FAST EFFICIENT AND SCALABLE CORE CONSISTENCY DIAGNOSTIC FOR THE PARAFAC DECOMPOSITION FOR BIG SPARSE TENSORS

Evangelos E. Papalexakis, Christos Faloutsos

School of Computer Science
Carnegie Mellon University

ABSTRACT

Multilinear analysis is pervasive in a wide variety of fields, ranging from Signal Processing to Chemometrics, and from Machine Vision to Data Mining. Determining the quality of a given tensor decomposition is a task of utmost importance that spans all fields of application of tensors. This task by itself is hard in its nature, since even determining the rank of a tensor is an NP-hard problem. Fortunately, there exist heuristics in the literature that can be effectively used for this task; one of these heuristics is the so-called Core Consistency Diagnostic (CORCONDIA) which is very intuitive and simple. However simple, computation of this diagnostic proves to be a very daunting task even for data of medium scale, let alone big tensor data. With the increase of the size of the tensor data that need to be analyzed, there grows the need for efficient and scalable algorithms to compute diagnostics such as CORCONDIA, in order to assess the modelling quality. In this work we derive a fast and exact algorithm for CORCONDIA which exploits data sparsity and scales very well as the tensor size increases.

Index Terms— Tensors, Tensor Decompositions, Scalability, Big Data

1. INTRODUCTION

Multilinear analysis and tensor decompositions have been increasingly popular in a very wide variety of fields, ranging from signal processing to data mining.

The majority of existing applications of tensor decompositions in fields such as Chemometrics, focus on dense data, where most of the values of a tensor are observed and non-zero [1]. However, recently, there has emerged an increasing interest in applying tensor decompositions on sparse data, where most of the coefficients of the tensor are unobserved; examples of such data can be links between web-pages [2, 3], computer networks [3, 4, 5], Knowledge Base data [6, 5], citation networks [3] and social networks [7, 3, 5].

As a running motivating example we choose that of social networks. Consider an online social network platform such as Facebook, which records relations and interactions between its users. Throughout the vast amount of Facebook users (estimated to be around 1.3 Billion), it is physically impossible to observe interactions between all users; i.e. a certain user interacts with a very small fraction of the total number of users. Thus, suppose that we observe a tensor of user interactions over time (i.e. the modes are (user, user, time)), the number of observed, non-zero values of this tensor will be very small, resulting in a highly sparse tensor with very high dimensions.

There are many challenges posed by the aforementioned type of tensors. In particular, traditional *dense* methods that assume that all data can (and should) be stored in main memory fall short. Computation of tensor decompositions in such scenarios was pioneered by Kolda and Bader in [2] and [8], where they introduce efficient in-memory computations for sparse tensors, taking advantage of the sparse structure and avoiding storing the entire data into memory. Subsequently, in [6], the authors apply this principle in a distributed *cloud* setting, where a tensor can span terabytes of storage.

Coming back to the example of the large sparse tensor that represents the time-evolving social network, the rank of its PARAFAC decomposition [9], i.e. the number of rank-one factors that we extract from the data, reflects the near cliques or communities in that network [10]. Thus, it is important to be able to determine the decomposition rank efficiently. Unfortunately, even determining the true rank of a tensor, contrary to the matrix case (where the solution is given to us by the Singular Value Decomposition, in polynomial time), is an NP-hard problem [11]. Fortunately, however, there exist heuristic methods which, given an F rank decomposition, are able to provide a *diagnostic* that captures how well this decomposition models the tensor data. The first such heuristic, CORCONDIA, appeared in [12] and subsequently described in detail in [13]; there exist more recent approaches which further extend the main idea of CORCONDIA [14].

The state of the art algorithms for CORCONDIA, so far have focused on dense and relatively small datasets; however, when we shift our attention to data of much larger scale (such as the running example of the time-evolving social network), state of the art approaches suffer, understandably so, from scalability issues. In this work, we make such diagnostics available for the analysis of very large tensors. In particular, our contributions are

- We derive an equivalent formula for computing CORCONDIA, and propose an efficient algorithm, able to scale on very high dimensional sparse tensors.
- We apply our algorithm to a big real-world time-evolving social network dataset, demonstrating its practicality in the analysis of big tensor/graph data.
- We make our code publicly available at `http://www.cs.cmu.edu/~epapalex/src/efficient_corcondia.zip`

2. BACKGROUND & PROBLEM FORMULATION

2.1. A Note on Notation

A tensor is denoted by $\underline{\mathbf{X}}$. A matrix is denoted by $\mathbf{X}$. A vector is denoted by $\mathbf{x}$. The symbol $\circ$ denotes the outer product. The symbol $\otimes$ denotes the Kronecker product. The symbol $\dagger$ denotes the Moore-Penrose pseudoinverse. The symbol $vec\,()$ denotes the vectorization operation.

2.2. Brief Introduction to Tensor Decompositions

Given a tensor $\underline{\mathbf{X}}$, we can decompose it according to the PARAFAC decomposition [9] as a sum of rank-one tensors:

$$\underline{\mathbf{X}} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

where the (i,j,k) entry of $\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$ is $\mathbf{a}_r(i)\mathbf{b}_r(j)\mathbf{c}_r(k)$. Usually, PARAFAC is represented in its matrix form $[\mathbf{A}, \mathbf{B}, \mathbf{C}]$, where the columns of matrix $\mathbf{A}$ are the $\mathbf{a}_r$ vectors (and accordingly for $\mathbf{B}, \mathbf{C}$). The PARAFAC decomposition is especially useful when we are interested in extracting the true latent factors that generate the tensor.

Another very popular Tensor decomposition is the Tucker 3 model [15], where a tensor is decomposed into rank-one factors times a core tensor:

$$\underline{\mathbf{X}} \approx \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} \underline{\mathbf{G}}(p,q,r)\mathbf{u}_p \circ \mathbf{v}_q \circ \mathbf{w}_r$$

where $\mathbf{U}, \mathbf{V}, \mathbf{W}$ are orthogonal. The Tucker 3 model is especially used for compression. Furthermore, PARAFAC can be seen as a restricted Tucker 3 model, where the core tensor $\underline{\mathbf{G}}$ is super-diagonal, i.e. non-zero values are only in the entries where $i = j = k$. This observation will be useful in order to motivate the CORCONDIA diagnostic.

2.3. Brief Introduction to CORCONDIA

As outlined in the Introduction, there exist a few diagnostics/heuristics for assessing the modelling quality of the PARAFAC decomposition. In this work, we will focus on CORCONDIA [12, 13], which is the simplest and most intuitive to describe. However, [14] which builds upon CORCONDIA can also benefit from our proposed algorithm.

In a nutshell, the idea behind CORCONDIA is the following: Given a tensor $\underline{\mathbf{X}}$ and its PARAFAC decomposition $\mathbf{A}, \mathbf{B}, \mathbf{C}$, one could imagine fitting a Tucker model where matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are the loading matrices of the Tucker 3 model and $\underline{\mathbf{G}}$ is the core tensor (which we need to solve for). Since, as we already mentioned, PARAFAC can be seen as a restricted Tucker 3 model with super-diagonal core tensor, if our PARAFAC modelling of $\underline{\mathbf{X}}$ using $\mathbf{A}, \mathbf{B}, \mathbf{C}$ is good, core tensor $\underline{\mathbf{G}}$ should be as close to super-diagonal as possible. If there are deviations from the super-diagonal, then this is a good indication that our PARAFAC model is somehow flawed (either the decomposition rank is not appropriate, or the data do not have the appropriate structure).

As it is highlighted in [12], since matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are not orthogonal, we may not use typical algorithms that are used to fit the Tucker model (e.g page 72 of [12]). Instead, we can pose the problem as the following least squares problem:

$$\min_{\underline{\mathbf{G}}} \|vec\,(\underline{\mathbf{X}}) - (\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C})\, vec\,(\underline{\mathbf{G}})\,\|_F^2$$

with solution: $vec\,(\underline{\mathbf{G}}) = (\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C})^{\dagger}\, vec\,(\underline{\mathbf{X}})$

3. PROBLEM DEFINITION & PROPOSED METHOD

Albeit simple and elegant, the solution of the Least Squares problem that lies in the heart of CORCONDIA suffers in the case of high dimensional data. In particular, this straightforward solution requires to first compute and store $(\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C})$ and then pseudoinvert it. Consider a $10^4 \times 10^4 \times 10^4$ tensor; even for a very low rank decomposition of $R = 10$, the aforementioned Kronecker product will be of size $10^{12} \times 10^3$, a fact which renders computing and storing such a matrix highly impractical (if not outright impossible), and subsequently, computing its pseudoinverse largely intractable. Even if the factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are sparse [16] (resulting in a sparse Kronecker product), pseudoinverting a matrix of such large dimensions is very computationally challenging.

In this section, we describe our proposed algorithm. Our "wish-list" of properties for our algorithm is the following:

- Avoid materializing any Kronecker product.
- Avoid directly pseudo-inverting the (potentially huge) aforementioned Kronecker product.
- Exploit any sparse structure in the factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and/or the tensor $\underline{\mathbf{X}}$.

In order to achieve the above, we need to reformulate the computation of CORCONDIA.

Claim 1. *The pseudoinverse* $(\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C})^{\dagger}$ *can be rewritten as*

$$(\mathbf{V_a} \otimes \mathbf{V_b} \otimes \mathbf{V_c})\left(\mathbf{\Sigma_a}^{-1} \otimes \mathbf{\Sigma_b}^{-1} \otimes \mathbf{\Sigma_c}^{-1}\right)\left(\mathbf{U_a}^T \otimes \mathbf{U_b}^T \otimes \mathbf{U_c}^T\right)$$

where $\mathbf{A} = \mathbf{U_a}\mathbf{\Sigma_a}\mathbf{V_a}^{\mathbf{T}}$, $\mathbf{B} = \mathbf{U_b}\mathbf{\Sigma_b}\mathbf{V_b}^{\mathbf{T}}$, *and* $\mathbf{C} = \mathbf{U_c}\mathbf{\Sigma_c}\mathbf{V_c}^{\mathbf{T}}$ *(i.e. the respective Singular Value Decompositions).*

Proof. For compactness, we show the two matrix case, but the extension to three matrices is straightforward. Using basic properties of the Kronecker product from [17], and rewriting $\mathbf{A}, \mathbf{B}$ using their SVD, we may write

$$\begin{aligned}(\mathbf{A} \otimes \mathbf{B}) &= \left[\left(\mathbf{U_a}\mathbf{\Sigma_a}\mathbf{V_a}^T\right) \otimes \left(\mathbf{U_b}\mathbf{\Sigma_b}\mathbf{V_b}^T\right)\right] \\ &= \left[\left(\mathbf{U_a}\mathbf{\Sigma_a}\right) \otimes \left(\mathbf{U_b}\mathbf{\Sigma_b}\right)\left(\mathbf{V_a} \otimes \mathbf{V_b}\right)^T\right] \\ &= \left[\left(\mathbf{U_a} \otimes \mathbf{U_b}\right)\left(\mathbf{\Sigma_a} \otimes \mathbf{\Sigma_b}\right)\left(\mathbf{V_a} \otimes \mathbf{V_b}\right)^T\right]\end{aligned}$$

By invoking properties shown in [18], we can show that show that $(\mathbf{U_a} \otimes \mathbf{U_b})$ is orthonormal and $(\mathbf{\Sigma_a} \otimes \mathbf{\Sigma_b})$ is diagonal with non-negative entries.

Then,because the SVD is unique, then

$$\mathbf{A} \otimes \mathbf{B} = \left[\left(\mathbf{U_a} \otimes \mathbf{U_b}\right)\left(\mathbf{\Sigma_a} \otimes \mathbf{\Sigma_b}\right)\left(\mathbf{V_a} \otimes \mathbf{V_b}\right)^T\right]$$

is the SVD of $\mathbf{A} \otimes \mathbf{B}$.

Since the above is the SVD, then the Moore-Penrose pseudoinverse will be $\left(\mathbf{V_a} \otimes \mathbf{V_b}\right)\left(\mathbf{\Sigma_a}^{-1} \otimes \mathbf{\Sigma_b}^{-1}\right)\left(\mathbf{U_a}^T \otimes \mathbf{U_b}^T\right)$ □

In [19] the authors first show an equivalent formulation of the Kronecker product pseudoinverse, which is however not expressed via the SVD, and thus suffers from numerical instabilities. Subsequently, they introduce a more complicated reformulation of the pseudoinverse of Kronecker products, where SVD is applied after computing the QR decomposition of the matrices involved. In this work we prefer the reformulation of Claim 1 since it is usually more efficient to compute in our case. In order to substantiate this claim we conducted the following experiment: Using Matlab (which to the best of our knowledge is the state of the art when it comes to efficient dense and sparse matrix computations in main memory), we computed the QR decomposition and the *economy* version of the SVD for a series of random matrices with increasing number of rows and fixed number of columns (a scenario which resembles the case where we have a tensor of increasing dimensions but we have a fixed rank). Figure 1 indicates the efficiency of SVD.

It is important to note here that in the case that the factors $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are sparse, we can exploit that fact using SVD for sparse matrices, to further speed up the computation. The straightforward algorithm that computes $(\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C})$ cannot take advantage of factor sparsity. Now, we can solve CORCONDIA without materializing any of the Kronecker products. Instead we observe that the equation we need to solve is

$$\begin{aligned}vec\left(\underline{\mathbf{G}}\right) = &\left(\mathbf{V_a} \otimes \mathbf{V_b} \otimes \mathbf{V_c}\right)\left(\mathbf{\Sigma_a}^{-1} \otimes \mathbf{\Sigma_b}^{-1} \otimes \mathbf{\Sigma_c}^{-1}\right) \\ &\left(\mathbf{U_a}^T \otimes \mathbf{U_b}^T \otimes \mathbf{U_c}^T\right) vec\left(\underline{\mathbf{X}}\right)\end{aligned}$$

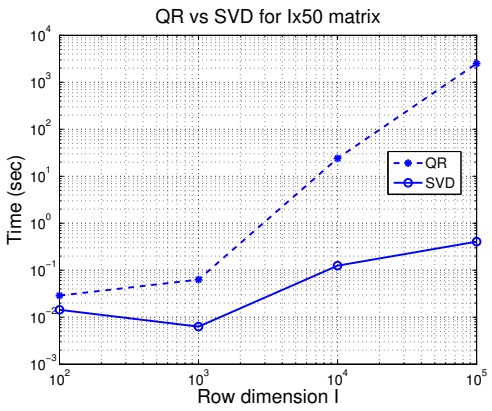


Fig. 1. QR vs. SVD

which is simply a series of Kronecker products times a vector computations.

In [20], the author provides an efficient algorithm that does not materialize the (potentially very large) Kronecker product, and is able to efficiently compute products of the the form. $(\mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \cdots \otimes \mathbf{A}_k)\,\mathbf{x}$. We use the algorithm of [20] in our proposed Algorithm. When the tensor is sparse, the above computations can be carried out very efficiently, again, using sparse matrix multiplication. In Algorithm 1, we show a listing of our efficient CORCONDIA algorithm.

Algorithm 1: Efficient CORCONDIA

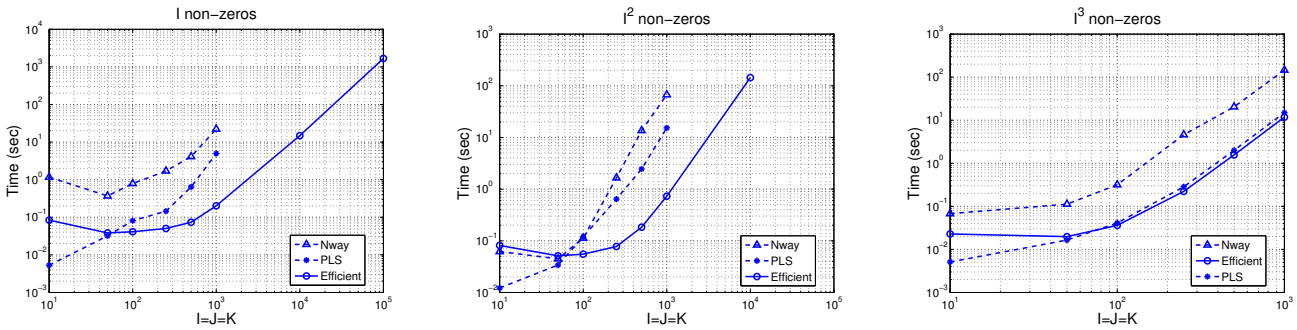
Input: Tensor $\underline{\mathbf{X}}$ and PARAFAC factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$.
Output: CORCONDIA diagnostic c.

1: Compute $\mathbf{A} = \mathbf{U_a}\mathbf{\Sigma_a}\mathbf{V_a}^T$
2: Compute $\mathbf{B} = \mathbf{U_b}\mathbf{\Sigma_b}\mathbf{V_b}^T$
3: Compute $\mathbf{C} = \mathbf{U_c}\mathbf{\Sigma_c}\mathbf{V_c}^T$
4: Calculate $\mathbf{y} = \left(\mathbf{U_a}^T \otimes \mathbf{U_b}^T \otimes \mathbf{U_c}^T\right) vec\left(\underline{\mathbf{X}}\right)$ using Algorithm of [20].
5: Calculate $\mathbf{z} = \left(\mathbf{\Sigma_a}^{-1} \otimes \mathbf{\Sigma_b}^{-1} \otimes \mathbf{\Sigma_c}^{-1}\right)\mathbf{y}$ using Algorithm of [20].
6: Calculate $vec\left(\underline{\mathbf{G}}\right) = \left(\mathbf{V_a} \otimes \mathbf{V_b} \otimes \mathbf{V_c}\right)\mathbf{z}$ using Algorithm of [20].
7: $c =$ FORMULA FOR CORCONDIA

4. EXPERIMENTS

We implemented Algorithm 1 in Matlab, using the Tensor Toolbox [21] (available at: `http://www.sandia.gov/~tgkolda/TensorToolbox/index-2.5.html`), which provides efficient manipulation and storage of sparse tensors. We make our code publicly available[1]. For comparisons, we used two baselines: the implementation of the

[1]Download our code at `http://www.cs.cmu.edu/~epapalex/src/efficient_corcondia.zip`



(a) Very Sparse (b) Sparse (c) Dense

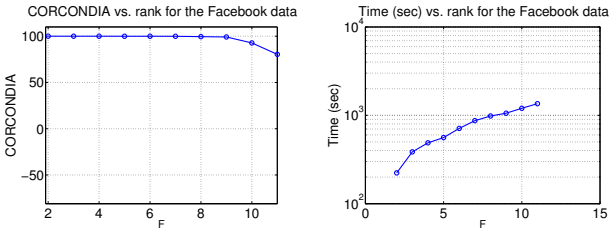
Fig. 2. Time vs $I = J = K$

N-way Toolbox for Matlab[22] (available at: `http://www.models.life.ku.dk/nwaytoolbox`), and the implementation of the PLS Toolbox (available at: `http://www.eigenvector.com/software/pls_toolbox.htm`). The PLS Toolbox is commercial, and is considered the state of the art for computing CORCONDIA, however, we include comparisons with the N-way Toolbox since it is freely available. All experiments were run on a workstation with 4 Intel(R) Xeon(R) E7- 8837 and 1TB of RAM.

Figure 2 shows the execution time as a function of the tensor mode dimension I for $I \times I \times I$ tensors for three cases: (a) very sparse tensors (with I non-zero values), (b) moderately sparse (with I^2 non-zero values), and (c) fully dense (with I^3 non-zero values). In Fig. 2(a), we see that our proposed algorithm is generally much faster than both baselines (note that the figures are in log-scales), while it keeps working for $I = 10^4$ where the baselines run out of memory. However, our proposed algorithm is able to scale up to $10^5 \times 10^5 \times 10^5$ tensors. In Fig. 2(b) we observe similar behavior, where the overall performance (for all three algorithms) is slightly lower, due to the existence of more non-zero values. Finally, in Fig. 2(c), as expected, we see that our proposed algorithm has the same performance as the state of the art PLS Toolbox implementation, since there is no sparsity to take advantage of.

5. CASE STUDY

In this section, we use our proposed algorithm in order to determine the appropriate number of components for the PARAFAC decomposition of a big real-world dataset. In particular, we chose a time-evolving social network dataset, more specifically a snapshot of Facebook (available at: `http://socialnetworks.mpi-sws.org/data-wosn2009.html`). This dataset records which user posted on another user's Wall and what date, forming a three-mode (User, User, Date) tensor, with dimensions: $63891 \times 63890 \times 1847$ and 737778 non-zero values. Despite the very large dimensions of the tensor, our algorithm is able to compute the CORCONDIA diagnostic for different values of the rank, as shown in Fig. 3(a). For each rank, we compute CORCONDIA 100 times and we show the maximum value attained, which reflects the best possible decomposition among those 100 iterations. In Fig. 3(b) we show the average time it took to compute the diagnostic for each rank.



(a) CORCONDIA vs. Rank (b) Time (sec) vs. Rank

Fig. 3. Analyzing the Facebook tensor

6. CONCLUSIONS

In this work we propose an efficient algorithm for computing the CORCONDIA diagnostic, especially for large sparse tensors. The important take-home point is that in cases where either the tensor or the factors or both are sparse, our proposed algorithm significantly outperforms the state of the art baselines and scales very well as the data size grows. In the fully dense scenario, our proposed algorithm is as good as the state of the art.

7. ACKNOWLEDGEMENTS

Research was supported by the National Science Foundation Grant No. IIS-1247489. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding parties. The authors would like to thank Prof. Rasmus Bro for valuable conversations.

8. REFERENCES

[1] R. Bro, "Parafac. tutorial and applications," *Chemometrics and intelligent laboratory systems*, vol. 38, no. 2, pp. 149–171, 1997.

[2] T.G. Kolda and B.W. Bader, "The tophits model for higher-order web link analysis," in *Workshop on Link Analysis, Counterterrorism and Security*, 2006, vol. 7, pp. 26–29.

[3] Tamara G Kolda and Jimeng Sun, "Scalable tensor decompositions for multi-aspect data mining," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 2008, pp. 363–372.

[4] K. Maruhashi, F. Guo, and C. Faloutsos, "Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis," in *Proceedings of the Third International Conference on Advances in Social Network Analysis and Mining*, 2011.

[5] Evangelos E Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos, "Parcube: Sparse parallelizable tensor decompositions," in *Machine Learning and Knowledge Discovery in Databases*, pp. 521–536. Springer, 2012.

[6] U Kang, Evangelos Papalexakis, Abhay Harpale, and Christos Faloutsos, "Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 316–324.

[7] B.W. Bader, R.A. Harshman, and T.G. Kolda, "Temporal analysis of social networks using three-way dedicom," *Sandia National Laboratories TR SAND2006-2161*, 2006.

[8] Brett W. Bader and Tamara G. Kolda, "Efficient MATLAB computations with sparse and factored tensors," *SIAM Journal on Scientific Computing*, vol. 30, no. 1, pp. 205–231, December 2007.

[9] R.A. Harshman, "Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis," 1970.

[10] Evangelos E Papalexakis, Leman Akoglu, and Dino Ience, "Do more views of a graph help? community detection and clustering in multi-graphs," in *Information Fusion (FUSION), 2013 16th International Conference on*. IEEE, 2013, pp. 899–905.

[11] Christopher J Hillar and Lek-Heng Lim, "Most tensor problems are np-hard," *Journal of the ACM (JACM)*, vol. 60, no. 6, pp. 45, 2013.

[12] Rasmus Bro, *Multi-way analysis in the food industry: models, algorithms, and applications*, Ph.D. thesis, 1998.

[13] Rasmus Bro and Henk AL Kiers, "A new efficient method for determining the number of components in parafac models," *Journal of chemometrics*, vol. 17, no. 5, pp. 274–286, 2003.

[14] Joao Paulo CL da Costa, Martin Haardt, and F Romer, "Robust methods based on the hosvd for estimating the model order in parafac models," in *Sensor Array and Multichannel Signal Processing Workshop, 2008. SAM 2008. 5th IEEE*. IEEE, 2008, pp. 510–514.

[15] Pieter M Kroonenberg and Jan De Leeuw, "Principal component analysis of three-mode data by means of alternating least squares algorithms," *Psychometrika*, vol. 45, no. 1, pp. 69–97, 1980.

[16] Evangelos E Papalexakis, Nicholas D Sidiropoulos, and Rasmus Bro, "From k-means to higher-way coclustering: multilinear decomposition with sparse latent factors," *Signal Processing, IEEE Transactions on*, vol. 61, no. 2, pp. 493–506, 2013.

[17] H Neudecker, "A note on kronecker matrix products and matrix equation systems," *SIAM Journal on Applied Mathematics*, vol. 17, no. 3, pp. 603–606, 1969.

[18] Charles F Van Loan, "The ubiquitous kronecker product," *Journal of computational and applied mathematics*, vol. 123, no. 1, pp. 85–100, 2000.

[19] Donald W Fausett and Charles T Fulton, "Large least squares problems involving kronecker products," *SIAM Journal on Matrix Analysis and Applications*, vol. 15, no. 1, pp. 219–227, 1994.

[20] Paul E Buis and Wayne R Dyksen, "Efficient vector and parallel manipulation of tensor products," *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 1, pp. 18–23, 1996.

[21] B.W. Bader and T.G. Kolda, "Matlab tensor toolbox version 2.2," *Albuquerque, NM, USA: Sandia National Laboratories*, 2007.

[22] C.A. Andersson and R. Bro, "The n-way toolbox for matlab," *Chemometrics and Intelligent Laboratory Systems*, vol. 52, no. 1, pp. 1–4, 2000.