toolbox (Bader and Kolda, 2008), where we sample the spectra with the constrained polynomial optimization of Cui, Dai, and Nie (2014) and the Newton correction method (and its orthogonal variant) of Jaffe, Weiss, and Nadler (2018). We report the relative difference between the largest magnitude z-eigenvalue found for $\underline{\boldsymbol{A}}$, $\underline{\boldsymbol{B}}$, and $\underline{\boldsymbol{B}} \otimes \underline{\boldsymbol{A}}$, and the inner product of the associated eigenvectors in Figure 3.1. This identified no exceptions to our theorem up to computational tolerances.

4. Faster higher-order graph alignment methods via Kronecker structure. We show how the tensor Kronecker product results from the previous section allow us to improve the higher-order network alignment algorithm TAME (Mohammadi et al., 2017) in two ways. First, Lemma 3.2 is the key to understanding how to use

the Kronecker structure to make the iteration $(\underline{\boldsymbol{T}}_B \otimes \underline{\boldsymbol{T}}_A)\mathbf{x}^{k-1}$ from TAME faster when $\mathbf{x} = \text{vec}(\boldsymbol{X})$ and $\boldsymbol{X}$ is low-rank and the number of modes—equivalent to the size of the motif—is not too large. Second, when expanding to larger motifs, such as 9-cliques, we will need to rely on our new simpler algorithm Λ-TAME (section 4.3), which is built upon our novel decoupling result, Theorem 3.3.

4.1. Background on higher-order graph alignment. Higher-order graph alignment considers motifs, or small subgraphs, beyond edges (Conte et al., 2004; Bayati et al., 2013; Singh, Xu, and Berger, 2008) that are matched between a pair of networks. A motif is simply a graph—usually small, like a triangle—and an instance of a motif in a graph is simply an instance of an isomorphic induced subgraph (Milo et al., 2002). From any graph we can induce a k-regular hypergraph H by identifying hyperedges with the presence of motifs with k vertices (Estrada and Rodríguez-Velázquez, 2006; Klymko, Gleich, and Kolda, 2014; Benson, Gleich, and Leskovec, 2015; Mohammadi et al., 2017). Then the full edge set of the hypergraph involves enumerating all the instances of the motif. This can be computationally demanding to enumerate complicated motifs but is fast for simple motifs like triangles and small cliques, and random sampling can make the process reasonable for larger cliques (Jain and Seshadhri, 2020). Analogously to the adjacency matrix, we use an adjacency tensor $\underline{\boldsymbol{A}}$ to denote the presence of these motifs or, equivalently, hyperedges. Formally,

$$\underline{A}(i_1, \ldots, i_k) = \begin{cases} 1 & \text{if nodes } i_1, \ldots, i_k \text{ form motif } M, \\ 0 & \text{else.} \end{cases}$$

Each permutation of the indices corresponds to a different orientation of the motif M. Consequently, the adjacency tensor of a hypergraph is a symmetric, cubical tensor for the motifs we consider.

The higher-order graph alignment problem we consider is defined in terms of a matching between the vertices of two graphs (Chertok and Keller, 2010; Park, Park, and Hebert, 2013; Mohammadi et al., 2017). For a pair of graphs A and B we characterize a matching between their vertex sets as a matrix.

DEFINITION 4.1 (matching matrix). *Let A and B be two graphs of size m and n, respectively; then we define the matching matrix $\boldsymbol{X} \in \{0,1\}^{m \times n}$ such that*

$$\boldsymbol{X}\mathbb{1}_n \leq \mathbb{1}_m, \quad \boldsymbol{X}^T\mathbb{1}_m \leq \mathbb{1}_n, \quad \textit{and } \boldsymbol{X}(i,i') = \begin{cases} 1 & \textit{if } i \in V_A \textit{ is matched to } i' \in V_B, \\ 0 & \textit{else.} \end{cases}$$

We suppose we are given a similarity tensor $\underline{\boldsymbol{S}}$ where entries can be indexed using a pair of tuples $\mathbf{i}$ to represent the vertices of a motif in graph A and $\mathbf{i}'$ to represent the vertices of motif from graph B. The value $\underline{S}(\mathbf{i}, \mathbf{i}')$ indicates the similarity of the motif at indices $\mathbf{i}$ in graph A to the motif at indices $\mathbf{i}'$ in graph B. A simple form of higher-order graph alignment problem is to optimize

$$\begin{aligned} &\text{maximize} && \textstyle\sum_{\mathbf{i}} \sum_{\mathbf{i}'} [\underline{S}(\mathbf{i}, \mathbf{i}') X(i_1, i_1') X(i_2, i_2') \cdots X(i_k, i_k')] \\ &\text{subject to} && \boldsymbol{X} \text{ is a matching.} \end{aligned}$$

The goal here is to find high-similarity entries $\underline{S}(\mathbf{i}, \mathbf{i}')$ where the vertices involved in the motifs are matched. This subsumes an edge based alignment framework (such as Feizi et al. (2019)) because $\mathbf{i}$ could have just been the pair (i, j).

We often find it convenient to write this objective as

$$\begin{aligned} &\text{maximize} && \underline{\boldsymbol{V}}\text{vec}(\boldsymbol{X})^k = \underline{\boldsymbol{V}}\mathbf{x}^k \\ &\text{subject to} && \boldsymbol{X} \text{ is a matching,} \end{aligned}$$

where we convert the similarity tensor $\underline{\boldsymbol{S}}$ into a tensor $\underline{\boldsymbol{V}}$ indexed with the same order with the vec operator. This tensor to "*operator* for vec($\boldsymbol{X}$)" transformation is something we repeatedly use and we write it as

(4.1)
$$\underline{\boldsymbol{S}} \underset{\text{vec}}{\Leftrightarrow} \underline{\boldsymbol{V}} \text{ means } S(i_1, \cdots, i_k, i'_1, \cdots, i'_k) = \underline{S}(\mathbf{i}, \mathbf{i}') = \underline{V}[\mathbf{i}\underline{\downarrow}\mathbf{i}'] = V[i_1\underline{\downarrow}i'_1, \cdots, i_k\underline{\downarrow}i'_k].$$

This vec-form makes the *eigenvector*-heuristic inspiration clear because eigenvectors optimize the generalized Rayleigh quotient $\underline{\boldsymbol{A}}\mathbf{x}^k$.

Choices of $\underline{\boldsymbol{S}}$ whose indices decompose into products of independent indices (i.e., (i,j,k) and (i',j',k')) give rise to tensors $\underline{\boldsymbol{V}}$ with Kronecker structure. This can arise via motifs as in TAME (Mohammadi et al., 2017) and weighted angle alignments (Park, Park, and Hebert, 2013) as well as alignments with nonmotifs (Feizi et al., 2019), whose edge based construction could generalize to hypergraphs.

In TAME (Mohammadi et al., 2017), we set $\underline{\boldsymbol{S}}$ to 1 if there is a triangle at both $\mathbf{i}$ in A and $\mathbf{i}'$ in B. If we denoted the triangles of A and B in the triangle adjacency tensors $\underline{\boldsymbol{T}}_A$ and $\underline{\boldsymbol{T}}_B$, respectively, TAME's similarity tensor $\underline{\boldsymbol{S}}$ would be $\underline{S}(i,j,k,i',j',k') = \underline{T}_A(i,j,k)\underline{T}_B(i',j',k')$. Though simple, this form is informative when given a matching $\boldsymbol{X}$, as

$$\sum_{i,j,k}\sum_{i',j',k'} \underbrace{\underline{T}_A(i,j,k)\underline{T}_B(i',j',k')}_{\underline{S}(\mathbf{i},\mathbf{i}')=\underline{S}(i,j,k,i',j',k')} X(i,i')X(j,j')X(k,k') = 6\begin{pmatrix}\text{the number of}\\ \text{triangles aligned}\\ \text{between A and B}\end{pmatrix}.$$

This also gives us a tensor $\underline{\boldsymbol{V}} = \underline{\boldsymbol{T}}_B \otimes \underline{\boldsymbol{T}}_A$. The computer vision algorithm HOFASM (Park, Park, and Hebert, 2013) approximates a similarity tensor $\underline{\boldsymbol{S}}$ between pairs of triplets of a image features with a tensor of the form $\underline{\boldsymbol{V}} = \sum_{r,s} \underline{\boldsymbol{B}}_{r,s} \otimes \underline{\boldsymbol{H}}_{r,s}$. For simplicity, we define the following objective function that will guide our subsequent research.

Definition 4.2 (global graph alignment). *Fix graphs A and B to have m and n vertices, respectively, an $m \times n$ prior weight matrix $\boldsymbol{W}$, and a motif M with k vertices. Let $\underline{\boldsymbol{S}}$ be a $2k$-mode similarity tensor where the $\underline{S}(\mathbf{i},\mathbf{i}')$ entry denotes the similarity between the motifs induced by the vertices $\mathbf{i}$ in graph A and $\mathbf{i}'$ in graph B. Then we wish to find a matching $\boldsymbol{X}$ between the vertices in A to the vertices in B which optimizes*

$$\begin{aligned} &\textit{maximize} && \textstyle\sum_{\mathbf{i}}\sum_{\mathbf{i}'}[\underline{S}(\mathbf{i},\mathbf{i}')X(i_1,i'_1)X(i_2,i'_2)\cdots X(i_k,i'_k)] + \sum_{i,i'} W(i,i')X(i,i') \\ &\textit{subject to} && \boldsymbol{X} \textit{ is a matching.} \end{aligned}$$

Equivalently, we let $\underline{\boldsymbol{V}}$ be the k-mode "vec-operator" form of $\underline{\boldsymbol{S}}$, i.e., $\underline{\boldsymbol{S}} \underset{vec}{\Leftrightarrow} \underline{\boldsymbol{V}}$ or $\underline{V}[i_1\underline{\downarrow}i'_1, \ldots, i_k\underline{\downarrow}i'_k] = S(\mathbf{i},\mathbf{i}')$. Then the problem is

$$\begin{aligned} &\textit{maximize} && \underline{\boldsymbol{V}}\textit{vec}(\boldsymbol{X})^k + \textit{trace}(\boldsymbol{W}^T\boldsymbol{X}) \\ &\textit{subject to} && \boldsymbol{X} \textit{ is a matching,} \end{aligned} \tag{4.2}$$

which makes the tensor-eigenvector inspiration clear (see section 2.2).

The tensor $\underline{\boldsymbol{S}}$ will change depending on what structure we will consider for the higher-order matching problem and we may adjust the weightings between the prior matrix and the affinity tensors (a similar edge based framework can be found in Berg, Berg, and Malik (2005)). Note that $\underline{\boldsymbol{S}}$ is not required to be symmetric in permutations of the first k entries, which are permutations of $\mathbf{i}$, but in the problems we consider in this paper, it will be, and likewise for permutations of the last k entries for $\mathbf{i}'$. Note that this means that $\underline{\boldsymbol{V}}$ is a symmetric tensor, although $\underline{\boldsymbol{S}}$ is not even cubical.

4.2. TAME and LowRankTAME. TAME is a spectral method that uses a tensor-eigenvector heuristic to guide an alignment. It arises from the network alignment literature in bioinformatics. The TAME method is a simple instance of the higher-order graph alignment framework (Definition 4.2) where, given two graphs A and B, we first enumerate triangles (or any motif of interest) in each, to build triangle adjacency tensors $\underline{\boldsymbol{T}}_A$ and $\underline{\boldsymbol{T}}_B$. Then we set $\underline{S}(\mathbf{i},\mathbf{i}') = \underline{T}_A(\mathbf{i})\underline{T}_B(\mathbf{i}')$. For this choice, we have

$$\underline{\boldsymbol{S}} \underset{\text{vec}}{\Leftrightarrow} \underline{\boldsymbol{V}} = \underline{\boldsymbol{T}}_B \otimes \underline{\boldsymbol{T}}_A. \tag{4.3}$$

This results in the following idealized optimization problem for TAME:

$$\begin{aligned} &\text{maximize} && (1-\alpha)\text{trace}(\boldsymbol{W}^T\boldsymbol{X}) + \tfrac{\alpha}{6}(\underline{\boldsymbol{T}}_B \otimes \underline{\boldsymbol{T}}_A)\text{vec}(\boldsymbol{X})^3 \\ &\text{subject to} && \boldsymbol{X} \text{ is a matching.} \end{aligned} \tag{4.4}$$

Here the value $\alpha/6$ arises because each triangle alignment gives six entries in $\underline{\boldsymbol{T}}_B \otimes \underline{\boldsymbol{T}}_A$ due to symmetry. The weight matrix $\boldsymbol{W}$ gives flexibility to bias the alignment toward certain nodes. When no prior matrix is available, we make use of a rank 1 matrix $\boldsymbol{W} = \frac{1}{mn}\mathbb{1}_m\mathbb{1}_n^T$, which gives a uniform bias everywhere.

The heuristic procedure used in TAME is to deploy the SS-HOPM algorithm (Kolda and Mayo, 2011) to seek a tensor eigenvector, or near tensor eigenvector, of $\boldsymbol{V} = \underline{\boldsymbol{T}}_B \otimes \underline{\boldsymbol{T}}_A$. We show the procedure in Algorithm 4.1. We present an affine-shift variant of the TAME method that includes the mixing parameter α to remix in the original iterate, whereas TAME (Mohammadi et al., 2017) fixed $\alpha = 1$. This choice sometimes helps boost performance a little bit.

Algorithm 4.1. TAME (Mohammadi et al., 2017) with affine shift.

Require: k-mode motif tensors $\underline{\boldsymbol{T}}_A, \underline{\boldsymbol{T}}_B$ for graphs A and B, mixing parameter α, shift β, tolerance ε, weights $\boldsymbol{W}$

Ensure: Alignment heuristic $\boldsymbol{X}$ and max-weight matching of $\boldsymbol{X}$

$\boldsymbol{X}_0 = \boldsymbol{W}/\|\boldsymbol{W}\|_F$ ▷ *Normalize first iterate*

for $\ell = 0, 1, \ldots$ until $|\lambda_{\ell+1} - \lambda_\ell| < \varepsilon$ **do**

  ▷*SS-HOPM iteration*

  $\boldsymbol{X}_{\ell+1} = \text{unvec}((\underline{\boldsymbol{T}}_B \otimes \underline{\boldsymbol{T}}_A)\text{vec}(\boldsymbol{X}_\ell)^{k-1})$ ▷*Implicitly*

  $\lambda_{\ell+1} = \text{trace}(\boldsymbol{X}_\ell^T \boldsymbol{X}_{\ell+1})$ ▷ *Estimate tensor-eval*

  $\boldsymbol{X}_{\ell+1} \leftarrow \alpha\boldsymbol{X}_{\ell+1} + \alpha\beta\boldsymbol{X}_\ell + (1-\alpha)\boldsymbol{X}_0$

  $\boldsymbol{X}_{\ell+1} \leftarrow \boldsymbol{X}_{\ell+1}/\|\boldsymbol{X}_{\ell+1}\|_F$

  Set $t_{\ell+1}$ to be the number of motifs aligned by a matching from $\boldsymbol{X}_{\ell+1}$

return $\boldsymbol{X}_\ell$ and the matching of $\boldsymbol{X}_\ell$ with the highest t_ℓ

Rounding with matching and scoring. At each iteration, we explicitly *round* the continuous valued $\boldsymbol{X}$ and compute a matching using a max-weight matching algorithm. Then the procedure returns the best iterate with the highest downstream objective (triangle alignment, mixture, or some other combination). Returning the full iterate information is helpful for further refinement of the solution using a local search strategy described in section 4.4. This max-weight matching step, which is executed at each iteration, becomes expensive after we optimize the linear algebra using the Kronecker theory.

Implicit multiplication. In TAME the authors make use of an implicit operation to compute the iterates of the tensor powers

$$\underline{\boldsymbol{T}}_B \otimes \underline{\boldsymbol{T}}_A \mathrm{vec}(\boldsymbol{X})^{k-1} \tag{4.5}$$

without forming $\underline{\boldsymbol{T}}_B \otimes \underline{\boldsymbol{T}}_A$. This computation still takes $O(\mathrm{nnz}(\underline{\boldsymbol{T}}_B)\mathrm{nnz}(\underline{\boldsymbol{T}}_A))$ work, where nnz is the number of nonzero entries in the sparse tensor. In the case of the uniform bias prior ($\boldsymbol{W} = \frac{1}{mn}\mathbb{1}_m\mathbb{1}_n^T$), the first iterate is rank 1, so we could apply Lemma 3.1 to decouple the operation. Because of the shift β, however, subsequent iterations will not remain rank 1, as the following observation clarifies.

Our observation. Suppose that $\boldsymbol{W}$ is rank 1 and we are dealing with a k-mode tensor. We can use Lemma 3.2 to bound the rank of successive iterates. Suppose that we have a rank r iterate $\boldsymbol{X}_\ell$. Then Lemma 3.2 shows that the *next iterate* $\boldsymbol{X}_{\ell+1}$ can grow to rank at most $r^{k-1} + \mathtt{rank}([\beta\boldsymbol{X}_\ell, (1-\alpha)\boldsymbol{W}]) \le r^{k-1} + r + 1$. This follows from the number of combinations of vectors in the lemma combined with the addition of the rank factors reintroduced by the shifts α and β. In the case of symmetric tensors, r^{k-1} can be reduced to $\binom{r+k-2}{k-1}$, but we use the loose upper-bound r^{k-1} for simplicity.

This analysis gives the worst-case scenario for the rank growth of the iterates. In practice, we find it extremely conservative. (See evidence in section 5.2.) This means that there is a useful low-rank strategy to employ with our theory. Namely, use Lemma 3.2 to compute the components $(\boldsymbol{U}, \boldsymbol{V}^T)$ of the next iterate $\boldsymbol{X}_{\ell+1} = \boldsymbol{U}\boldsymbol{V}^T$ in factored form. Then compute a minimal rank representation of $\boldsymbol{U}\boldsymbol{V}^T$, such as via SVD or any rank-revealing factorization applied to $\boldsymbol{U}$ and $\boldsymbol{V}$. As long as the rank does not get too big, this will be faster.

An exact low-rank TAME iteration. Let $\boldsymbol{W} = \boldsymbol{F}\boldsymbol{G}^T$ be the low-rank factors of the weight matrix $\boldsymbol{W}$ and let t be the rank of the initial matrix. The key idea of low-rank TAME is to compute a rank r factorization of the iterate $\boldsymbol{X}_\ell$ and use Lemma 3.2 to compute all the r^{k-1} terms in the summation expansion to give us $\boldsymbol{X}_{\ell+1} = \boldsymbol{U}_{\ell+1}\boldsymbol{V}_{\ell+1}^T$. (This is r^2 for triangle tensors.) The low-rank terms of the next iteration are found by running a rank-revealing factorization (such as the SVD or rank-revealing QR) on $\boldsymbol{U}_{\ell+1}$ and $\boldsymbol{V}_{\ell+1}$ concatenated with the low-rank terms of the previous iterations and initial iterate (scaled by the appropriate α and β). The full procedure is detailed in Algorithm 4.2.

The dominant terms in the overall runtime of this approach for k-node motifs is $O(\mathrm{nnz}(\underline{\boldsymbol{T}}_A) + \mathrm{nnz}(\underline{\boldsymbol{T}}_B))r^{k-1} + \mathrm{RRF}(m, (r^{k-1} + r + t)) + \mathrm{RRF}(n, (r^{k-1} + r + t))$, where RRF is the cost of the rank-revealing factorization. There are many options for the RRF, including randomized and tall-and-skinny approaches. In our codes and the pseudocode we use a rank-revealing method inspired by the R-SVD (which does a QR factorization before an SVD to reduce the work in the SVD) and the structure of our problem. More on the asymptotic runtime of the R-SVD versus SVD can be found in Golub and van Loan (2013, Figure 8.6.1). Representative values of the ranks r are typically 100 and are much smaller than n or m (see more discussion in section 5.2).

Algorithm 4.2. LowRankTAME with affine shift.

Require: k-mode motif tensors $\underline{\boldsymbol{T}}_A, \underline{\boldsymbol{T}}_B$ for graph A and B, mixing parameter α, shift β, tolerance ε, weights $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{V}^T$

Ensure: Alignment heuristic $\boldsymbol{X}$ and max-weight matching of $\boldsymbol{X}$

$C = \text{trace}((\boldsymbol{V}^T\boldsymbol{V})(\boldsymbol{U}^T\boldsymbol{U}))$ ▷$C = ||\boldsymbol{W}||^2$

$\boldsymbol{U}_0 = \boldsymbol{U}/\sqrt{C}; \boldsymbol{V} = \boldsymbol{V}_0/\sqrt{C}$ ▷*Normalize first iterate*

for $\ell = 0, 1, \ldots$ until $|\lambda_{\ell+1} - \lambda_\ell| < \varepsilon$ **do**

  ▷*Exact LowRank SS-HOPM iteration*

  ▷*Compute next iterate from low-rank factors, r = num cols of $\boldsymbol{U}_\ell, \boldsymbol{V}_\ell$*

  for each i_1 in $1 \ldots r$, i_2 in $1, \ldots, r$, $\cdots$, i_{k-1} in $1, \ldots, r$ **do** ▷*Using Lemma* 3.2

    append column $\underline{\boldsymbol{T}}_A(\boldsymbol{U}_\ell(:, i_1), \ldots, \boldsymbol{U}_\ell(:, i_{k-1}))$ to $\boldsymbol{U}_{\ell+1}$

    append column $\underline{\boldsymbol{T}}_B(\boldsymbol{V}_\ell(:, i_1), \ldots, \boldsymbol{V}_\ell(:, i_{k-1}))$ to $\boldsymbol{V}_{\ell+1}$

  ▷ *Estimate tensor-eval*

  $\lambda_{\ell+1} = \text{trace}((\boldsymbol{V}_{\ell+1}^T\boldsymbol{V}_\ell)(\boldsymbol{U}_\ell^T\boldsymbol{U}_{\ell+1}))$

  ▷ *Apply affine shift in low-rank factors*

  $\boldsymbol{U}_{\ell+1} \leftarrow [\sqrt{\alpha}\boldsymbol{U}_{\ell+1} \quad \sqrt{\alpha\beta}\boldsymbol{U}_\ell \quad \sqrt{1-\alpha}\boldsymbol{U}_0]$

  $\boldsymbol{V}_{\ell+1} \leftarrow [\sqrt{\alpha}\boldsymbol{V}_{\ell+1} \quad \sqrt{\alpha\beta}\boldsymbol{V}_\ell \quad \sqrt{1-\alpha}\boldsymbol{V}_0]$

  ▷*Rank-revealing factorization: Reduce to lowest rank terms*

  $\boldsymbol{Q}_{\boldsymbol{U}}, \boldsymbol{R}_{\boldsymbol{U}} = \texttt{QR}(\boldsymbol{U}_{\ell+1}); \boldsymbol{Q}_{\boldsymbol{V}}, \boldsymbol{R}_{\boldsymbol{V}} = \texttt{QR}(\boldsymbol{V}_{\ell+1});$

  $\hat{\boldsymbol{U}}, \hat{\boldsymbol{\Sigma}}, \hat{\boldsymbol{V}}^T = \texttt{svd}(\boldsymbol{R}_{\boldsymbol{U}}\boldsymbol{R}_{\boldsymbol{V}}^{\texttt{T}})$ ▷ *Discarding near zero singular values & vectors.*

  $\boldsymbol{U}_{\ell+1} \leftarrow \boldsymbol{Q}_{\boldsymbol{U}}\hat{\boldsymbol{U}}$; $\boldsymbol{V}_{\ell+1} \leftarrow \boldsymbol{Q}_{\boldsymbol{V}}(\hat{\boldsymbol{V}}\hat{\boldsymbol{\Sigma}})$

  ▷*Normalize*

  $C = \text{trace}((\boldsymbol{V}_{\ell+1}^T\boldsymbol{V}_{\ell+1})(\boldsymbol{U}_{\ell+1}^T\boldsymbol{U}_{\ell+1}))$

  $\boldsymbol{U}_{\ell+1} \leftarrow \boldsymbol{U}_{\ell+1}/\sqrt{C}; \boldsymbol{V}_{\ell+1} \leftarrow \boldsymbol{V}_{\ell+1}/\sqrt{C};$

  $\boldsymbol{X}_{\ell+1} = \boldsymbol{U}_{\ell+1}\boldsymbol{V}_{\ell+1}^T$

  Set $t_{\ell+1}$ to be the number of motifs matched by a matching from $\boldsymbol{X}_{\ell+1}$

return $\boldsymbol{X}_\ell$ and the matching of $\boldsymbol{X}_\ell$ with the highest t_ℓ

We find that using the R-SVD finds the singular values of the low-rank components of TAME with more precision, as shown in Figure A.1.

The primary limitation to contracting with low-rank components is how much memory explicitly computing the terms requires. When $r^k < \min\{m, n\}$, then building $\boldsymbol{U}$ and $\boldsymbol{V}$ is preferable because finding the low-rank components for the next iteration can be done more efficiently and accurately than a dense $\boldsymbol{X}$. (For accuracy, see section A.1.) If $r^k > \min\{\text{nnz}(\underline{\boldsymbol{T}}_A), \text{nnz}(\underline{\boldsymbol{T}}_B)\}$, then running the original TAME implicit multiplication procedure will be faster (as can be seen in the seven clique results of Figure SM1). However, when $\min\{m, n\} \le r^k < \min\{\text{nnz}(\underline{\boldsymbol{T}}_A), \text{nnz}(\underline{\boldsymbol{T}}_B)\}$, then the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ become *wide*. In these cases, the low-rank structure itself is only beneficial in reducing overall work. We can treat $\boldsymbol{X}$ as an accumulation parameter and update it with the outer product of the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ as we compute them. This can be made more efficient by computing batches of columns, but the best batch size will be system dependent and is a level of tuning we leave to end users.

4.3. Λ-TAME. The inspiration for using SS-HOPM in TAME is that TAME's objective function (4.4) is nearby the dominant eigenvector problem for $\underline{\boldsymbol{T}}_B \otimes \underline{\boldsymbol{T}}_A$. Given the observation in Theorem 3.3 that the dominant eigenvector is built from

Algorithm 4.3. Λ-TAME.

Require: k-mode motif tensors $\underline{\boldsymbol{T}}_A, \underline{\boldsymbol{T}}_B$ for graph A and B; mixing parameter α, shift β, max iterations L

Ensure: Alignment heuristic $\boldsymbol{X}$ and max-weight matching of $\boldsymbol{X}$

1: $\boldsymbol{U}(:,1) = \frac{\mathbb{1}_m}{\sqrt{m}}; \boldsymbol{V}(:,1) = \frac{\mathbb{1}_n}{\sqrt{n}}$ ▷ Initialize first columns
2: **for** $\ell = 1, \ldots, L$ **do**
3: $\quad \boldsymbol{U}(:,\ell+1) = \underline{\boldsymbol{T}}_A \boldsymbol{U}(:,\ell)^{k-1}; \boldsymbol{V}(:,\ell+1) = \underline{\boldsymbol{T}}_B \boldsymbol{V}(:,\ell)^{k-1}$
4: $\quad \boldsymbol{U}(:,\ell+1) \leftarrow \alpha \boldsymbol{U}(:,\ell+1) + \alpha\beta \boldsymbol{U}(:,\ell) + (1-\alpha)\boldsymbol{U}(:,1)$
5: $\quad \boldsymbol{V}(:,\ell+1) \leftarrow \alpha \boldsymbol{V}(:,\ell+1) + \alpha\beta \boldsymbol{V}(:,\ell) + (1-\alpha)\boldsymbol{V}(:,1)$
6: $\quad \boldsymbol{U}(:,\ell+1) = \frac{\boldsymbol{U}(:,\ell+1)}{\|\boldsymbol{U}(:,\ell+1)\|}; \quad \boldsymbol{V}(:,\ell+1) = \frac{\boldsymbol{V}(:,\ell+1)}{\|\boldsymbol{V}(:,\ell+1)\|}$
7: **Return** $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^T$ and the matching from $\boldsymbol{X}$

the dominant eigenvectors of $\underline{\boldsymbol{T}}_B$ and $\underline{\boldsymbol{T}}_A$, this suggests a new heuristic which can be run using only the tensor powers sequences of $\underline{\boldsymbol{T}}_B$ and $\underline{\boldsymbol{T}}_A$ independently, rather than combining them as is done in TAME. We then store each of the iterates into a pair of matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ and use the information in $\boldsymbol{U}$ and $\boldsymbol{V}$ to derive the matching. There exist many possible ways to derive a matching from the iterates stored in $\boldsymbol{U}$ and $\boldsymbol{V}$ (see Nassar et al. (2018) for many low-rank ideas). We found that performing a max-weight matching on $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^T$ was the most accurate for downstream alignment tasks in our initial investigation. This is a heuristic choice. Our only ad hoc justification is that if these had been matrices, this would have been a set of inner-products among the Krylov basis. We discuss additional useful refinement of $\boldsymbol{U}$ and $\boldsymbol{V}$ in the next section. We call this method Λ-TAME because it is inspired by our dominant Z-eigenvalue theorem. Again, we adopted an affine-shift variant of the TAME method that includes an α factor to *reintroduce* the original vector into the solution. This can be set to 1 so that the iterates are exactly those from the SS-HOPM method, but there are cases where $\alpha \neq 1$ helps. In the algorithm, both $\boldsymbol{U}$ and $\boldsymbol{V}$ can be computed in time proportional to the number of nonzeros of their tensors times the total number of iterations. Like LowRankTAME, the computational bottleneck of this algorithm becomes the matching and refinement steps (see Figure 5.6).

4.4. Matching refinement. Refining the final matching is a necessary addition when using TAME, LowRankTAME, or Λ-TAME. For each method, the result is both a low-rank matrix $\boldsymbol{X}^*$, along with the rank factors $\boldsymbol{U}, \boldsymbol{V}$, and a maximum weight matching computing on this matrix. In the original TAME method, the matrix $\boldsymbol{X}$ was improved by computing a maximum weight bipartite matching and then by looking locally for potential match swaps which montonically increase triangles aligned. Another approach using with a low-rank method is to use the information and matching produced to initialize and guide a more expensive network alignment method, such as Klau's algorithm (Klau, 2009), similar to what was done in Nassar et al. (2018).

TAME's *b*-matching local search refinement TAME's refines its produced matching by constructing local neighborhoods of nodes and looking for substitutes in its current matchings that increase the number of triangles aligned (or increases the number of edges while maintaining the triangles aligned). The authors construct a b-matching from the matrix $\boldsymbol{X}^*$ returned by TAME (using the 2-approximation algorithm (Khan et al., 2016)) and search the found matchings along with neighbor

substitutions as local neighborhoods. Each edge (i, i') in the matching, in order of their edge weight, searches the set of alternative matches

$$\left\{(i,j') \middle| \begin{array}{c}(i,j') \in \texttt{b-matching}(\boldsymbol{X}^*), \text{ or} \\ j' \text{ is connected to } i' \text{ in graph } B\end{array}\right\} \cup \left\{(j,i') \middle| \begin{array}{c}(j,i') \in \texttt{b-matching}(\boldsymbol{X}^*), \text{ or} \\ j \text{ is connected to } i \text{ in graph } A\end{array}\right\}$$

for a possible replacement, and immediately makes changes which improve the alignment. The full procedure is outlined in Mohammadi et al. (2017, section 4.5, Algorithm 4). The original method ascribes weights to the edges and triangles using the weights in the iterate returned by TAME, but our method doesn't weight the triangles or edges when measuring the change in alignment quality. The greedy swapping procedure can be run multiple times, but improvements tend to stop after 5 to 10 successive sweeps over all matched edges.

A new nearest neighbor local search refinement. The low-rank structure of $\boldsymbol{X}^*$ suggests an alternative to b-matching, for which even the 2-approximation is computationally costly on a large, dense matrix $\boldsymbol{X}^*$. Rather than b-matching, we treat the low-rank structure $\boldsymbol{X}^* = \boldsymbol{U}\boldsymbol{V}^T$ as an *embedding* of each vertex where rows of $\boldsymbol{U}$ give coordinates for each vertex in graph A and rows of $\boldsymbol{V}$ give coordinates for each vertex in graph B. Then we consider nearby vertices as alternative matches. For this task, a K nearest neighbors methodology applies. Each row of $\boldsymbol{U}$ embeds $i \in V_A$, so the rows of $\boldsymbol{U}$ which are close to $\boldsymbol{U}(i,:)$ in the 2-norm distance define a natural neighborhood of i. This leads us to construct sets of the form

$$\left\{(i,j') \middle| \begin{array}{c}j' \in K\texttt{-nearest}(\boldsymbol{V}(i',:), \boldsymbol{V}), \text{ or} \\ j' \text{ is connected to } i' \text{ in graph } B\end{array}\right\} \cup \left\{(j,i') \middle| \begin{array}{c}j \in K\texttt{-nearest}(\boldsymbol{U}(i,:), \boldsymbol{U}), \text{ or} \\ j \text{ is connected to } i \text{ in graph } A\end{array}\right\}$$

to search for changes to the matchings. Ball-trees are particularly suitable for finding close neighbors of points in low dimensional spaces and are empirically faster than b-matching with superior results.

Improving matchings with Klau's algorithm. Klau's algorithm (Klau, 2009) is an edge based graph matching/ network alignment method that uses a sequence of maximum weighted matchings to iterate toward a better solution. It can, in some instances, identify optimal solutions of the NP-hard graph matching objective with a corresponding proof of optimality. The algorithm is built from a Lagrangian decomposition of a tight linear program relaxation of the graph matching IQP (a weighted form of Definition 4.2). A full explanation of the algorithm can be found in Bayati et al. (2013, section 4.3). The primary input for Klau's method are the graphs A, B and a weighted bipartite graph between the vertex set of A and B that restricts and biases the set of possible alignments. The adjacency matrix of this bipartite graph is $\boldsymbol{L}$ and is called the *link matrix* or prior matrix. The method is most effective when $\boldsymbol{L}$ has only a few choices for alignments between the graphs.

Thus, we use the results of LowRankTAME or Λ-TAME to build $\boldsymbol{L}$. We include the matched edges within $\boldsymbol{L}$ and then expand using the neighborhoods of the matched nodes (much like TAME's local search). In Nassar et al. (2018), Klau's method was more accurate when given expanded results of b-matching. Given the low-rank structure of our methods, we further expand $\boldsymbol{L}$ by including edges in the found matchings with the k closest neighbors of (i, i') in their respective embedding spaces $\boldsymbol{U}$ and $\boldsymbol{V}$.

5. Empirical comparisons in our network alignment application. The major demonstration of the new Kronecker product theory is in terms of its impact on network alignment algorithms described in the previous sections. We have implementations of TAME which compute contractions using the original implicit

form and new versions using our and existing tensor Kronecker theory. These are all generalized to work with any order motif. We focus on cliques as the motif. We use TuranShadow (Jain and Seshadhri, 2017) to sample the network for cliques at random. Equivalently, we use cliques to induce a hypergraph where the nodes are the same and the cliques are hyperedges. Our codes are implemented in Julia and are available from https://www.cs.purdue.edu/homes/ccolley/project_pages/TensorKroneckerProducts.html. In this section, we validate the algorithms and show we can achieve similar results with greatly improved runtimes. Some highlights of our results are the following:

1. Iterates of TAME are low rank on real and synthetic data and LowRankTAME computes them an order of magnitude faster for small enough motifs (section 5.2).
2. The Λ-TAME vector information can be produced quickly for any size motif.
3. When the Λ-TAME vector information is refined using the nearest neighbor information and Klau's algorithm, it aligns more triangles and edges than the refined TAME information. Also, it has end to end runtimes 1 to 2 orders of magnitude faster than the C++ TAME implementation (section 5.4).

We use all the same parameters as the original research where they were accessible and will discuss our reasoning for our choices for unlisted parameters. Our experiment environment uses Intel Xeon Platinum 8168 CPU (@ 2.70 GHz) processors with 24 cores, although none of our methods use multicore parallelism. We compare our methods against one another as well as LowRankEigenAlign (Nassar et al., 2018). LowRankEigenAlign utilizes a low-rank structure discovered in the EigenAlign (Feizi et al., 2019) algorithm and improves its scalability with minimal changes or even sometimes improvements to accuracy. LowRankEigenAlign has been tested on similar real-world and synthetic alignment problems, and its low-rank structure makes it a comparable method in terms of memory to Λ-TAME. LowRankEigenAlign also gives a low-rank embedding which allows us to refine its results in a manner similar to Λ-TAME and LowRankTAME.

5.1. Data for network alignment experiments. There are two types of data that we use in evaluating the new network alignment algorithms. The first is a subset of the LVGNA (Vijayan and Milenković, 2017) PPI graph collection. Each pair of networks in this collection gives an alignment problem. Network statistics are in the supplemental materials (Table A.1). Each vertex represents a protein and the edges represent interactions. The networks range in size from 2871 to 16060 vertices and all but the largest networks have fewer triangles than edges.

The second type of data involves synthetic random geometric graphs. To generate a random geometric graph, we randomly sample n points in the unit square. Then each point adds undirected edges to the k nearest neighbors, where k is drawn from a log-normal distribution centered at $\log 5$ with $\sigma = 1$. We then create a pair of networks to align from this starting reference graph by independently perturbing them from a noise model. The two noise models we consider are (i) a microbiological inspired partial duplication procedure (Bhan, Galas, and Dewey, 2002; Chung et al., 2003; Hermann and Pfaffelhuber, 2014) and (ii) the Erdős–Rényi noise model from Feizi et al. (2019, section 3.4). When using the duplication noise mode, we incrementally duplicate 25% new nodes in the network, copying their existing edges with probability $p_{edge} = .5$. For the Erdős–Rényi noise we randomly delete edges with probability $p = .05$ and randomly add in edges with probability $q = \frac{p\rho}{1-\rho}$, where ρ is the density of

the network. A few experiments have different choices for these parameters, which will be explicitly noted. We further randomize the permutation of the perturbed network to avoid any influences due to node order in what might happen in the presence of tied values. (Prior work and experience have shown a startlingly strong effect due to biases when this permutation step is not present.) Within each experiment, algorithms are always tested on exactly the same set of networks instead of separate draws from the same distribution.

5.2. Low-rank structure in TAME. For our first set of experiments, we want to show that the iterations from TAME (Algorithm 4.1) remain low rank when we start with a uniform or unbiased iterate as the weight matrix: $\boldsymbol{X}_0 = \frac{1}{mn}\mathbb{1}_m\mathbb{1}_n^T$, which is rank 1. We further investigate this behavior on larger motifs. To do this, we report matrix rank using the LowRankTAME algorithm instead of the raw TAME algorithm, which are identical in exact arithmetic (see Aside 1).

Given either a pair of networks from LVGNA or a synthetic alignment problem, we plot the maximum rank from any iteration on any trial ($\alpha \in \{.5, 1.0\}$ running 15 iterations) as determined by the rank function in Julia, as we vary the β parameter in the alignment problem. (See Figure 5.1.) These results, along with trendlines for the maximum rank over multiple repetitions of the synthetic experiments, show that the rank is often below 250 even though the largest networks have 10k vertices.

ASIDE 1. *This choice of exact LowRankTAME vs. TAME to evaluate rank is made both because it is faster to compute but also because preliminary experiments showed that TAME caused the finite precision rank to grow even when the result is mathematically rank* 1 *(*$\alpha = 1.0$, $\beta = 0.0$, *by lemma* 3.1*). This is well-known to happen to finite precision computations, for instance in the power method. Details of this test are in the App.* A.1.

Our rank experiments show the synthetic problems have higher ranks than the LVGNA collection. For the LVGNA collection, large problems tend to have smaller rank, whereas we do see the rank grow with the size of the synthetic problems. For the synthetic problems, we also see that increasing β produces higher ranks because these problems incorporate more of the previous iterate via an affine shift. The behavior with $\beta = 100$ for the LVGNA collection is rather different, with many small dips. On further investigation, we found this occurs because $\beta = 100$ is nearly an eigenvalue of these problems. We verified in subsequent experiments that shifting by the estimated eigenvalue gives very small rank, although we do not report these experiments in the interest of space.

In Figure 5.2, we investigate rank behavior for larger clique motifs. We see that though the rank of iterates does not change dramatically (second column of density plots), the runtime of TAME and LowRankTAME consistently grows (blue and red density plots in the third column), even when the number of motifs within the networks declines (first column of density plots). As the motif size grows, the time spent using the low-rank contraction routines approaches the runtime of TAME's implicit contraction. This becomes salient when memory constraints require a user to use the accumulation form of LowRankTAME, as then even the low-rank components are found from a dense matrix $\boldsymbol{X}_\ell$, rather than being able to benefit from two R-SVD calls. In summary, Figure 5.2 indicates that for small motifs (less than size 6), we can improve the runtime and accuracy using the contraction theory shown in this paper, but those benefits are reduced or even eliminated for problems with larger motifs.

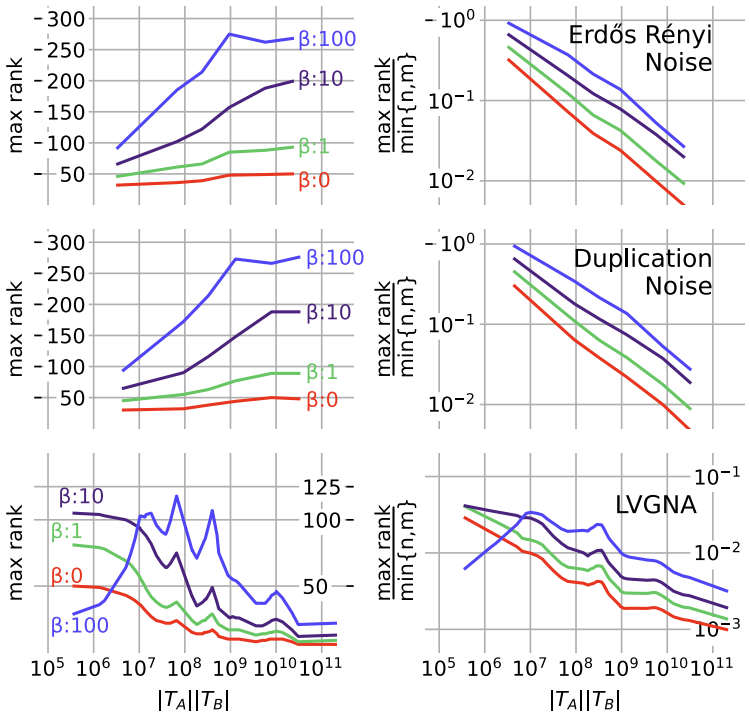


FIG. 5.1. *Both real-world and synthetic results (reference graphs are generated with* 100, 500, 1000, 2000, 5000, *and* 10000 *vertices) are low rank with respect to the size of the networks. We compute the maximum rank over any iterate from runs with any the affine shift values* $\alpha = 0.5, 1.0$*, and we plot the maximum rank directly for the synthetic networks and loess smoothing trendlines (using* 30% *approximate neighbors) for the LVGNA experiments. The maximum rank of any iterate over synthetic network alignment problems were consistently higher than PPI problems, but both are low when put in the context of their maximum possible ranks (right-hand plots). The similarity of results between the two noise models is expected as they start with the same reference graph. We generally see that rank increases as* β *increases except for* $\beta = 100$*, which is discussed in the text.*

5.3. Alignment accuracy in synthetic networks. The next set of experiments transitions from runtime to accuracy where we test how well the best low-rank results produced by the TAME method, Λ-TAME, and LowRankEigenAlign can be refined by local search and Klau's algorithm using the K-nearest neighbor strategy. We focus on the synthetic problems where there is a single reference graph that is subject to two independent perturbations. The goal is to find the alignment between the vertices of the original reference graph, which we regard as the *correct* answer. Each combination of methods is compared using the accuracy

$$\text{accuracy} = \frac{\text{number of aligned pairs of vertices from the reference graph}}{\text{total number of vertices in the reference graph}}$$

and their triangle alignment score (how many triangles they match compared to the maximum possible). We use max iterations $L = 15$, stopping tolerance $\varepsilon = 10^{-6}$, and $K = 2 * \texttt{rank}(\boldsymbol{X}^*)$ throughout the experiments—except in figures where K is varied. We focus on our experiments which vary the size of alignment problems. Additional parameters of our noise models are studied in supplement SM3.2.

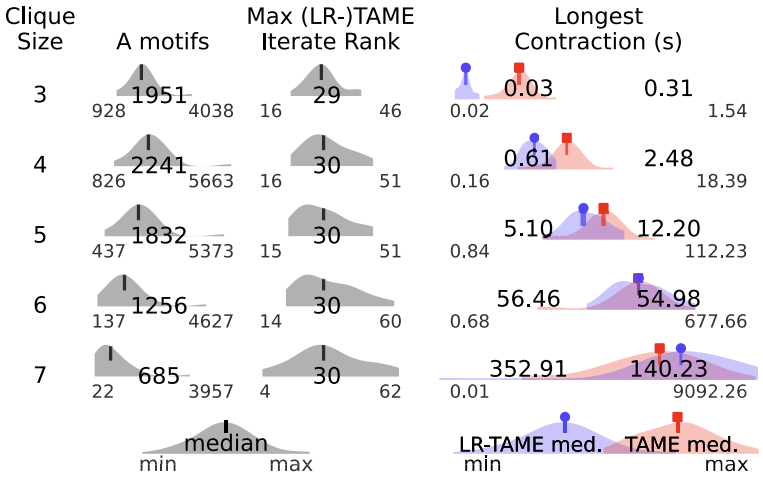


FIG. 5.2. *Experiments on rank from synthetic experiments where the reference graph has* 100 *vertices and is perturbed with* 20% *duplicated nodes (instead of the default* 25%*). We compute statistics over* 25 *trials and* 15 *iterations of each method (LR-TAME for LowRankTAME and "med." for median). The figures show density plots of the worst results over all the trials. A more detailed analysis of our data can be found in Figure* SM1*. Exploiting the low-rank structure is most effective for small motifs. We see that time spent computing contractions for TAME and LowRankTAME grows as the motif size increases, even though the rank of TAME's iterates and number of motifs found within the reference network decline. As a point of comparison,* Λ*-TAME's runtime is reasonably constant across each experiment and the longest* Λ*-TAME contraction time of any trial was* 0.0132*s.*

The first set of experiments focuses on triangles. These experiments show that all three methods require refinement to get practical results, especially as the problems get larger (Figure 5.3). These experiments show that LowRankTAME with the local search strategy K-NN had the best performance for the largest problems, although Λ-TAME with Klau's refinement was slightly better at intermediate sized problems for the duplication noise model.

We can also see that triangles matched is a good proxy for the accuracy of the matchings, although depending on the noise models, there may be deviations.

Moreover, with the K-NN refinement and Λ-TAME 's scalability, we can get fast accurate matchings for not only large networks, but also increasingly larger clique sizes (Figure 5.4). Accuracy remains high when the vertex coverage, the fraction of the total vertex set involved in motifs, remains high. In contrast to the results with LowRankTAME (Figure 5.2), using Λ-TAME has a practical runtime for large cliques. Klau's algorithm can offer an additional benefit if a longer runtime can be tolerated.

We also find that our default choice of K gets good performance. Increasing K can improve accuracy slightly, but Klau's algorithm runtime is more sensitive to the sparsity of the input link matrix. Local search remains very fast with a modest increase in runtime as the local search neighborhoods are expanded. It's unsurprising to see that Klau's algorithm matches the fewest motifs given the objective function is focused on aligning edges between networks.

5.4. Biological networks. We now turn our attention to considering the performance within real-world networks from biology. We report the end to end runtime,

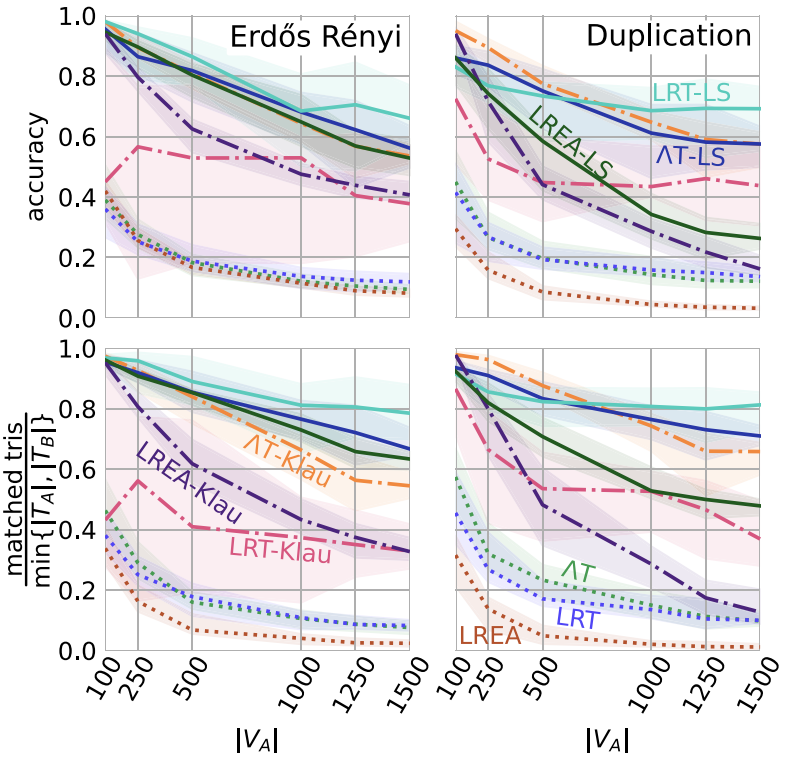


FIG. 5.3. *We consider aligning two independent perturbations of a single reference graph using either the Erdős–Rényi model (left column) or the duplication noise model (right column) based on matching triangles. We compare three methods, LowRankTAME (LRT), Λ-TAME (ΛT), and LowRankEigenAlign (LREA), with three refinement schemes: None, Klau, and local search (LS). Across all methods, the ground truth accuracy (top row), which is generally not known, is closely aligned with the number of matched triangles (bottom row), which is easy to compute, suggesting that the latter is a useful proxy. This shows that refinement is an important step as the methods without refinement have dramatically lower accuracy (fine dots) than either local search (solid lines) or Klau (dash-dots). We plot the median of* 20 *trials with* 20*th–*80*th percentile ribbons.*

triangles matched, and edges matched of each refined method relative to TAME, over pairs of alignment problems from LVGNA in Figure 5.5 (see section SM3.3 for non-comparative results). We use max iterations $L = 15$, stopping tolerance $\varepsilon = 10^{-6}$, and nearest neighbors $K = 2 * \texttt{rank}(\boldsymbol{X}^*)$ for refinement. We see that Λ-TAME refined with local search aligns more triangles and edges and runs much faster than the original TAME. All methods tested align more edges than TAME, though improving with local search was much more likely to increase triangles matched. Methods using Klau's algorithm or LowRankEigenAlign increased the number of edges aligned, but aligned fewer triangles. This was observed in the synthetic results in Figure 5.3 and is unsurprising given each method's focus on edges.

Returning to timing, we compare the fraction of time spent in the matching versus matrix/ tensor operations in Figure 5.6. We can see that the Kronecker theory here makes the time to compute the contractions fast enough to change the primary bottleneck of the algorithm when using triangle adjacency tensors. Put simply, Λ-TAME always spends more time on the bipartite matching and refinement and LowRankTAME spends more time there on the biggest problems. A primary reason

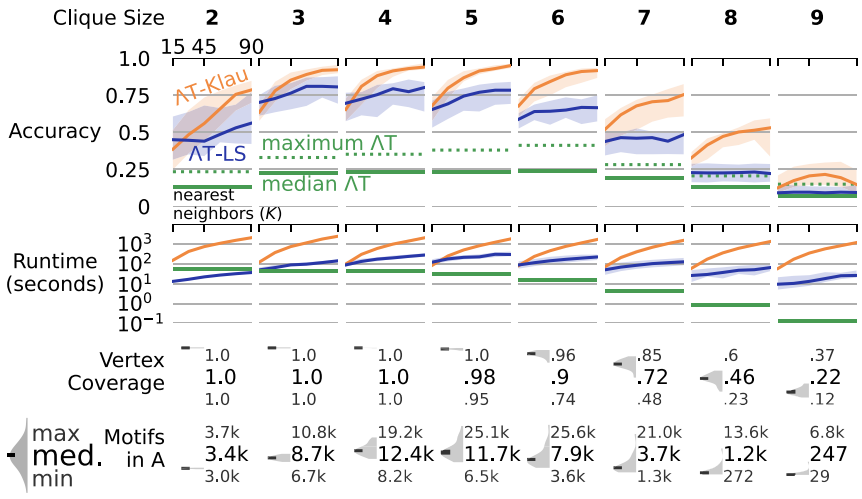


FIG. 5.4. *We consider the same scenario for Figure* 5.3 *but now look at aligning networks based on cliques of size* 2 (*edges*), *size* 3 (*triangles*), *and up to size* 9 *on networks with* 500 *vertices in the duplication model. Cliques are sampled using TuranShadow with* 10^6 *samples, which will find the vast majority. We focus on the* Λ*-TAME* (Λ*T*) *method as LowRankTAME would take a prohibitively long time* (*days*). *We also vary the number of nearest neighbors considered in the refinement step for both Klau and local search* (*LS*) *in the horizontal piece of the microplots to understand that behavior, as well as its impact on runtime* (*second row*). *The top row* (*accuracy*) *shows that accuracy declines after the clique size is larger than* 5 *or* 6 *for either refinement strategy. To understand this behavior, we look at the total number of motifs found* (*bottom row*) *and the vertex coverage of those motifs* (*third row*). *These are shown as density plots with the max, min, and median values shown. This shows that accuracy declines once the vertex coverage begins to decline.*

for why Λ-TAME and LowRankTAME spend so much time computing the matchings is that we cannot take advantage of the low-rank structure. We compute the maximum matching using the primal-dual algorithm (Dantzig, Ford, and Fulkerson, 1956), which must touch each entry of $\boldsymbol{X}^* = \boldsymbol{U}\boldsymbol{V}^T$ at least once, making it more efficient to form $\boldsymbol{X}^*$ explicitly at the beginning of the algorithm. This suggests potential future research for new algorithms which can properly make use of the low-rank structure, while still computing a maximum weighted matching.

6. Discussion. The major focus of our paper is on demonstrating how the theory on tensor Kronecker products in section 3 enables us to accelerate the graph matching algorithm TAME (section 4): (i) by making the same algorithm faster with LowRankTAME, (ii) by giving a new, faster algorithm (Λ-TAME), and (iii) by providing new augmentations of TAME's local search and Klau's algorithm which can make use of the low rank structure within the iterates.

One interesting theory question we have not pursued is the opposite of the example from the introduction that shows diagonal tensors have eigenvectors that are not a Kronecker product. Put concretely, is there a class of tensors where no new eigenvectors emerge after taking a Kronecker product? It would also be worth considering if and how Theorem 3.3 generalizes to H-eigenvalues or even generalized tensor eigenvalues. Our current approach is not sufficient as the proof of Theorem 3.3 relies on Zhang, Ling, and Qi (2012, Theorem 2.1). It is possible that there could be types of tensor eigenvalues to which this theorem does not generalize.

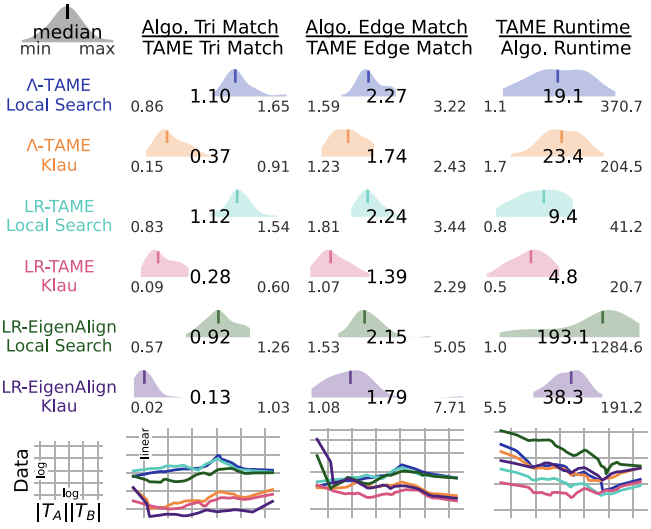


FIG. 5.5. *For networks in the LVGNA collection, we compare the number of triangles (left column), edges (middle column), and runtime (right column) between the low-rank methods and the original TAME method (including its end-to-end b-matching refinement time). These are shown as density plots over all* 45 *pairwise alignment problems. Larger values and values larger than* 1 *are better for all experiments. The final row shows the Loess-smoothed plot of the raw data against the problem size, which shows minimal size-dependent effects—beyond those expected due to runtime. Note that* Λ*-TAME with local search consistently aligns more triangles and edges than TAME while running about* 20 *times faster. Refining with LocalSearch tends to be faster than using Klau's algorithm, though we expect the sparsity of the input matrices to be the same. These experiments show that refinement can be very problem dependent and local search is particularly successful here.*

On the application side, the new method Λ-TAME's runtime is heavily dominated by the rounding and refinement procedures, as seen in Figure 5.6. Our implementation uses the primal-dual algorithm which is an effective solution when $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}^T$ is explicitly realized as a dense matrix. New matching methods which compute the maximum matching of $\boldsymbol{X}$ while only using $\boldsymbol{U}$ and $\boldsymbol{V}$ would be useful to improve scalability (even if only an approximation). For large enough problems, there are also low-rank matching heuristics from Nassar et al. (2018) to consider for additional scalability, although the results from these methods were noticeably worse for our case compared with using the exact max-weight matching.

The low-rank structure offers a few benefits even beyond the reduced runtime. First, we are able to explicitly *store* a large number of TAME iterates as low-rank factorizations. Sending $O(mr)$ data is much faster between cores, which may offer a foothold in known parallel matching challenges (Sathe, Schenk, and Burkhart, 2012; Bertsekas and Castañon, 1991). Applying Theorem 3.3 recursively suggests an immediate algorithm for multinetwork alignments. Each network's embeddings can be computed independently in a fashion similar to that used in (Nassar et al., 2021). Furthermore, we also see opportunities to incorporate multiple network motifs within adjacency tensors. Smaller motifs could be encoded into the off diagonal components (see Aside 2). Motif complexes could be encoded into the off diagonal components in a way that wouldn't change contraction or eigenvector definitions.

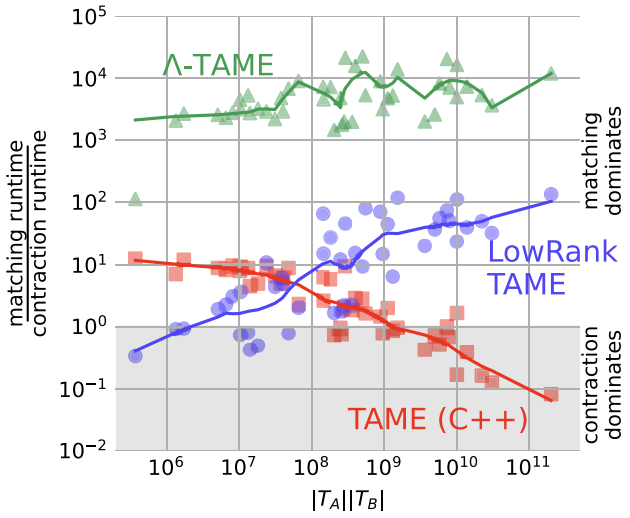


FIG. 5.6. *We compare the time spent working on tensor-vector multiplication/contraction compared with the time spent rounding* $\boldsymbol{X}_\ell$. *These show that the new bottleneck of* Λ*-TAME and LowRankTAME is the time spent on rounding the continuous iterates to discrete matchings, in contrast to the original TAME method.*

The fashion in which we construct the embeddings is also closely related to various graph kernels (Vishwanathan et al., 2010; Kriege, Johansson, and Morris, 2020), including the random walk kernel on a direct product graph. Graph kernels have long been used to align small chemicographs (graphs that represent small chemical molecules). In this case, we are able to generate a direct factorization of a graph kernel between vertices of two graphs into a product of features on each graph. This is a common paradigm (Vishwanathan et al., 2010) involving matrix Kronecker products—although we are unaware of any research on this for higher-order analogues of the graph kernels involving tensor Kronecker products that would be needed for our perspective. When viewed in this light, our research has the potential to open new directions in this space in terms of efficient graph kernels on hypergraphs.

ASIDE 2. *Tensors can have more than one* "diagonal" *by grouping non-zeros by the multiplicity of their indices. In a triangle adjacency tensor the non-zeros are of the form* (i,j,k) *for distinct vertices and the traditional diagonal is comprised of the indices* (i,i,i). *A third order tensor also has entries which only have two unique vertices, and the presence of an edge could be marked in an entry of the form* (i,i,j) *or* (i,j,j). *These off diagonals are referred to as q-multiplicity tensors in (Yan et al.,* 2015, *Def.* 2*).*

In summary, our theory and experiments show how the computational demands of methods with tensor Kronecker products may be reduced by orders of magnitude with no change in quality, or accelerated even further with useful approximate results. We are excited about future opportunities with tensor Kronecker products due to the widespread use of matrix Kronecker products, and we suspect that these theorems, or alternative generalizations that use a specific structure in novel problems, will be a key element in this future research.

Appendix A. Additional information.

A.1. TAME rank 1 singular value experiments. These experiments explain why we use the exact LowRankTAME iteration instead of the original TAME iteration to study rank when using triangle adjacency tensors. They show TAME produces iterates that would be detected as at least rank 2 even when the answer is provably rank 1, whereas LowRankTAME does not. Figure A.1 plots the maximum second largest normalized singular values of $\boldsymbol{X}_\ell$, σ_2 of all 15 iterations for TAME and LowRankTAME of the rank 1 iteration case for the LVGNA and our synthetic alignments. Hollow points are values small enough to be considered zero (and hence would be rank 1), and filled points are large enough to be measured as nonzero (and hence would be rank 2). The LVGNA experiments align all pairs of distinct networks. The synthetic experiments are measured over 50 trials using random geometric graphs. Seeded networks are perturbed by both ER and duplication noise models using default parameters. The difference in singular values arises because TAME works with a vectorized representation of a matrix, which slowly drifts off the exact rank 1 manifold in finite precision arithmetic, whereas LR-TAME maintains a minimal rank representation because of the lack of shifts. This phenomenon is similar to how the power method when started from a vector numerically orthogonal to the dominant eigenvector nonetheless may converge to the dominant eigenvector due to loss of exact orthogonality over the iterations.

A.2. PPI graph statistics. We use networks from the LVGNA project (Meng, Striegel, and Milenković, 2016), the statistics of which (unique edges and triangles) are given in Table A.1. These networks have been aligned with a variety of contemporary methods in Vijayan and Milenković (2017), Meng, Striegel, and Milenković (2016), and Nassar et al. (2018) to make our results comparable with prior research. We remove any directional edges from the network before the enumerating triangles. As our methods are focused on triangle motifs, we use only networks with more than 150 triangles. We also include the largest sampled z-eigenvalue found, which—like the standard power method—is related to the behavior of the methods with shifts in section 5.2.

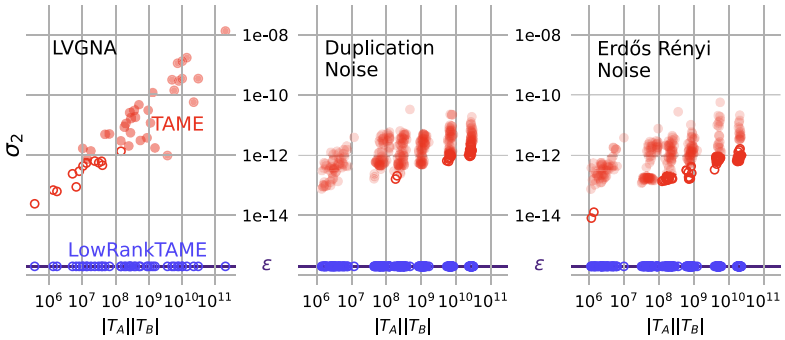


FIG. A.1. *LowRankTAME accurately captures the rank structure when it is provably rank* 1 *for triangle adjacency tensors. TAME uses an implicit contraction that frequently produces iterates* $\boldsymbol{X}_\ell$ *with nondominant singular values large enough to be nonzero for matrices which are provably rank* 1 ($\alpha = 1.0$, $\beta = 0.0$, *by Lemma* 3.1).

TABLE A.1
LVGNA network statistics.

Graph name	Vertices	Edges	Triangles	Sampled λ
worm_Y2H1	2871	5194	536	10.076
worm_PHY1	3003	5501	692	12.664
fly_Y2H1	7094	23356	2501	21.207
yeast_Y2H1	3427	11348	9503	56.680
human_Y2H1	9996	39984	15177	72.919
human_PHY2	8283	19697	19190	50.872
yeast_PHY2	3768	13654	26295	94.564
fly_PHY1	7885	36271	58216	217.541
yeast_PHY1	6168	82368	381812	454.921
human_PHY1	16060	157649	525238	488.136

Acknowledgments. We thank O. Eldaghar for the fruitful conversations when designing our figures and C. Cui for discussions on the code from (Cui, Dai, and Nie, 2014).

REFERENCES

L. AKOGLU, M. MCGLOHON, AND C. FALOUTSOS (2008), *RTM: Laws and a recursive generator for weighted time-evolving graphs*, in Proceedings of ICDM, pp. 701–706.

B. W. BADER AND T. G. KOLDA (2008), *Efficient MATLAB computations with sparse and factored tensors*, SIAM J. Sci. Comput., 30, pp. 205–231.

K. BATSELIER AND N. WONG (2017), *A constructive arbitrary-degree kronecker product decomposition of tensors*, Numer. Linear Algebra Appl., 24, e2097.

M. BAYATI, D. F. GLEICH, A. SABERI, AND Y. WANG (2013), *Message-passing algorithms for sparse network alignment*, ACM Trans. Knowl. Discov. Data., 7, pp. 3:1–3:31, https://doi.org/10.1145/2435209.2435212.

M. BAZARAA AND O. KIRCA (1983), *A branch-and-bound-based heuristic for solving the quadratic assignment problem*, Nav. Res. Logist. Quart., 30, pp. 287–304.

A. R. BENSON AND D. F. GLEICH (2019), *Computing tensor z-eigenvectors with dynamical systems*, SIAM J. Matrix Anal. Appl., 40, pp. 1311–1324.

A. R. BENSON, D. F. GLEICH, AND J. LESKOVEC (2015), *Tensor spectral clustering for partitioning higher-order network structures*, in Proceedings of SDM, pp. 118–126, https://doi.org/10.1137/1.9781611974010.14.

A. C. BERG, T. L. BERG, AND J. MALIK (2005), *Shape matching and object recognition using low distortion correspondences*, in Proceedings of CVPR, pp. 26–33.

D. P. BERTSEKAS AND D. A. CASTAÑON (1991), *Parallel synchronous and asynchronous implementations of the auction algorithm*, Parallel Comput., 17, pp. 707–732, https://doi.org/10.1016/S0167-8191(05)80062-6.

A. BHAN, D. J. GALAS, AND T. G. DEWEY (2002), *A duplication growth model of gene expression networks*, Bioinformatics, 18, pp. 1486–1493.

V. D. BLONDEL, A. GAJARDO, M. HEYMANS, P. SENELLART, AND P. VAN DOOREN (2004), *A measure of similarity between graph vertices: Applications to synonym extraction and web searching*, SIAM Rev., 46, pp. 647–666.

R. BURKARD, M. DELL'AMICO, AND S. MARTELLO (2012), *Assignment Problems*, revised ed., SIAM, Philadelphia, https://doi.org/10.1137/1.9781611972238.

D. CARTWRIGHT AND B. STURMFELS (2013), *The number of eigenvalues of a tensor*, Linear Algebra Appl., 438, pp. 942–952.

M. CHERTOK AND Y. KELLER (2010), *Efficient high order matching*, IEEE Trans. Pattern Anal. Mach. Intell., 32, pp. 2205–2215.

F. CHUNG, L. LU, T. G. DEWEY, AND D. J. GALAS (2003), *Duplication models for biological networks*, J. Comput. Biol., 10, pp. 677–687.

D. CONTE, P. P. FOGGIA, C. SANSONE, AND M. VENTO (2004), *Thirty years of graph matching in pattern recognition*, Intern. J. Pattern. Recognit. Artif. Intell, 18, pp. 265–298, https://doi.org/10.1142/S0218001404003228.

C.-F. Cui, Y.-H. Dai, and J. Nie (2014), *All real eigenvalues of symmetric tensors*, SIAM J. Matrix Anal. Appl., 35, pp. 1582–1601.
G. B. Dantzig, L. R. Ford, Jr., and D. R. Fulkerson (1956), *A Primal-Dual Algorithm*, Technical report, RAND, Santa Monica, CA.
L. De Lathauwer, P. Comon, B. De Moor, and J. Vandewalle (1995), *Higher-order power method*, in Proceedings of Nonlinear Theory and Its Applications, NOLTA '95, pp. 91–96.
N. Eikmeier, A. S. Ramani, and D. F. Gleich (2018), *The hyperkron graph model for higher-order features*, in Proceedings of the International Conference on Data Mining, pp. 941–946, https://doi.org/10.1109/ICDM.2018.00115.
E. Estrada and J. A. Rodríguez-Velázquez (2006), *Subgraph centrality and clustering in complex hyper-networks*, Phys. A, 364, pp. 581–594.
S. Feizi, G. Quon, M. Mendoza, M. Medard, M. Kellis, and A. Jadbabaie (2019), *Spectral alignment of graphs*, IEEE Trans. Network Sci. Eng., 7, pp. 1182–1197.
G. H. Golub and C. van Loan (2013), *Matrix Computations*, Johns Hopkins University Press, Baltimore.
F. Hermann and P. Pfaffelhuber (2014), *Large-Scale Behavior of the Partial Duplication Random Graph*, preprint, https://arxiv.org/abs/1408.0904.
A. Jaffe, R. Weiss, and B. Nadler (2018), *Newton correction methods for computing real eigenpairs of symmetric tensors*, SIAM J. Matrix Anal. Appl., 39, pp. 1071–1094.
S. Jain and C. Seshadhri (2017), *A fast and provable method for estimating clique counts using Turán's theorem*, in Proceedings of the 26th International Conference on World Wide Web, pp. 441–449.
S. Jain and C. Seshadhri (2020), *The power of pivoting for exact clique counting*, in Proceedings of WSDM, pp. 268–276.
A. Khan, A. Pothen, M. Mostofa Ali Patwary, N. R. Satish, N. Sundaram, F. Manne, M. Halappanavar, and P. Dubey (2016), *Efficient approximation algorithms for weighted b-matching*, SIAM J. Sci. Comput., 38, pp. S593–S619.
G. W. Klau (2009), *A new graph-based method for pairwise global network alignment*, BMC Bioinform., 10, S59.
C. Klymko, D. Gleich, and T. G. Kolda (2014), *Using Triangles to Improve Community Detection in Directed Networks*, preprint, https://arxiv.org/abs/1404.5874.
T. G. Kolda and J. R. Mayo (2011), *Shifted power method for computing tensor eigenpairs*, SIAM J. Matrix Anal. Anal., 32, pp. 1095–1124.
T. G. Kolda and J. R. Mayo (2014), *An adaptive shifted power method for computing generalized tensor eigenpairs*, SIAM J. Matrix Anal. Appl., 35, pp. 1563–1581.
G. Kollias, S. Mohammadi, and A. Grama (2011), *Network similarity decomposition (NSD): A fast and scalable approach to network alignment*, IEEE Trans. Knowl. Data Eng,, 24, pp. 2232–2243.
T. C. Koopmans and M. Beckmann (1957), *Assignment problems and the location of economic activities*, Econometrica, pp. 53–76.
N. M. Kriege, F. D. Johansson, and C. Morris (2020), *A survey on graph kernels*, Appl. Network Sci., 5, https://doi.org/10.1007/s41109-019-0195-3.
E. L. Lawler (1963), *The quadratic assignment problem*, Management Sci., 9, pp. 586–599.
L.-H. Lim (2005), *Singular values and eigenvalues of tensors: A variational approach*, in Proceedings of the 1st International Workshop on Computational Advances in Multi-sensor Adaptive Processing, pp. 129–132.
N. Malod-Dognin and N. Pržulj (2015), *L-GRAAL: Lagrangian graphlet-based network aligner*, Bioinformatics, 31, pp. 2182–2189.
L. Meng, A. Striegel, and T. Milenković (2016), *Local versus global biological network alignment*, Bioinformatics, 32, pp. 3155–3164.
R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon (2002), *Network motifs: Simple building blocks of complex networks*, Science, 298, pp. 824–827.
S. Mohammadi, D. F. Gleich, T. G. Kolda, and A. Grama (2017), *Triangular alignment TAME: A tensor-based approach for higher-order network alignment*, IEEE/ACM Trans. Comput. Biol. Bioinform., 14, pp. 1446–1458.
H. Nassar, G. Kollias, A. Grama, and D. F. Gleich (2021), *Scalable algorithms for multiple network alignment*, SIAM J. Sci. Comput., 43, pp. S592–S611, https://doi.org/10.1137/20m1345876.
H. Nassar, N. Veldt, S. Mohammadi, A. Grama, and D. F. Gleich (2018), *Low rank spectral network alignment*, in Proceedings of the 2018 World Wide Web Conference, pp. 619–628.
S. Park, S.-K. Park, and M. Hebert (2013), *Fast and scalable approximate spectral matching for higher order graph matching*, IEEE Trans. Pattern Anal. Mach. Intell., 36, pp. 479–492.
R. Patro and C. Kingsford (2012), *Global network alignment using multiscale spectral signatures*, Bioinformatics, 28, pp. 3105–3114.

A. H. Phan, A. Cichocki, P. Tichavsky, D. P. Mandic, and K. Matsuoka (2012), *On revealing replicating structures in multiway data: A novel tensor decomposition approach*, in Proceedings of LVA/ICA, pp. 297–305.

A. H. Phan, A. Cichocki, P. Tichavsky, R. Zdunek, and S. Lehky (2013), *From basis components to complex structural patterns*, in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3228–3232.

L. Qi (2005), *Eigenvalues of a real supersymmetric tensor*, J. Symbolic Comput., 40, pp. 1302–1324.

L. Qi and Z. Luo (2017), *Tensor Analysis: Spectral Theory and Special Tensors*, SIAM, Philadelphia.

Y. Qi (2018), *A very brief introduction to nonnegative tensors from the geometric viewpoint*, Mathematics, 6, p. 230.

S. Ragnarsson-Torbergsen (2012), *Structured Tensor Computations: Blocking, Symmetries and Kronecker Factorizations*, Ph.D. thesis, Cornell University.

M. Sathe, O. Schenk, and H. Burkhart (2012), *An auction-based weighted matching implementation on massively parallel architectures*, Parallel Comput., 38, pp. 595–614, https://doi.org/10.1016/j.parco.2012.09.001.

J.-Y. Shao (2013), *A general product of tensors with applications*, Linear Algebra Appl., 439, pp. 2350–2366.

T. Shen, Z. Zhang, Z. Chen, D. Gu, S. Liang, Y. Xu, R. Li, Y. Wei, Z. Liu, Y. Yi, et al. (2018), *A genome-scale metabolic network alignment method within a hypergraph-based framework using a rotational tensor-vector product*, Sci. Rep., 8, pp. 1–16.

R. Singh, J. Xu, and B. Berger (2008), *Global alignment of multiple protein interaction networks with application to functional orthology detection*, Proc. Natl. Acad. Sci. USA, 105, pp. 12763–12768.

C. Stark, B.-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers (2006), *Biogrid: A general repository for interaction datasets*, Nucleic Acids Res, 34, (suppl 1), pp. D535–D539.

L. Sun, B. Zheng, C. Bu, and Y. Wei (2016), *Moore-penrose inverse of tensors via einstein product*, Linear Multilinear Algebra, 64, pp. 686–698.

W. Sun, Y. Chen, and H. C. So (2018), *Tensor completion using kronecker rank-1 tensor train with application to visual data inpainting*, IEEE Access, 6, pp. 47804–47814.

V. Vijayan and T. Milenković (2017), *Multiple network alignment via multimagna++*, IEEE/ACM Trans. Comput. Biol. Bioinform., 15, pp. 1669–1682.

S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt (2010), *Graph kernels*, J. Mach. Learn. Res., 11, pp. 1201–1242.

J. Yan, C. Zhang, H. Zha, W. Liu, X. Yang, and S. M. Chu (2015), *Discrete hyper-graph matching*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1520–1528.

R. Zass and A. Shashua (2008), *Probabilistic graph and hypergraph matching*, in Proceedings of CVPR, pp. 1–8.

X. Zhang, C. Ling, and L. Qi (2012), *The best rank-1 approximation of a symmetric tensor and related spherical optimization problems*, SIAM J. Matrix Anal. Appl., 33, pp. 806–821.