EXACT PARTITIONING OF HIGH-ORDER PLANTED MODELS WITH A TENSOR NUCLEAR NORM CONSTRAINT

Chuyang Ke *Jean Honorio*

Purdue University

ABSTRACT

We study the problem of exact partitioning of the hypergraphs generated by high-order planted models. A high-order planted model assumes some underlying cluster structures, and simulates high-order interactions by placing hyperedges among nodes. Example models include the disjoint hypercliques, the densest subhypergraphs, and the hypergraph stochastic block models. We show that exact partitioning of high-order planted models is achievable through solving a convex optimization problem with a tensor nuclear norm constraint. Our analysis provides the statistical upper bounds for our approach to succeed on recovering the true underlying cluster structures, with high probability.

***Index Terms*—** High-order Planted Models, Hypergraphs, Exact Partitioning

1. INTRODUCTION

The problem of partitioning high-order models have been studied for some long time. Graph-theoretic problems including cuts, colorings, and traversals have been analyzed in earlier works [1, 2]. In the past decade, researchers have started looking at spectral theory and algebraic connectivity of hypergraphs [3, 4, 5, 6]. Certain high-order models of interests have received more attention from an algorithmic point of view. This includes the densest subhypergraphs [7], the hypergraph stochastic block models (SBMs) [8, 9], and the hypergraph planted cliques [10, 11]. Despite years of research, however, very little is known about the *exact partitioning* conditions in general high-order planted models.

In this paper we propose a convex optimization approach for exact partitioning of high-order planted models. A high-order planted model generates hypergraphs, which simulate multi-entity interactions in a network. Our model class formulation is highly general and subsumes models analyzed and applied in prior literature, including the disjoint hypercliques, the densest subhypergraphs, and the hypergraph stochastic block models. When the order is set to 2, our definition of high-order planted models reduces to regular planted models with ordinary graphs. Our analysis establishes the regime in which exact recovery of hidden cluster structures is possible from noisy observation of hypergraphs.

Related works. In addition to the works mentioned above, there has been a lot of research on the partitioning of certain high-order planted models. For the densest subhypergraphs, [12] and [13] proposed theoretical objective function approximation algorithms. Our goal, arguably more challenging, is to recover the true underlying clustering structure. For hypergraph SBMs, approaches include truncating the hypergraph to a multigraph [8] or an ordinary graph with a weighted adjacency matrix [14]. It is worth highlighting that our result is not merely an extension. Our definition of high-order planted models is highly general, and the convex optimization formulation with a tensor nuclear norm constraint is novel. Moreover our approach does not approximate, or truncates the hypergraph to an ordinary graph. To the best of authors' knowledge, we are the first one applying tensor nuclear norm methods in high-order partitioning problems.

Summary of our contributions. We provide a series of novel results in this paper:

- We propose the highly general class definition of high-order planted models, and demonstrate that our model class definition subsumes several existing planted models, including the disjoint hypercliques, the densest subhypergraphs, and the hypergraph stochastic block models.
- We formulate the problem of exact partitioning in high-order planted models as a novel tensor optimization problem with a tensor nuclear norm constraint, and we establish the regime in which hidden cluster structures can be recovered correctly.

2. PRELIMINARIES

2.1. High-order Tensors

In this section, we introduce the notations that will be used in the paper. We use lowercase font (e.g., a, b, u, v) for scalars and vectors, uppercase font (e.g., A, B, C) for matrices, and calligraphic font (e.g., $\mathcal{A}, \mathcal{B}, \mathcal{C}$) for tensors. We use $\mathbb{R}$ to denote the set of real numbers.

For any integer n, we use $[n]$ to denote the set $\{1, \ldots, n\}$. For clarity when dealing with a sequence of objects, we use the superscript (i) to denote the i-th object in the sequence, and subscript j to denote the j-th entry. For example, for a

sequence of vectors $\{x^{(i)}\}_{i\in[n]}$, $x_2^{(1)}$ represents the second entry of vector $x^{(1)}$. The notation $\otimes$ is used to denote outer product of vectors, for example, $x^{(1)}\otimes\ldots\otimes x^{(m)}$ is a tensor of order m, such that $(x^{(1)}\otimes\ldots\otimes x^{(m)})_{i_1,\ldots,i_m} := x_{i_1}^{(1)}\ldots x_{i_m}^{(m)}$. We use $\mathbf{1}$ to denote the all-one vector.

Let $\mathcal{A} = (\mathcal{A}_{i_1,\ldots,i_m})$ be an m-order tensor of size $n_1\times\cdots\times n_m$. Tensor $\mathcal{A}$ is *symmetric* if it is invariant under any permutation of its indices, i.e., $\mathcal{A}_{\sigma(i_1),\ldots,\sigma(i_m)} = \mathcal{A}_{i_1,\ldots,i_m}$, for any permutation $\sigma : [m]\to[m]$. We use $\mathcal{I}$ to do denote the *identity tensor*, such that $\mathcal{I}_{i_1,\ldots,i_1} = \cdots = \mathcal{I}_{i_m,\ldots,i_m}$, and all other entries are 0.

For tensor $\mathcal{A} = (\mathcal{A}_{i_1,\ldots,i_m})$ and $\mathcal{B} = (\mathcal{B}_{i_1,\ldots,i_m})$ of same size, we define the *inner product* of $\mathcal{A}$ and $\mathcal{B}$ as $\langle\mathcal{A},\mathcal{B}\rangle := \sum_{i_1,\ldots,i_m}\mathcal{A}_{i_1,\ldots,i_m}\mathcal{B}_{i_1,\ldots,i_m}$. Tensor addition and subtraction are defined entrywise, e.g., $\mathcal{A}+\mathcal{B} = (\mathcal{A}_{i_1,\ldots,i_m}+\mathcal{B}_{i_1,\ldots,i_m})$. With slight abuse of notation, for any constant $c\in\mathbb{R}$, we use $\mathcal{A} < c$ to denote entrywise inequality.

For any vector $u\in\mathbb{R}^n$, we denote the corresponding m-th order *rank-one* tensor as $u^{\otimes m}$, where $(u^{\otimes m})_{i_1,\ldots,i_m} := u_{i_1}\ldots u_{i_m}$. For any symmetric tensor $\mathcal{A}$, we define its *spectral norm* as $|||\mathcal{A}||| := \sup_{u\in\mathbb{S}^{n-1}}|\langle\mathcal{A},u^{\otimes m}\rangle|$, where $\mathbb{S}^{n-1}$ denotes the unit sphere. Similarly we define its *nuclear norm* as $|||\mathcal{A}|||_* := \inf\left\{\sum_{i=1}^r|\lambda_i| : A = \sum_{i=1}^r\lambda_i u^{(i)\otimes m}, u^{(i)}\in\mathbb{S}^{n-1}\right\}$. It is worth mentioning that, like the Schatten p-norms in the matrix case, the tensor spectral norm and tensor nuclear norm are also dual to each other [15].

2.2. High-order Planted Models

We now introduce the definition of *high-order planted models*.

Definition 1 (High-order Planted Models)**.** *A high-order planted model is denoted as $\mathcal{M}(n,m,r,k,p,q)$, where n is the number of vertices, and m is the order of the model. It is assumed that uniformly at random, rk out of n vertices are grouped into r clusters of equal size k, and the remaining $n-rk$ vertices do not belong any cluster. p,q are signal parameters bounded by* 0 *and* 1.

Model $\mathcal{M}(n,m,r,k,p,q)$ generates a random hypergraph $\mathcal{G} = (\mathcal{V},\mathcal{E})$ in the following way. For each subset $\{v_{i_1},\ldots,v_{i_m}\}\subset\mathcal{V}$ of m vertices, if all are from the same cluster, nature adds the hyperedge $(v_{i_1},\ldots,v_{i_m})$ to $\mathcal{E}$ with probability p; otherwise, nature adds the hyperedge with probability q.

Our goal is to recover the cluster membership of vertices in model $\mathcal{M}$ from the observed hypergraph $\mathcal{G}$. For any $i\in[r]$, we use $y^{(i)*}\in\{0,1\}^n$ to denote the true membership vector of cluster i, such that $y_j^{(i)*} = 1$ if vertex j is in cluster i, and 0 otherwise. We introduce the agreement tensor $\mathcal{Y}^* = \sum_{i=1}^r y^{(i)*\otimes m}$. It is not hard to see that $\mathcal{Y}^*$ is 0-1 valued, as the clusters are non-overlapping. Thus, $\mathcal{Y}^*$ encodes all cluster membership information (up to the permutation of clusters, as there is no way to distinguish between clusters without prior knowledge). In particular, in the two cluster case with $r=2$ and $rk=n$, $y^{(i)*}$'s can be encoded by a single vector $y\in\{-1,+1\}^n$. Let $\mathcal{A}$ be the adjacency tensor of hypergraph $\mathcal{G}$. From the definition above, $\mathcal{A}$ is a symmetric tensor. Each entry in $\mathcal{A}$ is generated to be 1 with probability p if the corresponding entry in $\mathcal{Y}^*$ is 1; otherwise it is generated to be 1 with probability q. The problem now reduces to recover $\mathcal{Y}^*$ from the observation of $\mathcal{A}$.

Classical models. Here are some classical models covered by our definition.

- **Densest Subhypergraph**: $0<q<p<1, r=1$. In this case there exists a dense subhypergraph of size k in the observed hypergraph $\mathcal{G}$ (see e.g., [7]).
- **Hypergraph Stochastic Block Model**: $0<q<p<1, n=rk, r\geq 2$. In this case there exists r dense subhypergraphs of size k in the observed hypergraph $\mathcal{G}$ (see e.g., [8, 16]).
- **Disjoint Hypercliques**: $p=1, 0<q<1$. In this case, r hypercliques of size k are planted in the observed hypergraph $\mathcal{G}$. A hyperclique is the generalization of graph cliques. In a hyperclique, every distinct m-tuple is connected by a hyperedge (see e.g., [10, 11]).

3. THEORETICAL GUARANTEES OF EXACT PARTITIONING

In this section, we propose and analyze an algorithm which recovers the true underlying cluster structures in high-order planted models. Recall that $\mathcal{Y}^*$ is the true agreement tensor. We say an algorithm achieves *exact partitioning*, if its output $\mathcal{Y}$ is identical to $\mathcal{Y}^*$.

Algorithm 1 Exact Partitioning of High-order Models

Input: adjacency tensor $\mathcal{A}$

Output: estimated agreement tensor $\mathcal{Y}$

$$\begin{aligned} \underset{\mathcal{Y}}{\text{maximize}} \quad & \langle\mathcal{A},\mathcal{Y}\rangle \\ \text{subject to} \quad & |||\mathcal{Y}|||_* \leq rk^{m/2} \\ & \left\langle\mathbf{1}^{\otimes m},\mathcal{Y}\right\rangle = rk^m \\ & 0\leq\mathcal{Y}\leq 1 \end{aligned} \tag{1}$$

In the following analysis we examine the statistical conditions for problem (1) to succeed with high probability. Note that the objective function and constraints in problem (1) are convex. To be clear, while our convex optimization problem might be NP-hard due to the tensor nuclear norm constraint [15], the utility of this constructive procedure is as a proof technique: it allows us to apply convex optimization tools to characterize the statistical upper limits of exact partitioning of high-order planted models. **Our analysis establishes the regime in which given the adjacency tensor $\mathcal{A}$, the true underlying cluster structures $\mathcal{Y}^*$ can be recovered by problem** (1) **perfectly.**

Remark on exact partitioning. It is worth mentioning that Algorithm 1 does not require any rounding step and outputs the exact solution. Hypergraph partitioning algorithms in prior literature either are objective function approximation algorithms [12, 13], or unfold hypergraphs into matrices [8, 14]. Note that the groundtruth $\mathcal{Y}^*$ is a feasible solution to problem (1). Our analysis states that if certain statistical conditions are satisfied, with high probability no other feasible solution $\mathcal{Y} \neq \mathcal{Y}^*$ will achieve a better objective value.

We now present the main theorem, which provides a sufficient condition for (1) to succeed with high probability.

Theorem 1. *Consider any hypergraph $\mathcal{G}$ sampled from a high-order model $\mathcal{M}(n, m, r, k, p, q)$. Let $\mathcal{A}$ be the adjacency tensor of $\mathcal{G}$. If*

$$\frac{(p-q)^2}{p(1-q)} = \Omega\left(\frac{nm^5 \log m}{k^{m-1}}\right), \tag{2}$$

then problem (1) *recovers the underlying cluster structure of $\mathcal{M}$ perfectly with probability at least $1 - O(1/n)$.*

The proof can be found in Appendix B in the full version.

Remark on rates. In high-order planted models, p and q are signal parameters that determine the signal-to-noise ratio (SNR) of the model. This is implied by the left-hand side of (2): as the gap $p - q$ becomes larger, SNR becomes higher and exact partitioning gets easier. On the right-hand side, one can notice that as n gets larger, the whole term becomes smaller (remember that $n = rk$). From an information-theoretical point of view this is intuitive, as a larger number of samples leads to easier recovery of the true signal.

It would also be interesting to compare our rates with those of ordinary planted models. In the case of $m = 2$, our condition becomes $(p-q)^2k^2 \geq Cp(1-q)kn$ for some constant C, while the condition in [17] is $(p-q)^2k^2 \geq C(p(1-q)k\log n + q(1-q)n)$. Comparison on the right-hand sides shows that our bound only requires a slightly higher order ($\Omega(kn)$ versus $\Omega(k\log n + n)$).

Remark on convexity. To be clear, while the tensor nuclear norm constraint in problem (1) might be NP-hard to compute [15], the constructive procedure has two utilities. First, as a proof technique, it enables us to apply convex optimization tools to characterize the statistical upper limits of exact partitioning of high-order planted models. Second, the convex formulation allows us to implement efficient gradient-based solvers. This is customary in many fields of machine learning, including deep learning and Bayesian networks: solving for the exact solution is NP-hard, but numerical methods are feasible and efficient. To see this, we provide simulation results on both synthetic and real-world datasets in the next section.

4. SIMULATION RESULTS

In this section, we test the proposed convex optimization formulation (1) using a projected gradient descent solver. Projected gradient descent is a standard method to solve constrained convex optimization problems. Our projected gradient descent solver in Algorithm 2 works as follows: starting from an initial point $\mathcal{Y}^{(0)}$, until the stopping condition is met, the algorithm repeats the assignment $\mathcal{Y}^{(k+1)} \leftarrow P(\mathcal{Y}^{(k)} + \eta\mathcal{A})$, where η is the step size of each iteration, and $P(\cdot)$ is a projection operator. The projection operator tries to find the "closest" point to $\mathcal{Y}^{(k)} + \eta\mathcal{A}$, that fulfills all the constraints in the convex problem (1). To fulfill $\|\|\mathcal{Y}\|\|_* \leq rk^{m/2}$, $\langle \mathbf{1}^{\otimes m}, \mathcal{Y}\rangle = rk^m$, and $0 \leq \mathcal{Y} \leq 1$, the algorithm scales all diagonal of $\mathcal{Y}^{(k)}$ so that their summation is less than $rk^{m/2}$, scales all entries in $\mathcal{Y}^{(k)}$ so that their summation is rk^m, maps all negative entries to 0, and maps all entries being greater than 1 to 1. In addition, Algorithm 2 invokes the helper function in Algorithm 3, using the method in Equation (3.1) in [18], which tries to find a unit vector $w \in \mathbb{R}^n$ such that $\left\langle \mathcal{Y}^{(k)}, w^{\otimes m}\right\rangle$ is minimized. Algorithm 2 then subtracts the tensor $\left\langle \mathcal{Y}^{(k)}, w^{\otimes m}\right\rangle \cdot w^{\otimes m}$ from $\mathcal{Y}^{(k)}$, making the projected tensor positive semidefinite in the direction of w. We assume the point $P(\mathcal{Y}^{(k)} + \eta\mathcal{A})$ satisfies the constraints, after repeating the projection procedure for a number of iterations.

Experiment 1: To validate Theorem 1, we generate synthetic fourth order planted models following the procedure in Definition 1. We consider the setting of the hypergraph stochastic block model with $r = 2$, $k = n/2$, and encode the membership vector y^* using $+1$ and -1's. We test cases with n being 30, 60, and 120, respectively. In the experiment the signal parameters p and q iterate over the range $(0, 1)$, with an interval of 0.05. We run 20 trials for each pair of p and q. During each trial, a hypergraph $\mathcal{G}$ is sampled, and we invoke Algorithm 2 to solve for the agreement tensor $\hat{\mathcal{Y}}$. To evaluate the performance of the proposed algorithm, we run Algorithm 4 (similar to Algorithm 3 but finds positive eigenvalues; see full version Appendix C) to compute $\hat{y}$, the top tensor eigenvector of $\hat{\mathcal{Y}}$, and then compare the corresponding sign vector $\text{sign}(\hat{y})$ against the groundtruth y^*.

We report the empirical probability of exact recovery $\mathbb{P}\{\hat{y} = y^*\}$ in Figure 1. We plot the empirical probability against $C := \log\left(\frac{(p-q)^2 n}{p(1-q)k^3}\right)$, which is equivalent to the log of the left-hand side of (2) divided by its right-hand side. Our result suggests that as long as C is greater than the constant threshold of 10, the proposed method performs well and recovers the underlying group structure with high probability. This matches our theoretic findings in Theorem 1.

Experiment 2: We implement the proposed algorithm on a real-world dataset, *email-Eu-core* [19]. During each trial, we extract n most connected nodes from the dataset, and convert the induced subgraph into fourth order hypergraphs by counting the existence of *star motifs* and *cycle motifs* (see Figure 4). We then follow the same procedure used in the previous experiment. We also compare our results with the motif clustering algorithm [20].

We report the number of correctly recovered labels in Fig-

ure 2. The result suggests that our method performs well compared to the motif clustering method on both hypergraphs. In particular, the figure suggests that our tensor method is more robust to noises ($n \geq 70$), since as n gets larger, newly extracted nodes are less connected, and the community structure signal becomes weaker. We also report the runtime of the proposed method in Figure 3. The runtime can be fitted almost perfectly by a third order polynomial, suggesting the polynomial efficiency of our method.

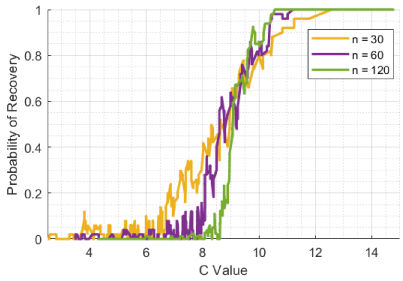

Fig. 1: Simulations using synthetic fourth order planted models with different values of signal parameters p and q. In this case $m = 4$. The x-axis is set by $C := \log\left(\frac{(p-q)^2 n}{p(1-q)k^3}\right)$. The y-axis is the probability of exact partitioning $\mathbb{P}\{\hat{y} = y^*\}$. This matches our theoretic findings in Theorem 1.

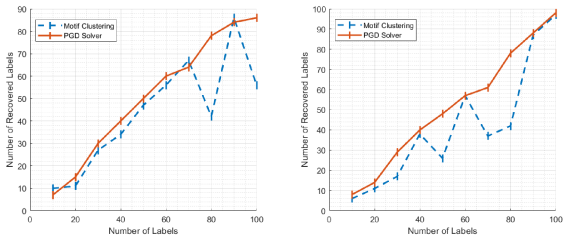

Fig. 2: Simulations using the real-world dataset *email-Eu-core* [19]. We generate fourth-order hypergraphs from the dataset using the star motif and the cycle motif, respectively (Figure 4). In this case $m = 4$. The dashed curve uses the motif clustering algorithm in [20]. The result suggests that the proposed method performs well on real-world datasets.

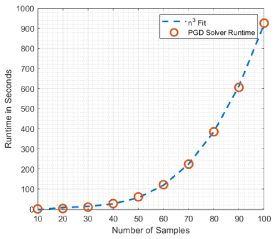

Fig. 3: The runtime of Algorithm 2 versus the number of samples. The runtime can be fitted almost perfectly by a third order polynomial, suggesting the polynomial efficiency of our method.

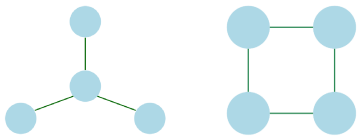

Fig. 4: Star motif and cycle motif. Motifs are small structured units in networks. To convert an ordinary (pairwise) graph to a fourth order hypergraph, we iterate over all 4-tuple of nodes (i, j, k, l) in the original graph. If the subgraph induced by (i, j, k, l) has the same structure as the desired motif, we add a hyperedge (i, j, k, l) to the new hypergraph.

Algorithm 2 Projected Gradient Descent Solver

Input: adjacency tensor $\mathcal{A}$, step size ζ
Output: Agreement tensor $\mathcal{Y}$

$\mathcal{Y} \leftarrow \frac{rk^m}{n^m}\mathbf{1}^{\otimes m}$
for each outer iteration **do**
  $\mathcal{Y} \leftarrow \mathcal{Y} + \zeta\mathcal{A}$ {gradient descent step}
  for each inner iteration **do**
    $w \leftarrow$ Algorithm 3 (Y)
    $w \leftarrow \frac{1}{\|w\|}w$
    $\mathcal{W} \leftarrow w^{\otimes m}$
    $c_w \leftarrow \langle \mathcal{W}, \mathcal{Y} \rangle$ {run projection step if $\mathcal{Y}$ is not PSD}
    if $c_w < 0$ **then**
      $\mathcal{Y} \leftarrow \mathcal{Y} - c_w\mathcal{Y}$ {make $\mathcal{Y}$ PSD in direction of w}
      $\mathcal{Y} \leftarrow \min(\max(\mathcal{Y}, 0), 1)$ {project to $[0, 1]$}
      $\mathcal{Y} \leftarrow \frac{rk^m}{\sum_{i_1,\ldots,i_m}\mathcal{Y}_{i_1,\ldots,i_m}} \cdot \mathcal{Y}$ {ensure $\langle \mathbf{1}^{\otimes m}, \mathcal{Y} \rangle = rk^m$}
      if $\sum_i \mathcal{Y}_{i,\ldots,i} > rk^{m/2}$ **then**
        $\mathcal{Y}_{i,\ldots,i} \leftarrow rk^{m/2} \cdot \frac{\mathcal{Y}_{i,\ldots,i}}{\sum_i \mathcal{Y}_{i,\ldots,i}}$ {ensure $\|\mathcal{Y}\|_* \leq rk^{m/2}$}

Algorithm 3 Negative Tensor Eigenvalue Searcher

Input: Target tensor $\mathcal{Y}$, step size γ
Output: Target vector w

Initialize w; $f_1 \leftarrow \infty$
for each iteration **do**
  $f_1' \leftarrow f_1$ {record previous obj. value}
  $f_1 \leftarrow \frac{1}{2m}\langle \mathcal{I}, w^{\otimes m} \rangle^2 + \frac{1}{m}\langle \mathcal{Y}, w^{\otimes m} \rangle$ {check obj. value}
  if $f_1 < 0$ or $\langle \mathcal{Y}, w^{\otimes m} \rangle < 0$ **then**
    break {can certify negative definiteness}
  if $f_1 > f_1'$ **then**
    break {obj. is starting to grow; stop}
  $\nabla f_1 \leftarrow \langle \mathcal{I}, w^{\otimes m} \rangle \mathcal{I}w^{\otimes m-1} + \mathcal{Y}w^{\otimes m-1}$
  $w \leftarrow w - \gamma\nabla f_1$ {gradient descent step}
if $f_1 \geq 0$ and $\langle \mathcal{A}, w^{\otimes m} \rangle \geq 0$ **then**
  $w \leftarrow \mathbf{0}$

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 2134209-DMS.

References

[1] Claude Berge, *Hypergraphs: combinatorics of finite sets*, vol. 45, Elsevier, 1984.

[2] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar, "Multilevel hypergraph partitioning: applications in vlsi domain," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 1, pp. 69–79, 1999.

[3] Kelly J Pearson and Tan Zhang, "On spectral hypergraph theory of the adjacency tensor," *Graphs and Combinatorics*, vol. 30, no. 5, pp. 1233–1248, 2014.

[4] Joshua Cooper and Aaron Dutle, "Spectra of uniform hypergraphs," *Linear Algebra and its applications*, vol. 436, no. 9, pp. 3268–3292, 2012.

[5] Shenglong Hu and Liqun Qi, "Algebraic connectivity of an even uniform hypergraph," *Journal of Combinatorial Optimization*, vol. 24, no. 4, pp. 564–579, 2012.

[6] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Advances in neural information processing systems*, 2007, pp. 1601–1608.

[7] Luca Corinzia, Paolo Penna, Luca Mondada, and Joachim M Buhmann, "Exact recovery for a family of community-detection generative models," in *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 415–419.

[8] Chiheon Kim, Afonso S Bandeira, and Michel X Goemans, "Community detection in hypergraphs, spiked tensor models, and sum-of-squares," in *2017 International Conference on Sampling Theory and Applications (SampTA)*. IEEE, 2017, pp. 124–128.

[9] Laura Florescu and Will Perkins, "Spectral thresholds in the bipartite stochastic block model," in *Conference on Learning Theory*, 2016, pp. 943–959.

[10] Anru Zhang and Dong Xia, "Tensor svd: Statistical and computational limits," *IEEE Transactions on Information Theory*, vol. 64, no. 11, pp. 7311–7338, 2018.

[11] Wei-Zhi Nie, An-An Liu, Yue Gao, and Yu-Ting Su, "Hyper-clique graph matching and applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 6, pp. 1619–1630, 2018.

[12] Eden Chlamtác, Michael Dinitz, Christian Konrad, Guy Kortsarz, and George Rabanca, "The densest k-subhypergraph problem," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[13] Richard Taylor, "Approximations of the densest k-subhypergraph and set union knapsack problems," *arXiv preprint arXiv:1610.04935*, 2016.

[14] Debarghya Ghoshdastidar, Ambedkar Dukkipati, et al., "Consistency of spectral hypergraph partitioning under planted partition model," *The Annals of Statistics*, vol. 45, no. 1, pp. 289–315, 2017.

[15] Shmuel Friedland and Lek-Heng Lim, "Nuclear norm of higher-order tensors," *Mathematics of Computation*, vol. 87, no. 311, pp. 1255–1281, 2018.

[16] Chiheon Kim, Afonso S Bandeira, and Michel X Goemans, "Stochastic block model for hypergraphs: Statistical limits and a semidefinite programming approach," *arXiv preprint arXiv:1807.02884*, 2018.

[17] Yudong Chen and Jiaming Xu, "Statistical-computational phase transitions in planted models: The high-dimensional setting," in *International Conference on Machine Learning*, 2014, pp. 244–252.

[18] Lixing Han, "An unconstrained optimization approach for finding real eigenvalues of even order symmetric tensors," *Numerical Algebra, Control & Optimization*, vol. 3, no. 3, pp. 583, 2013.

[19] Jure Leskovec and Andrej Krevl, "Snap datasets: Stanford large network dataset collection," 2014.

[20] Austin R Benson, David F Gleich, and Jure Leskovec, "Higher-order organization of complex networks," *Science*, vol. 353, no. 6295, pp. 163–166, 2016.