RESEARCH ARTICLE

Hypergraph partitioning using tensor eigenvalue decomposition

Deepak Maurya*, Balaraman Ravindran

Computer Science and Engineering, Robert Bosch Centre for Data Science and AI, Indian Institute of Technology Madras, Chennai, India

* maurya@cse.iitm.ac.in

Citation: Maurya D, Ravindran B (2023) Hypergraph partitioning using tensor eigenvalue decomposition. PLoS ONE 18(7): e0288457. https://doi.org/10.1371/journal.pone.0288457

Editor: Ilya Safro, University of Delaware, UNITED STATES

Received: June 29, 2021

Accepted: June 27, 2023

Published: July 21, 2023

Data Availability Statement: We have worked with synthetic data and NOT real-world data as the work is more theory oriented. We have included the code to generate the synthetic data as a Supporting information files.

Funding: This work was partially supported by Intel research grant RB/18-19/CSE/002/INTI/BRAV to Balaraman Ravindran The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Abstract

Hypergraphs have gained increasing attention in the machine learning community lately due to their superiority over graphs in capturing *super-dyadic* interactions among entities. In this work, we propose a novel approach for the partitioning of *k*-uniform hypergraphs. Most of the existing methods work by reducing the hypergraph to a graph followed by applying standard graph partitioning algorithms. The reduction step restricts the algorithms to capturing only some weighted pairwise interactions and hence loses essential information about the original hypergraph. We overcome this issue by utilizing *tensor*-based representation of hypergraphs, which enables us to capture actual super-dyadic interactions. We extend the notion of minimum ratio-cut and normalized-cut from graphs to hypergraphs and show that the relaxed optimization problem can be solved using eigenvalue decomposition of the Laplacian tensor. This novel formulation also enables us to remove a hyperedge completely by using the "hyperedge score" metric proposed by us, unlike the existing reduction approaches. We propose a hypergraph partitioning algorithm inspired from spectral graph theory and also derive a tighter upper bound on the minimum positive eigenvalue of even-order hypergraph Laplacian tensor in terms of its conductance, which is utilized in the partitioning algorithm to approximate the normalized cut. The efficacy of the proposed method is demonstrated numerically on synthetic hypergraphs generated by stochastic block model. We also show improvement for the min-cut solution on 2-uniform hypergraphs (graphs) over the standard spectral partitioning algorithm.

1 Introduction

In machine learning, interacting systems are often modeled as graphs. In graph modeling, an interacting object is represented as a node, and an edge captures the interaction between a pair of objects. A conventional approach is to quantify the extent of interaction by associating a positive *weight* to the corresponding edge. This graph formulation is further utilized for various standard machine-learning applications in different domains, such as biology [1], VLSI [2], computer vision [3], transport [4], clustering [5], and semi-supervised learning [6]. Learning on graphs has been an active area of research, ranging from spectral graph theory [7] to recently proposed graph neural networks [8]. A graph representation is limited to capturing only *pairwise* interaction, whereas many real-world systems may involve interactions that may be more complex than the simple pairwise formulation [9]. For instance, a collaboration network may involve agents interacting at a group level (also called super-dyadic interactions), which can not be captured by modeling the system as a graph.

Recently, *hypergraphs* have been used to represent and analyze such complex *super-dyadic* relationships. Hypergraphs are generalizations of graphs where an edge could potentially connect multiple nodes. These edges are commonly referred to as *hyperedges*. A k-uniform hypergraph refers to the case when all hyperedges are constrained to contain exactly k nodes.

Graph partitioning is an interesting problem that involves partitioning the set of nodes in a graph into multiple subsets such that nodes in one subset are more "similar" to each other as compared to nodes in any other subset. Graph partitioning is utilized in various fields such as biology [1], VLSI [2], and computer vision [3]. One of the widely accepted approaches for graph partitioning is minimizing the ratio-cut or normalized-cut [5] objective function using spectra of the graph [7]. Similarly, hypergraph partitioning has been used in a variety of applications in several domains, such as circuit designing [10], image segmentation [11], object segmentation in videos [12], citation networks [13], and semi-supervised learning [14]. In this work, we define the ratio-cut and normalized-cut on hypergraphs and propose a spectral partitioning algorithm.

Existing hypergraph modeling frameworks can be classified into two paradigms, based on whether they reduce the hypergraph to a graph explicitly [15] or implicitly [13, 16]. These reduction based approaches are quite popular in the machine learning community due to the scalability to large datasets [17–19], and provable performance guarantees of graph-based algorithms [20]. Thus most of the existing approaches make use of hypergraph reduction to utilize standard graph-based algorithms, which defeats the motivation behind using hypergraphs. As graphs are limited to capture only dyadic interactions, the reduction-based approaches fail to model the desired super-dyadic relationships.

Ihler et al. [21] show that the reduction-based approaches can not model a hypergraph cut, i.e., the complete removal of a hyperedge from a given hypergraph. After reducing a hypergraph to a graph, partitioning is performed on the graph. During that process, any partitioning algorithm removes some edges from the graph, which is not guaranteed to have any correspondence to the hyperedges in the original hypergraph. Also, note that two or more non-isomorphic hypergraphs may reduce to the same graph. An example for such a case is presented in S1 File. In order to bridge this existing gap, we propose a hypergraph partitioning algorithm in this work, which removes the hyperedges directly without using reduction to the graph. We use the tensor representation of hypergraphs and further the tensor eigenvalue decomposition for hypergraph partitioning. Note that tensor eigenvalue decomposition is NP-hard for general tensors and cannot be approximated unless P = NP [22].

Tensors have gained increasing attention for modeling hypergraphs, primarily in the mathematics community. For instance, Hu et al. [23] extended the fundamental and well-known theorem in spectral graph theory relating cardinality of zero eigenvalue of the Laplacian of a graph to the number of connected components to the uniform hypergraphs. Specifically, they proved that the algebraic multiplicity of zero eigenvalue of a symmetric Laplacian tensor is equal to the sum of the number of even-bipartite connected components and the number of connected components excluding the number of singletons in the given hypergraph. Such insights can not be revealed from the clique reduction methods and its variants [15]. In the machine learning community, tensor representation of hypergraphs has not gained much attention, except for a few works [24, 25]. In this work, we utilize the tensor representation of hypergraphs for detecting densely connected components by extending the notion of ratio-cut and normalized from graphs to hypergraphs [26]. We propose the novel "hyperedge score" that captures the structural variation of multiple nodes in a hyperedge [27, 28]. The key contributions and outline of this work are presented in the following subsection.

1.1 Our contributions

We make the following contributions in this work:

- We propose the ratio-cut and normalized cut for k-uniform hypergraphs. Further, we prove that the solution to the minimization of relaxed ratio-cut or normalized cut problem can be obtained from the eigenvector corresponding to the minimum positive eigenvalue of the unnormalized and normalized Laplacian tensor respectively.
- We propose a novel metric termed as "hyperedge score", which is defined over each existing hyperedge and is a function of the eigenvector corresponding to minimum positive eigenvalue. This hyperedge score metric is used by our partitioning algorithm to remove the hyperedge directly without performing any reduction on hypergraphs [15].
- We also derive a tighter upper bound on the minimum positive eigenvalue of the normalized Laplacian tensor in terms of hypergraph conductance for even order hypergraphs.
- We demonstrate the efficacy of the proposed algorithm on synthetic hypergraphs ($k = 2$ and $k = 4$) generated by stochastic block model (SBM).
- We compare the performance on synthetic graphs (2-uniform hypergraphs) generated by SBM. We also report $n/8$ times improvement of ratio-cut over the conventional spectral partitioning for *cockroach graph*, where n is the number of nodes.

1.2 Outline

The preliminaries of hypergraph notation and tensor representation are covered in Section 2. The proposed hypergraph partitioning algorithm is presented in Section 3. The functioning and efficacy of the proposed algorithm is demonstrated in Section 4 by experiments on synthetic and real hypergraphs. The main manuscript ends with concluding remarks in Section 5. The numerical details of the illustrative examples are presented in the S1 File after references.

2 Preliminaries

In this section, we briefly discuss the prevalent approach of representing hypergraphs and their partitioning. A hypergraph G is defined as a pair of $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of entities called vertices or nodes and $E = \{e_1, e_2, \ldots, e_m\}$ is a set of non-empty subsets of V referred to as hyperedges.

The strength of interaction among nodes in the same hyperedge is quantified by the positive weight represented by $w_e = \{w_{e_1}, w_{e_2}, \ldots, w_{e_m}\}$.

The vertex-edge incidence matrix is denoted by $\mathbf{H}$ and has the dimension $|V| \times |E|$. The entry $h(i, j)$ is defined to be 1 if $v_i \in e_j$ and 0 otherwise.

The degree of node v_i is defined by $d_{v_i} = \sum_{e_j \in E} w_{e_j} h(i, j)$. We can also define two diagonal matrices, $\mathbf{W}$, $\mathbf{D}$, with the dimension of $m \times m$, $n \times n$, containing the hyperedge weights and node degrees respectively. Note that there is no loss of information in this form of representation of hypergraphs until this point. This implies that a unique hypergraph can be constructed for a given incidence matrix.

2.1 Reducing a hypergraph to a graph

Now, we discuss the widely-accepted approach for hypergraph reduction in the machine learning community. The fundamental idea is to reduce a hypergraph to graph and subsequently apply standard graph-based algorithms. In this subsection, we briefly discuss the merits and demerits of these approaches and articulate the reasons for choosing the tensor based representation of hypergraphs.

Definition 1. *The clique expansion for hypergraph $G(V, E)$ builds a graph $G_x(V, E_x \subseteq V^2)$ by replacing each hyperedge with the corresponding clique, $E_x = \{(v_i, v_j): v_i, v_j \in e_l, e_l \in E\}$* [15]. *The edge weight $w_x(u, v)$ is given by $w_x(u, v) = \sum_{u,v \in e_l, e_l \in E} w(e_l)$.*

The same could be stated in matrix form as

$$\mathbf{A} = \mathbf{H}\mathbf{W}\mathbf{H}^T - \mathbf{D} \tag{1}$$

where **A** represents the adjacency matrix for reduced hypergraph. Another traditional hypergraph reduction approach is star expansion [29]. Most of the other reduction approaches are build on these. Please see [15] and the references therein, for more details.

This reduction step is very convenient as we can now employ any graph algorithms that scale well and come with theoretical guarantees. A natural question arises on the need for these different reduction based approaches. We believe that each of these reduction approaches preserves a few *but* not all hypergraph properties in the reduction step. The preserved hypergraph property may be useful for the end task of learning on hypergraphs. For example, clustering results can be improved on hypergraphs by preserving node degrees during reduction [30].

More often, the reduction step loses vital information about hypergraphs as two different hypergraphs can reduce to the same graph. This can be seen directly from Eq (1) as two distinct hypergraphs having different **H** and **W** can reduce to the same adjacency matrix **A**. An illustrative example of the same is presented in S1 File.

2.2 Tensor representation of hypergraphs

In this subsection, we briefly review the tensor-based representation of hypergraphs [31, 32]. A natural representation of hypergraphs is a k-order n-dimensional tensor $\mathcal{A}$, which consists of n^k entries and is defined by:

$$a_{i_1 i_2 \ldots i_k} = \begin{cases} \dfrac{w_{e_j}}{(k-1)!} & \text{if} \quad \{i_1, i_2, \ldots, i_k\} \in E, \quad 1 \leq i_1, \ldots, i_k \leq n \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

It should be noted that $\mathcal{A}$ is a "*super-symmetric*" tensor, i.e, $a_{i_1 i_2 \ldots i_k} = a_{\sigma(i_1 i_2 \ldots i_k)}$, where $\sigma(i_1, i_2, \ldots i_k)$ denotes any permutation of the elements in the set $\{i_1, i_2, \ldots, i_k\}$. The order or mode of the tensor refers to the hyperedge cardinality, which is k for $\mathcal{A}$. The degree of all the vertices can be represented by k-order n-dimensional diagonal tensor $\mathcal{D}$. The Laplacian tensor $\mathcal{L}$ is defined as follows:

$$\mathcal{L} = \mathcal{D} - \mathcal{A} \tag{3}$$

An example demonstrating the tensor representation of a 4-uniform hypergraph is presented in S1 File. The normalized Laplacian tensor, denoted by $\mathscr{L}$ can also be defined in a similar manner:

$$\ell_{i_1 i_2 \ldots i_k} = \begin{cases} -\dfrac{w_{e_j}}{(k-1)!}\prod_{i_j=1}^{k} \dfrac{1}{\sqrt[k]{d_{i_j}}} & \text{if} \quad \{i_1, i_2, \ldots, i_k\} \in E \\ 1 & \text{if } i_1 = i_2 \ldots = i_k = i, \qquad i = \{1, 2, \ldots, n\} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

For the sake of completeness, we define the tensor eigenvalue decomposition as:

$$\mathcal{L}\mathbf{x}^{k-1} = \lambda\mathbf{x}, \qquad \text{such that} \quad \mathbf{x}^T\mathbf{x} = 1 \tag{5}$$

where $(\lambda, \mathbf{x}) \in (\mathbb{R}, \mathbb{R}^n \setminus \{0\}^n)$ is called the Z-eigenpair and $\mathcal{L}\mathbf{x}^{k-1} \in \mathbb{R}^n$, whose i^{th} component is:

$$[\mathcal{L}\mathbf{x}^{k-1}]_i = \sum_{i_k=1}^{n} \ldots \sum_{i_3=1}^{n}\sum_{i_2=1}^{n} l_{i i_2 i_3 \ldots i_k} x_{i_2} x_{i_3} \ldots x_{i_k} \tag{6}$$

The expression for the tensor Laplacian of a hypergraph ($\mathcal{L}\mathbf{x}^k$) can be computed using the above and $\mathcal{L}\mathbf{x}^k = (\mathcal{L}\mathbf{x}^{k-1})^T\mathbf{x}$. This is a k^{th} order polynomial in n variables which can be simplified as stated in the following theorem.

Theorem 2. *The expression for tensor Laplacian of a hypergraph can be simplified using*

$$\begin{aligned} \mathcal{L}\mathbf{x}^k &= \sum_{i_1, i_2, \ldots, i_k = 1}^{n} l_{i_1 i_2 \ldots i_k} x_{i_1} x_{i_2} \ldots x_{i_k} \\ &= \sum_{e_j \in E} w_{e_j} \left(\sum_{i_t \in e_j} x_{i_t}^k - k \prod_{i_t \in e_j} x_{i_t} \right) = \sum_{e_j \in E} w_{e_j} k \left(\underset{i_t \in e_j}{AM}(x_{i_t}^k) - \underset{i_t \in e_j}{GM}(|x_{i_t}|^k)(-1)^{n_{s,j}} \right) \end{aligned} \tag{7}$$

where $n_{s,j} = |\{i_t : x_{i_t} < 0, i_t \in e_j\}|$, *AM and GM stand for arithmetic and geometric means respectively.*

Proof.

$$\begin{aligned} \mathcal{L}\mathbf{x}^k &= \sum_{i_k=1}^{n} \ldots \sum_{i_2=1}^{n}\sum_{i_1=1}^{n} (d_{i_1 i_2 \ldots i_k} - a_{i_1 i_2 \ldots i_k}) x_{i_1} x_{i_2} \ldots x_{i_k} \\ &= \sum_{i=1}^{n} d(v_i) x_i^k - \sum_{i_k=1}^{n} \ldots \sum_{i_2=1}^{n}\sum_{i_1=1}^{n} a_{i_1 i_2 \ldots i_k} x_{i_1} x_{i_2} \ldots x_{i_k} \\ &= \sum_{i=1}^{n}\sum_{i_k=1}^{n} \ldots \sum_{i_3=1}^{n}\sum_{i_2=1}^{n} a_{i i_2 i_3 \ldots i_k} x_i^k - \sum_{i_k=1}^{n} \ldots \sum_{i_2=1}^{n}\sum_{i_1=1}^{n} a_{i_1 i_2 \ldots i_k} x_{i_1} x_{i_2} \ldots x_{i_k} \\ &= \sum_{i=1}^{n} \left(\sum_{(i_2, i_3, \ldots, i_k) \in e_j} \frac{w_{e_j}}{(k-1)!} x_i^k \right) - \left(\sum_{(i_1, i_2, \ldots, i_k) \in e_j} \frac{w_{e_j}}{(k-1)!} x_{i_1} x_{i_2} \ldots x_{i_k} \right) \end{aligned} \tag{8}$$

As there are $(k - 1)!$ and $k!$ permutations of the first and second term respectively:

$$\begin{aligned}
\mathcal{L}\mathbf{x}^k &= \sum_{e_j \in E} w_{e_j} \left(\sum_{i_t \in e_j} \frac{(k-1)!}{(k-1)!} x_{i_t}^k - \frac{k!}{(k-1)!} x_{i_1} x_{i_2} \ldots x_{i_k} \right) \\
&= \sum_{e_j \in E} w_{e_j} \left(\sum_{i_t \in e_j} x_{i_t}^k - k \prod_{i_t \in e_j} x_{i_t} \right) \\
&= \sum_{e_j \in E} w_{e_j} \left(k \frac{\sum_{i_t \in e_j} x_{i_t}^k}{k} - k \left(\prod_{i_t \in e_j} |x_{i_t}|^k \right)^{\frac{1}{k}} (-1)^{n_{s,j}} \right) \\
&= \sum_{e_j \in E} w_{e_j} k \left(\underset{i_t \in e_j}{AM}(x_{i_t}^k) - \underset{i_t \in e_j}{GM}(|x_{i_t}|^k) \, (-1)^{n_{s,j}} \right)
\end{aligned} \tag{9}$$

The above polynomial expression can be viewed as generalization of the graph, as for any edge $\{a, b\}$, the objective function $(x_a - x_b)^2 = x_a^2 + x_b^2 - 2x_a x_b$.

Theorem 3. *The expression for the normalized tensor Laplacian of a hypergraph*

$$\mathscr{L}\mathbf{x}^k = \sum_{e_j \in E} w_{e_j} \left(\sum_{i_t \in e_j} \frac{x_{i_t}^k}{d_{i_t}} - k \prod_{i_t \in e_j} \frac{x_{i_t}}{\sqrt[k]{d_{i_t}}} \right) = \sum_{e_j \in E} w_{e_j} k \left(\underset{i_t \in e_j}{AM} \left(\frac{x_{i_t}^k}{d_{i_t}} \right) - \underset{i_t \in e_j}{GM} \left(\frac{|x_{i_t}|^k}{d_{i_t}} \right) (-1)^{n_{s,j}} \right)$$

where $n_{s,j} = |\{i_t : x_{i_t} < 0, i_t \in e_j\}|$.

Proof. Similar to Theorem 2

This theorem for hypergraphs will be used in the later sections for proving other theorems. With basics covered in this section, we focus on the main problem of hypergraph partitioning in the next section.

3 Partitioning of hypergraphs

We start this section with brief review of spectral graph theory for partitioning of graphs [26] and further propose these ideas for hypergraphs.

3.1 Partitioning of graphs

Let the p parts of a partition of vertex set V be denoted by sets $C_1, C_2, \ldots, C_p$ such that

$$C_i \neq \emptyset, \quad C_i \subset V, \quad \cup_{i=1}^{p} C_i = V, \quad C_i \cap C_j = \emptyset, \quad \forall i, j \in [p], \quad \text{and} \quad i \neq j \tag{10}$$

The two most commonly used objective function of graph partitioning are Ratio cut [33] and Normalized cut [34]:

$$\text{Ratio Cut}(C_1, C_2, \ldots, C_p) = \sum_{i=1}^{p} \frac{\text{cut}(C_i, \bar{C}_i)}{2|C_i|}, \quad \text{where} \quad \text{cut}(C_i, \bar{C}_i) = \sum_{r \in C_i, s \in \bar{C}_i} w_{rs} \tag{11}$$

$$\text{Normalized Cut}(C_1, C_2, \ldots, C_p) = \sum_{i=1}^{p} \frac{\text{cut}(C_i, \bar{C}_i)}{2\text{vol}(C_i)}, \quad \text{where} \quad \text{vol}(C_i) = \sum_{r \in C_i} d_r \tag{12}$$

where w_{rs} denotes the weight of the edge between nodes r and s, and d_r denotes the degree of

r^{th} node. It is well known that the solution to the relaxed version of minimizing the ratio cut and normalized cut can be obtained from the Fiedler vector of unnormalized and normalized Laplacians, respectively.

The approximation made in the relaxation step is theoretically analyzed [35–37].

3.2 Ratio-cut and normalized-cut for hypergraphs

We start the discussion with a formal description of the problem. Let $C_1, C_2, \ldots, C_p$ be the p parts of a partition as defined in Eq (10). For a given hypergraph $G(V, E, W_e)$, we intend to remove a subset of hyperedges $\partial E \subseteq E$, such that $G \setminus \partial E$ produces a partition with at least p disjoint parts [38, 39]. The hyperedge boundary ∂E can be defined as:

$$\partial E = \{e_j \in E : e_j \cap C_i \neq \varnothing, e_j \cap \bar{C}_i \neq \varnothing\} \tag{13}$$

for some $i \in [p]$. It basically denotes the set of hyperedges which "cross" the parts of the partition. The next step is to define the objective function to be minimized for obtaining optimal partitions. The measures described in Eqs (11) and (12) is proposed for graphs and hence not well-suited for hypergraphs as discussed in Example 1 shortly. We propose the following generalization of ratio-cut and normalized-cut for hypergraphs.

Definition 4. *The cut cost for the* i^{th} *part* C_i *is denoted by* $w_h(C_i)$ *and the total cut cost denoted by* $w_{h,t}(V)$ *for all the* p *parts of a partition is defined as*:

$$w_h(C_i) = \sum_{e_j \in \partial E} |C_i \cap e_j| w_{e_j}, \qquad w_{h,t}(V) = \frac{1}{k} \sum_{i=1}^{p} w_h(C_i) \tag{14}$$

The cut cost for a partition and total cut cost defined in Eq (14) reduces to numerator term in Eqs (11) and (12) for $k = 2$ because the term $|C_i \cap e_j|$ reduces to unity $\forall e_j \in \partial E$ in graphs. We further demonstrate the merits of this cut cost by the following example.

Example 1. *Consider the 3-uniform hypergraph shown in* Fig 1. *Consider the partitions obtained after removing hyperedges* e_2 *and* e_3. *Let* $C_1 = \{v_1, v_2, v_3\}$, $C_2 = \{v_4\}$, $C_3 = \{v_5\}$. *The*

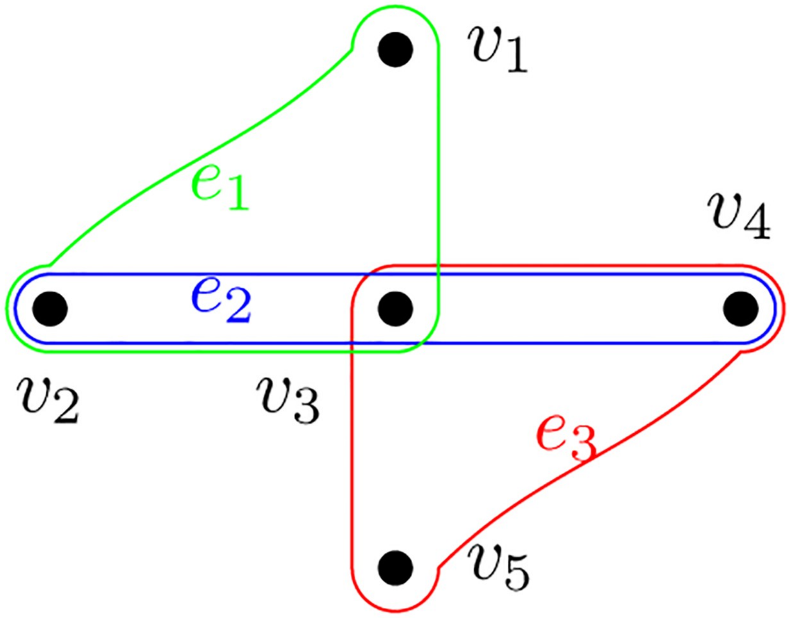


Fig 1. Hypergraph: H_1.

https://doi.org/10.1371/journal.pone.0288457.g001

partition cost is given by

$$\begin{aligned} w_h(C_1) &= 2w_{e_2} + w_{e_3}, \quad w_h(C_2) = w_{e_2} + w_{e_3}, \quad w_h(C_3) = w_{e_3} \\ w_{h,t}(V) &= w_{e_2} + w_{e_3} \end{aligned}$$

It should be noted that w_{e_1} is not reflected in the above cut costs because hyperedge e_1 is not cut. It could be easily verified that this cut cost is not equivalent to clique reduction approach. The cut costs derived for the reduced hypergraph are as follows:

$$\begin{aligned} w_g(C_1) &= 2(w_{e_2} + w_{e_3}), \quad w_g(C_2) = 2(w_{e_2} + w_{e_3}), \quad w_g(C_3) = 2w_{e_3} \\ w_{h,t}(V) &= 2w_{e_2} + 3w_{e_3} \end{aligned}$$

Note that the cut costs derived from both approaches are different. On further inspection, we infer $w_g(C_i) = 2w_h(C_i)$ for $i = \{2, 3\}$, which means the cut cost for partitions C_2 and C_3 in the reduced hypergraph are just a scaled version of costs involved in original hypergraph. The same relation does not hold for partition C_1 due to the presence of the term $|C_i \cap e_j|$ in Eq (14). Please refer to S1 File for the computation of these cut costs.

From this illustrative example, it can be inferred that the proposed cut cost for hypergraphs defined in Eq (14) carries more information about the cut as compared to reduced hypergraphs. The term $|C_i \cap e_j|$ in Eq (14) will lead to a greater penalty for removing hyperedges with more elements from C_i. A hyperedge with higher $|C_i \cap e_j|$ is likely to have more association with partition C_i, so the corresponding cut should be penalized more.

Minimizing the total cut cost defined in Eq 14 directly may lead to "unbalanced" partitions with minimum cost. To bypass such trivial and undesirable partitions, we propose the normalization.

Definition 5. *The Ratio-cut and Normalized-cut for p partitions are defined as*:

$$Ratio - Cut(C_1, C_2, \ldots, C_p) = \sum_{i=1}^{p} \frac{w_h(C_i)}{k|C_i|^{k/2}} \tag{15}$$

$$N - Cut(C_1, C_2, \ldots, C_p) = \sum_{i=1}^{p} \frac{w_h(C_i)}{k(\mathrm{vol}(C_i))^{k/2}} \tag{16}$$

where $w_h(C_i)$ is defined in Eq (14). The above term for ratio-cut and normalized-cut simplifies to Eqs (11) and (12) respectively for $k = 2$. Compared to the similar objective function proposed in literature [13, 19], our objective function normalizes the exponential factor in the denominator. This helps us bypass the partitions with singletons or fewer nodes compared to normalization less than the exponential factor (like linear). Another perspective can be seen from the motivation behind the introduction of $|C_i|$ or $|vol(C_i)|$ term in the denominator of ratio-cut and normalized cut for graphs ($k = 2$). An exponential factor of this normalization factor probably helps us to produce more balanced partitions, which is very much required for hypergraphs. For example, consider a hypergraph with one hyperedge $e_1 = \{v_1, v_2, v_3\}$ with 3 nodes v_1, v_2, and v_3. Cutting one hyperedge will produce three singletons which we consider as three partitions. A similar definition of normalized associativity can be seen in literature [20, 40].

3.3 Hypergraph partitioning algorithm

We wish to find the partition $C_1, \ldots, C_p$ which minimizes the ratio-cut or normalized-cut. It should also be noted that p is fixed. For further discussion, we focus on the minimization of ratio-cut, and the same approach can be extended for normalized-cut, as shown later. The optimal partitions can be obtained by solving:

$$(C_1, C_2, \ldots, C_p) = \underset{(C_1, C_2, \ldots, C_p)}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{p} \frac{w_h(C_i)}{|C_i|^{k/2}} \tag{17}$$

Unfortunately, the above problem is NP-hard [26, 41–43]. Inspired from spectral graph theory, we propose to solve a relaxed version of the optimization problem mentioned above.

Theorem 6. *The minimization of ratio-cut in* Eq (15) *can be equivalently expressed as*

$$\min\left(\sum_{i=1}^{p} \mathcal{L}\mathbf{f}_i^k\right) = \min\left(\sum_{i=1}^{p} \sum_{e_j \in \partial E} |C_i \cap e_j| \frac{w_{e_j}}{|C_i|^{k/2}}\right), \qquad f_{i,j} = \begin{cases} \frac{1}{\sqrt{|C_j|}} & v_i \in C_j \\ 0 & \textit{otherwise} \end{cases} \tag{18}$$

where we define p indicator vectors $\mathbf{f}_j$ *and its* i^{th} *element, denoted by* $f_{i,j}$ *indicates if the vertex* v_i *belongs to* j^{th} *part of partition, denoted by* C_j. *The solution to the above problem after relaxing* $\mathbf{f}_i \in \mathbb{R}^n$ *rather than an indicator vector can be derived from the eigenvector corresponding to the minimum positive eigenvalue stated in* Eq (5).

Proof. Given a partition of p disjoint sets $\{C_1, C_2, \ldots, C_p\}$, define the p indicator variables $\mathbf{f}_j = (f_{1,j}, f_{2,j}, \ldots, f_{n,j})^\top$ defined as

$$f_{i,j} = \begin{cases} \frac{1}{\sqrt{|C_j|}} & v_i \in C_j \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

where $i \in [n]$ and $j \in [p]$.

For any part C_i, we compute $\mathcal{L}\mathbf{f}_i^k$

$$\mathcal{L}\mathbf{f}_i^k = \sum_{i_k=1}^{n} \cdots \sum_{i_2=1}^{n} \sum_{i_1=1}^{n} l_{i_1 i_2 \ldots i_k} f_{i_1,i} f_{i_2,i} \cdots f_{i_k,i} \tag{20}$$

We use Theorem 2 to compute the above term

$$\mathcal{L}\mathbf{f}_i^k \quad = \sum_{e_j \in E} w_{e_j} \left(\sum_{i_t \in e_j} f_{i_t,i}^k - k \prod_{i_t \in e_j} f_{i_t,i} \right) \tag{21}$$

There can be three cases for each hyperedge:

1. $e_j \subseteq C_i$: All the nodes in a hyperedge e_j are assigned as $\frac{1}{|C_i|^{1/2}}$. Both the terms ($\sum_{i_t \in e_j} f_{i_t,i}^k$ and $k\prod_{i_t \in e_j} f_{i_t,i}$) will be $k\frac{1}{|C_i|^{k/2}}$ and the overall term ($\mathcal{L}\mathbf{f}_i^k$) reduces to 0.
2. $e_j \subseteq \bar{C}_i$: All the nodes in hyperedge e_j are assigned 0. Both the terms will be zero and overall term will be zero.
3. $e_j \in \partial E$: Some of the nodes are assigned $\frac{1}{\sqrt{|C_i|}}$. The second term ($k\prod_{i_t \in e_j} f_{i_t,i}$) will be zero and the first term ($\sum_{i_t \in e_j} f_{i_t,i}^k$) will reduce to $|C_i \cap e_j| \frac{w_{e_j}}{|C_i|^{k/2}}$.

So the overall term reduce to

$$\mathcal{L}\mathbf{f}_i^k = \sum_{e_j \in \partial E} w_{e_j} |C_i \cap e_j| \frac{w_{e_j}}{|C_i|^{k/2}} \tag{22}$$

Summing over the parts, we arrive at

$$\sum_{i=1}^{p} \mathcal{L}\mathbf{f}_i^k = \sum_{i=1}^{p} \sum_{e_j \in \partial E} w_{e_j} |C_i \cap e_j| \frac{w_{e_j}}{|C_i|^{k/2}} \tag{23}$$

The RHS term in Eq (23) is same as the defined ratio cut for hypergraphs (Eq (15)). It should be noted that $\mathbf{f}_\mathbf{i}^T \mathbf{f}_\mathbf{i} = 1$. As the objective function and constraint are the same under relaxation, the solution to the relaxed optimization problem can be derived from tensor eigenvalue decomposition.

We continue the discussion on partitioning with the following example. Note that a certain ratio-cut approximation is involved while utilizing Theorem 6 for proposing the hypergraph partitioning algorithm using tensor EVD. This approximation is theoretically analyzed in Theorem 8 and Theorem 9.

Example 2. *Consider the 3-uniform hypergraph shown in Fig 2. The colored number indicates the hyperedge weight. It is clear that the optimal partitions are $A_1 = \{2, 3, 4, 5, 6, 7\}$ and $\bar{A}_1$. The Fiedler eigenvector for this hypergraph is*

$$\mathbf{f}^\star = [0.33 \quad 0.16 \quad 0.17 \quad 0.13 \quad -0.05 \quad 0.05 \quad 0.12 \quad 0.39 \quad 0.38 \quad 0.43 \quad 0.39 \quad 0.38]$$

A standard approach in spectral graph theory is to use the sign of the elements in the Fiedler vector for partitioning [26]. *For example, $C_1 = \{i | \mathbf{f}^\star(i) < 0, i \in [n]\}$.Hence, the partitions are $C_1 = \{5\}$ and $C_2 = V \backslash C_1$, which is clearly not optimal.*

From the above example, it is clear that the traditional approach of partitioning does not yield desired partitions for hypergraphs. This is primarily because the eigenvectors of the Laplacian tensor of a hypergraph can not be interpreted in the same way as the eigenvectors of the Laplacian matrix of a graph.

To understand the implication of minimum ratio-cut associated with minimum positive $\lambda^\star$, we analyze the computation of Laplacian objective function using the Fiedler vector:

$$l_{e_j}(\mathbf{f}^\star) = w_{e_j} \left(\sum_{i_t \in e_j} f_{i_t}^k - k \prod_{i_t \in e_j} f_{i_t} \right), \qquad \lambda^\star = \sum_{e_j \in E} l_{e_j}(\mathbf{f}^\star) \tag{24}$$

where $l_{e_j}(\mathbf{f}^\star)$ denotes the "score" for hyperedge e_j computed for the eigenvector $\mathbf{f}^\star$. With a slight

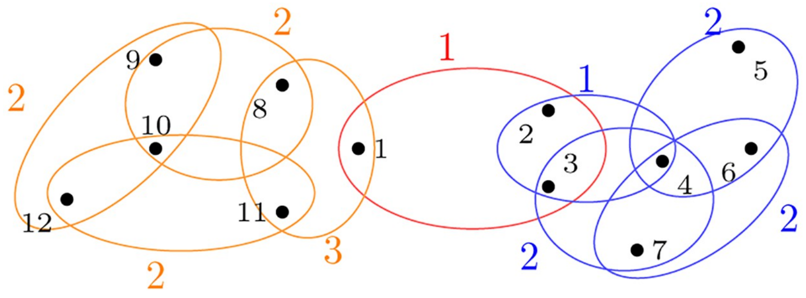


Fig 2. Hypergraph: H_2.

https://doi.org/10.1371/journal.pone.0288457.g002

abuse of terminology, we argue that a higher value of this score indicates the corresponding hyperedges are "close" to separator boundary ∂E. The measure of closeness between two nodes is quantified by the minimum number of hyperedges to be traversed for reaching one node to another.

This can be validated easily by careful inspection of hyperedge score $l_{e_j}(\mathbf{f})$, when the vector $\mathbf{f}$ is treated as the cluster indicator variable shown in Eq (18). The hyperedge score will be non-zero only for the hyperedges on the separating boundary for such ideal choice of $\mathbf{f}$. The same can be also interpreted as the score being zero $\forall e_j \in \{E \setminus \partial E\}$. We carry forward the same intuition and prefer to cut the hyperedges with a "higher" score.

The score may not be exactly zero for any hyperedge if the Fiedler vector is used for the score computation as it is obtained for the relaxed minimization of the ratio-cut (Theorem 6). Applying this approach on Example 2, we report a maximum score of 0.017 for the hyperedge {1, 2, 3} and hence cut it to obtain the optimal partitions. It should be noted that we obtain the optimal partitions directly without computing the ratio-cut value $n - 1$ times and taking minimum value like the existing sweep cut-based approaches [33]. The proposed algorithm is summarized in Algorithm 1.

Algorithm 1: Hypergraph Partitioning Algorithm

```
Result: Partitions
Construct the tensor Laplacian and derive the Fiedler eigenpair ($\lambda^\star$, $\mathbf{f}^\star$).
Calculate the hyperedge score, $l_{e_j}(\mathbf{f}^\star)$ by using Eq (24).
while number of parts < $p$ do
  Remove hyperedges with maximum cost (hyepredge score).
end
```

The intuition behind using the hyperedge score for deriving ∂E is motivated from spectral graph theory. It is interesting to note that this novel use of hyperedge scores helps to compute a better ratio-cut for the cockroach graph presented in Section 4.

A similar analysis can be performed for the minimization of the normalized cut of hypergraphs.

Corollary 7. *The solution to the relaxed optimization problem of minimizing normalized cut mentioned in Proposition 5 can be derived using the eigenvector corresponding to the minimum positive eigenvalue of the normalized Laplacian tensor defined in* Eq (4).

The proof of this corollary is very similar to the proof of Theorem 6. In this case, we choose the indicator variable as

$$f_{i,j} = \begin{cases} \frac{1}{\sqrt{\mathrm{vol}(C_j)}} & v_i \in C_j \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

The next step is to compute the $\mathcal{L}\mathbf{x}^k$, where the normalized Laplacian tensor $\mathcal{L}$ is defined in Eq 4. The rest of the proof is very similar to the proof of Theorem 6.

We perform the theoretical analysis of the proposed algorithm and derive an interesting bound on the approximation made in normalized cuts.

Theorem 8. *The upper bound on the minimum positive eigenvalue of an even order k-uniform hypergraph is*

$$\lambda_1 \leq k\phi(G), \qquad \phi(G) = \min_{C \subseteq V} \frac{\sum_{e_j \in \partial E} w_{e_j}}{\min\{vol(C), \mathrm{vol}(\bar{C})\}}, \qquad \mathrm{vol}(C) = \sum_{i_j \in C} d_{i_j} \tag{26}$$

where λ_1 is the smallest eigenvalue satisfying Eq (5) *for normalized tensor Laplacian $\mathcal{L}$ and $\phi(G)$ refers to the conductance of hypergraph.*

Proof. Let $\mathbf{x}$ be a $n \times 1$ vector with $x_{i_t} \in \left\{ \frac{\sqrt[k]{d_{i_t}}}{\omega}, \frac{-\sqrt[k]{d_{i_t}}}{\omega} \right\}$, where ω is defined as

$$\omega = \left(\sum_{i_t=1}^{n} d_{i_t}^{\frac{2}{k}} \right)^{\frac{1}{2}}$$

It can be easily verified that $\mathbf{x}^T \mathbf{x} = 1$. Substitute $\mathbf{x}$ in the expression for normalized hypergraph Laplacian defined in Eq (4). Please note that the signs of x_{i_t} correspond to an arbitrary cut $(C, \bar{C})$.

$$\begin{aligned} \lambda_1 \leq \mathcal{L}\mathbf{x}^k &= \sum_{e_j \in E} w_{e_j} \left(\sum_{i_t \in e_j} \frac{x_{i_t}^k}{d_{i_t}} - k \prod_{i_t \in e_j} \frac{x_{i_t}}{\sqrt[k]{d_{i_t}}} \right) \\ &= \sum_{e_j \in E} w_{e_j} \left(\frac{k - n_{s,j}}{\omega^k} + \frac{n_{s,j}(-1)^k}{\omega^k} - k(-1)^{n_{s,j}} \frac{1}{\omega^k} \right) \end{aligned} \tag{27}$$

where $n_{s,j} = |\{i_t : x_{i_t} < 0, i_t \in e_j\}|$. For even order hypergraphs, the above can be reduced to

$$\begin{aligned} \lambda_1 &\leq \sum_{e_j \in E} w_{e_j} \left(\frac{k}{\omega^k} - k(-1)^{n_{s,j}} \frac{1}{\omega^k} \right) \\ &\leq \sum_{e_j \in \partial E} w_{e_j} \left(\frac{2k}{\omega^k} \right) = \sum_{e_j \in \partial S} w_{e_j} \left(\frac{2k}{\left(\sum_{i=1}^n d_i^{\frac{2}{k}} \right)^{\frac{k}{2}}} \right) \\ &\leq \sum_{e_j \in \partial E} w_{e_j} \left(\frac{2k}{(\sum_{i=1}^n d_i)^{\frac{2}{k} \times \frac{k}{2}}} \right) \\ &\leq \sum_{e_j \in \partial E} w_{e_j} \frac{2k}{2 \min\left(\mathrm{vol}(C), \mathrm{vol}(\bar{C})\right)} \\ &\leq k\phi(G) \qquad \text{for any } C \subseteq V \end{aligned} \tag{28}$$

This theorem helps to analyze the order of approximation involved in relaxing the N-min cut problem by deriving the solution through tensor EVD. The tightness of the bound indicates the goodness of the approximation. Several other attempts have been made to derive such approximation bounds for hypergraphs. For example, Chen et al. [27] utilize a different Laplacian tensor and the following hyperedge score to derive similar bound on λ_1 of a different tensor:

$$l_{e_j}(\mathbf{x}) = \sum_{i_k \in e_j} (x_{i_k} - \bar{x})^k, \quad \bar{x} = \frac{1}{k}\sum_{i_k \in e_j} x_{i_k}, \qquad \lambda_1 \leq 2^{k/2}\phi(G) \tag{29}$$

This is a weaker bound of exponential nature whereas we have proposed tighter bound of linear nature in Theorem 8.

Theorem 9. *The upper bound on the minimum positive eigenvalue of the unnormalized Laplacian tensor of an even order k-uniform hypergraph is*:

$$\lambda_1 \leq k\phi_r(G), \qquad \phi_r(G) = \min_{C \subseteq V} \frac{\sum_{e_j \in \partial E} w_{e_j}}{\min\{|C|, |\bar{C}|\}}, \qquad 30)$$

Proof. Let $\mathbf{x}$ be $n \times 1$ vector with $x_{i_j} \in \{\frac{1}{\omega}, \frac{-1}{\omega}\}$, where ω is defined as

$$\omega = (|V|)^{\frac{1}{2}}$$

Please note that $\mathbf{x}^T \mathbf{x} = 1$. Proceeding in a similar manner to the proof of Theorem 8:

$$\begin{aligned}
\lambda_1 \leq \mathcal{L}\mathbf{x}^m \quad &= \sum_{e_j \in E} w_{e_j}\left(\sum_{i_t \in e_j} x_{i_t}^k - k\prod_{i_t \in e_j} x_{i_t}\right) \\
&= \sum_{e_j \in E} w_{e_j}\left(\frac{k - n_{s,j}}{\omega^k} + \frac{n_{s,j}(-1)^k}{\omega^k} - k(-1)^{n_{s,j}}\frac{1}{\omega^k}\right) \\
&\leq \sum_{e_j \in \partial E} w_{e_j}\left(\frac{2k}{\omega^k}\right) = \sum_{e_j \in \partial S} w_{e_j}\left(\frac{2k}{|V|^{k/2}}\right) \\
&\leq \sum_{e_j \in \partial E} w_{e_j}\left(\frac{2k}{|V|}\right) \\
&\leq \sum_{e_j \in \partial E} w_{e_j}\left(\frac{2k}{2\min|C|, |\bar{C}|}\right) \\
&\leq k\phi(G) \qquad \text{for any } C \subseteq V
\end{aligned}$$

where $n_{s,j} = |\{i_t : x_{i_t} < 0, i_t \in e_j\}|$ and $\phi_r(G)$ is defined in Eq 30.

It should be noted $\phi_r(G)$ defined in Eq 30 is a slightly modified version of the conductance, $\phi(G)$ defined in Eq 26. Also, note that $\phi_r(G) = d \times \phi(G)$ for d-regular hypergraph because $vol(C) = d \times |C|$ for this particular case. A d-regular hypergraph is a hypergraph where each node is constrained to have degree of exactly d.

3.4 Computation of tensor eigenvectors

The computation of eigenvectors of real super-symmetric tensors is quite challenging and not straightforward as in the case of real symmetric matrices. It is actually NP-hard for general tensors and cannot be approximated unless P = NP [22]. This is primarily due to the non-orthogonality of tensor eigenvectors. There are several other challenging aspects, for example, real symmetric tensors can have complex eigenpairs, unlike the case of matrices. Also, a real symmetric matrix of size $n \times n$ can have a maximum of n eigenvalues, whereas a tensor can have much larger number of eigenpairs [31, 44]. Most of the existing works on computation of eigenpairs have been for tensors with special structure [45] or the extreme eigenvalues such as maximum or minimum eigenvalue [46, 47]. As discussed in Section 3.3, only the Fiedler vector is required for partitioning a given hypergraph. As the Fiedler vector is not one of the extreme eigenvectors, the above methods are not helpful for our case.

Recently proposed algorithm to compute all the eigenvalues of a tensor utilizes homotopy methods [48]. They pose the problem as finding the roots of a vector of high order polynomials generated from $P(\mathbf{y}) = \mathcal{L}\mathbf{x}^{k-1} - \lambda\mathbf{x} = \mathbf{0}$, where $\mathbf{y} = [\mathbf{x} \quad \lambda] \in \mathbb{R}^{n+1}$. As it is tough to compute the zeros of $P(\mathbf{y})$ directly, the core idea of linear homotopy methods is to construct a vector function $H(\mathbf{y}, t) = (1 - t)Q(\mathbf{y}) + tP(\mathbf{y})$, where $t \in [0, 1]$ and $Q(\mathbf{y})$ is a suitable vector polynomial whose roots can be computed easily. The next step is to slowly iterate from the solution of $H(\mathbf{y}, t = 0) = Q(\mathbf{y}) = \mathbf{0}$ to $H(\mathbf{y}, t = 1) = P(\mathbf{y}) = \mathbf{0}$. Despite the novel formulation, this approach is forced to compute all the complex eigenpairs even if we are interested in real eigenpairs only.

Before proceeding to the main discussion on the computation of the Fiedler vector, it should be noted that one of the eigenvectors for minimum eigenvalue can be found analytically by exploiting the particular structure of Laplacian tensor [32]. In fact, the minimum eigenvalue of Laplacian tensor is known to be 0, and the corresponding eigenvector is $\mathbf{x} = \frac{1}{\sqrt{n}}[1 \quad 1 \cdots 1]$. There can be other eigenvectors for the zero eigenvalue whose graphical implication is discussed in the literature [23]. For our problem, we may use the approach by Cui et al. [49], which computes all the real eigenvalues sequentially from maximum to minimum by using Jacobian semidefinite relaxations in polynomial optimization to avoid computing all eigenvalues.

They formulate the following problem to compute λ_{i+1} assuming λ_i is known:

$$\begin{aligned} \max \quad & f(\mathbf{x}) = \mathcal{L}\mathbf{x}^k \\ \text{such that} \quad & f(\mathbf{x}) \leq \lambda_k - \delta \quad \& \quad h_r(\mathbf{x}) = 0, \quad (r = 1, \ldots, 2n-2) \end{aligned} \tag{31}$$

where $0 < \delta < \lambda_i - \lambda_{i+1}$ and $h_r(\mathbf{x})$ is defined as:

$$h_r(\mathbf{x}) = \sum_{i+j=r+2} \frac{\partial f(\mathbf{x})}{\partial x_i}\frac{\partial g(\mathbf{x})}{\partial x_j} - \frac{\partial f(\mathbf{x})}{\partial x_j}\frac{\partial g(\mathbf{x})}{\partial x_i} \tag{32}$$

where $g(\mathbf{x}) = x_1^2 + x_2^2 + \ldots + x_n^2 - 1$ is a normalization constraint. They further utilize Lasserre's hierarchy of semidefinite relaxations [50] to solve the above problem.

The computation of the objective function $f(\mathbf{x})$ and the constraints $h_r(\mathbf{x})$ is expensive and takes $O(n^k)$ for general tensors. Using Theorem 2, the objective function can be computed in linear time $O(m)$ for Laplacian tensors. The constraint can also be simplified using:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \sum_{e_p \in E_i} k w_{e_p} \left(x_i^{k-1} - k \prod_{t \in \{e_p \setminus i\}} x_t \right) \tag{33}$$

where $E_i = \{e_q | i \cap e_q \neq \emptyset, e_q \in E\}$. This approach is very helpful as all the eigenvalues need not be computed for the Fiedler eigenvalue. Hence, these closed form expression for the case of Laplacian tensor can be utilized to reduce the number of function evaluations in optimization methods as compared to general tensors.

3.5 Related works

As stated earlier, most of the existing methods utilize hypergraph reductions either implicitly [13, 15] or explicitly or coarsening [10] or scalable heuristic methods [19]. For example, Ghoshdastidar et al. [51] utilize the tensor-based representation of hypergraphs but construct a matrix by concatenating the slices of the tensor. Further, they apply the standard spectral partitioning algorithm on the covariance of that matrix. These variants of hypergraph reduction differ in the method of expanding a hyperedge and produce graphs with different edge weights. The Laplacian objective function (Eq (7)) of any graph is second-order polynomial, which captures weighted interaction among two nodes. A second-order polynomial is insufficient for capturing super-dyadic interaction among multiple nodes ($\geq$3) of a hyperedge. Also, note that multiple hypergraphs may reduce to the same graph.

Hein et al. [52] discuss the incapability of reduction methods in preserving the hyperedge cuts for general hypergraphs. We utilize the Laplacian tensor (Eq (7)) to penalize these multiple cuts differently. Few other recent works try to capture these multiple ways of splitting nodes. For example, Li et al. [53] proposes non-uniform clique expansion and provides quadratic approximation under submodularity constraints of the inhomogeneous cost function. Li et al. [28] extends the notion of p-Laplacian from graphs to hypergraphs by introducing the following hyperedge score:

$$l_{e_j}(\mathbf{x}) = \max_{i_k, i'_k \in e_j} |x_{i_k} - x_{i'_k}|^p \tag{34}$$

Ideally, any definition of hyperedge score should capture the non-uniformity among the nodes in a hyperedge, but the above equation fails to capture the variation perfectly. For example, consider two hyperedges with cardinality 4 and node labels assigned as {0, 1, 1, 2} and {0, 1, 2, 2}. Eq (34) computes the maximum difference and hence will not differentiate among these two hyperedges but the AM-GM difference (Eq (24)) will capture the variation among all the nodes of the hyperedge. Various other similar formulation of the hyperedge score function are considered in literature [54–56].

4 Experiments

We compare the proposed algorithm and sign-based Fiedler vector partitioning on cockroach graphs. Further, the proposed algorithm is examined on synthetic graphs and hypergraphs generated by Erdős Rényi Model [57] and Stochastic Block Model (SBM) [58]. The numerical details of Fiedler vector and hyperedge scores are presented in S1 File.

4.1 Proposed algorithm vs sign-based partitioning on cockroach graph

Consider the cockroach graph with $4t$ nodes, as shown in Fig 3. Von Luxburg [26] shows that the conventional sign-based Fiedler vector partitioning does not produce the optimal ratio-cut for cockroach graph. In this example, we show that the proposed algorithm performs better. For comparing the proposed method and sign-based Fiedler vector partitioning, we compute the ratio-cut value by these algorithms.

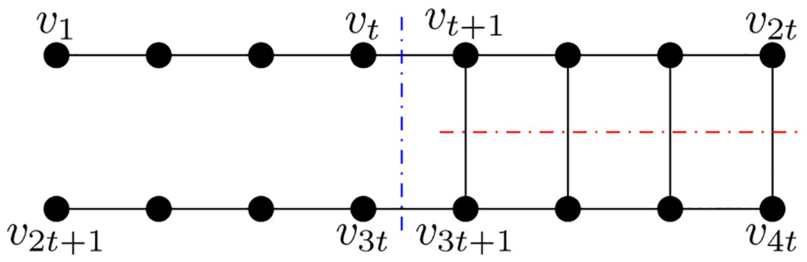


Fig 3. Cockroach graph.

https://doi.org/10.1371/journal.pone.0288457.g003

Table 1. Edge-score for graph in Example 4.1.

Edge	Score
$\{v_3, v_4\}$	0.0371
$\{v_9, v_{10}\}$	0.0371
$\{v_4, v_{10}\}$	0.0228
$\{v_2, v_3\}$	0.0228
$\{v_8, v_9\}$	0.0222
$\{v_1, v_2\}$	0.0222
$\{v_7, v_8\}$	0.0066
$\{v_4, v_5\}$	0.0066
$\{v_{10}, v_{11}\}$	0.0029
$\{v_5, v_{11}\}$	0.0029
$\{v_6, v_{12}\}$	0.0002
$\{v_{11}, v_{12}\}$	0.0002
$\{v_5, v_6\}$	0.0002

https://doi.org/10.1371/journal.pone.0288457.t001

The analysis shown in this example is valid for general t but we have presented the numerical details for $t = 3$. The Fiedler vector for this graph ($t = 3$) is given by:

$$\mathbf{f} = \begin{bmatrix} -0.49 & -0.41 & -0.26 & -0.07 & -0.02 & -0.01 & 0.49 & 0.41 & 0.26 & 0.07 & 0.02 & 0.01 \end{bmatrix}^T \quad (35)$$

So the partitions defined based on the sign of elements in $\mathbf{v}$ are $A_1 = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $A_2 = \bar{A_1}$. So, the ratioCut $(A_1, \bar{A_1}) = \frac{3}{6} + \frac{3}{6} = 1$.

The next step is to apply the proposed algorithm. The edge score computed from the proposed algorithm are presented in Table 1. The edges $\{v_3, v_4\}$ and $\{v_9, v_{10}\}$ are removed as they are of maximum edge score of 0.0371. So the partitions are $B_1 = \{v_1, v_2, v_3, v_7, v_8, v_9\}, B_2 = \bar{B_1}$. Therefore, ratioCut $(B_1, \bar{B_1}) = \frac{2}{6} + \frac{2}{6} = 0.66$. It can be clearly observed that the partitions obtained from the proposed algorithm have a lower ratioCut value compared to the existing method.

In general, the traditional spectral partitioning makes the red cut shown in the graph and the partition is $A_1 = \{v_1, \ldots, v_{2t}\}$ and the ratio-cut $(A_1, \bar{A_1}) = \frac{t}{2t} + \frac{t}{2t} = 1$. We utilize the edge scores as suggested in the proposed algorithm and report that the edges $\{v_t, v_{t+1}\}$ and $\{v_{3t}, v_{3t+1}\}$ have maximum scores. On cutting these edges, the obtained partition is $B_1 = \{v_1, v_2, \ldots, v_t, v_{2t+1}, \ldots, v_{3t}\}$ and hence the ratio-cut $(B_1, \bar{B_1}) = \frac{2}{2t} + \frac{2}{2t} = \frac{2}{t}$. Therefore, the solution obtained by proposed algorithm is $t/2$ times better than the traditional approach. We have verified it numerically for $t = \{3, 4, \ldots, 50\}$.

4.2 Proposed algorithm vs sign-based partitioning on synthetic graphs & hypergraphs

In this example, we consider different types of synthetic graphs and compare the ratio-cut values computed by the existing and proposed methods. We define the following metric, termed as percentage improvement (PI) to showcase the proposed algorithm's performance:

$$\mathrm{PI} = \frac{(R_f - R_p)}{R_f} \times 100 \quad (36)$$

where R_f, R_p denotes the ratio-cut value by sign based Fiedler partitioning and proposed algorithm, respectively. A positive value of PI indicates the proposed algorithm has produced a better ratio-cut value and the magnitude of the value represents the extent of the improvement.

4.2.1 Proposed algorithm vs sign-based partitioning on graphs generated by ER model. We begin with the study on random graphs generated from the Erdős Rényi Model [57] denoted by $G(n, p)$, where n is the number of nodes and p is the probability of an edge between any two nodes. We compare the ratio-cut values 2 partitions values on 100 different graphs for $n = 100$ and for each value of $p = \{0.2, 0.4, 0.6\}$. Fig 4 shows the result as a histogram for different values of $p = \{0.2, 0.4, 0.6\}$. It can be seen that the proposed algorithm performs better than the sign based Fiedler partitioning in all cases.

4.2.2 Proposed algorithm vs sign-based partitioning on graphs generated by SBM. We perform a similar analysis for another graph generation model, referred to as the stochastic

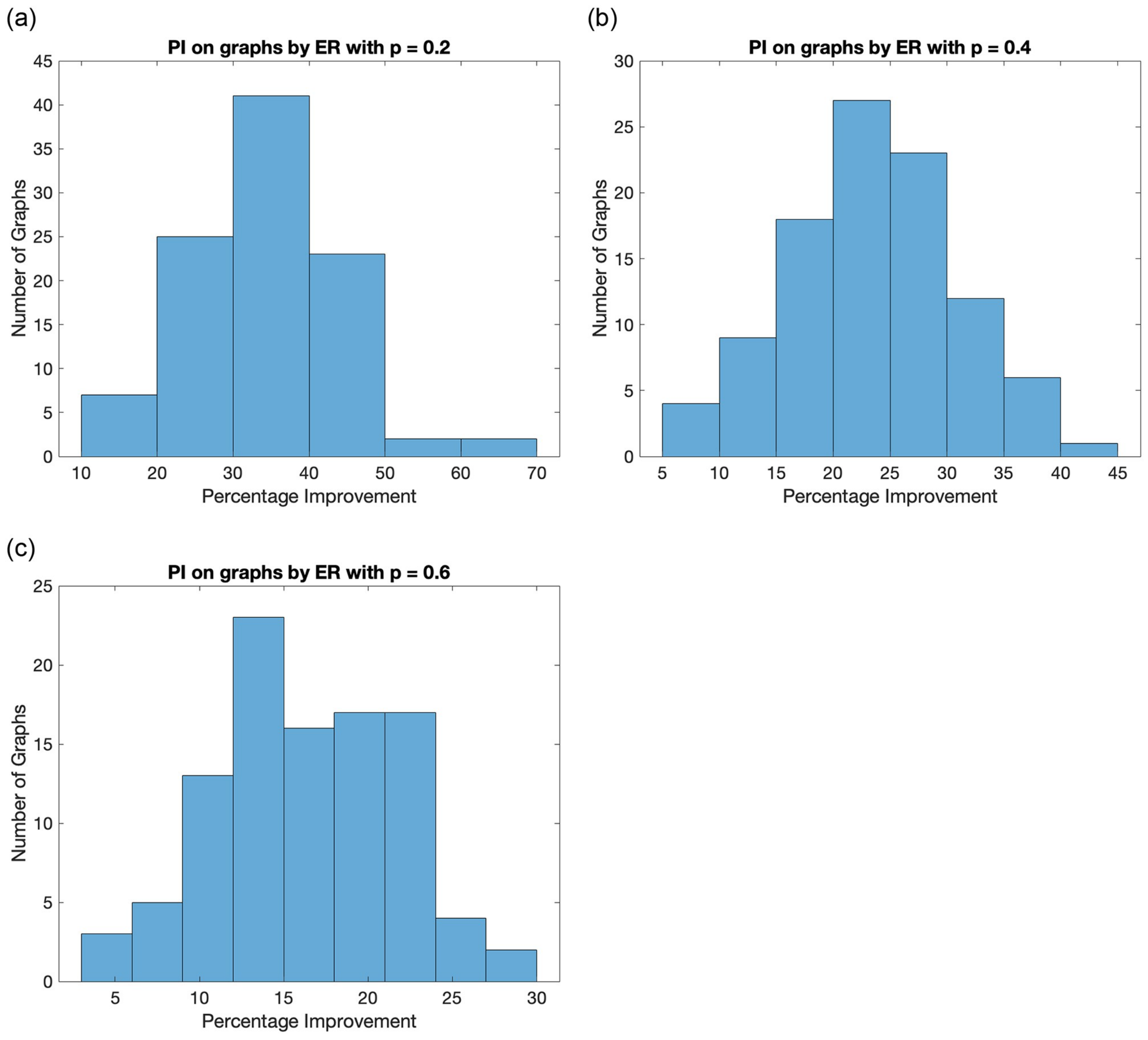


Fig 4. Histogram plot for percentage improvement on ratio-cut value by the proposed method for graphs generated by the ER model for different values of *p*. It shows that the proposed algorithm performs better for all the generated graphs.

https://doi.org/10.1371/journal.pone.0288457.g004

block model (SBM). This model provides us the freedom to control the number of parts, the number of nodes in each part (denoted by n_1, n_2), the probability of an edge within a part (denoted by p), and across the partition (q). Note that $p = q$ yields ER model with $n = n_1 + n_2$ as discussed previously.

We consider the graphs for multiple combinations of probabilities p, q and 2 partitions with $n_1 = n_2 = 50$. It should be noted that we consider the SBM with assortative community structure, which implies $p > q$. We generate 100 random graphs for each of these settings and compare the ratio-cut values. A histogram plot summarizing the results is presented in Fig 5.

It is evident from Fig 5 that the proposed algorithm produces a lower ratio-cut value for most of the graphs generated by SBM.

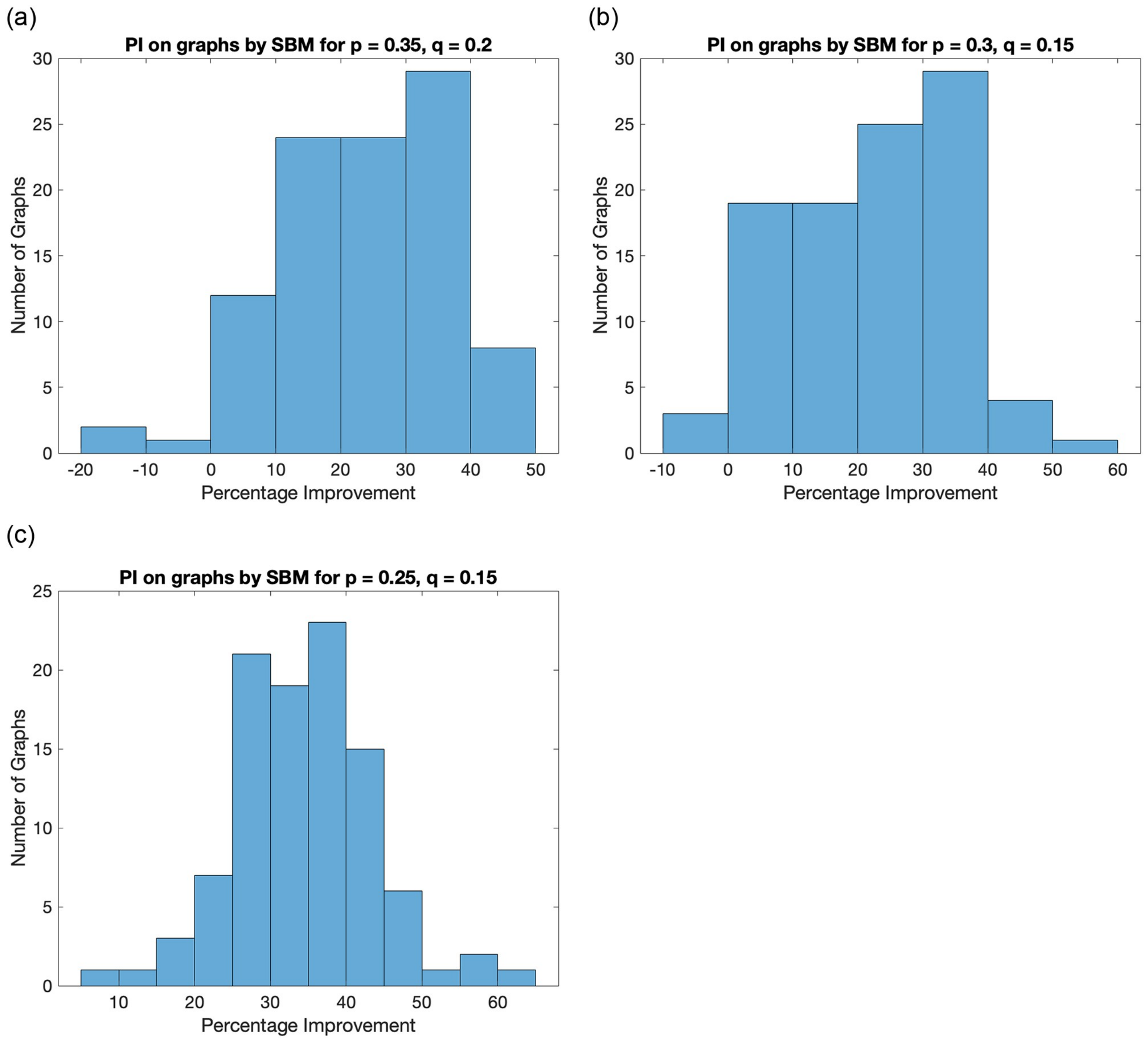


Fig 5. Histogram plot for percentage improvement on ratio-cut value by the proposed method for graphs generated by the SBM for different values of intra-cluster probability (p) and inter-cluster probability (q). It confirms that the proposed algorithm performs better for most of the generated graphs as there are very few cases of negative PI.

https://doi.org/10.1371/journal.pone.0288457.g005

4.2.3 Proposed algorithm vs sign-based partitioning on hypergraphs generated by SBM. We perform a similar analysis on synthetic hypergraphs generated by SBM [51]. We generate 100 random 4-uniform hypergraphs with 2 partitions, 60 nodes, and relatively small values of intra-cluster probability (p) and inter-cluster probability (q) as compared to the case of graphs. This is primarily because the number of possible hyperedges for a 4–uniform hypergraph is $\binom{n}{4}$, which is much larger than $\binom{n}{2}$ as compared to the case of graphs.

The proposed algorithm is compared to the conventional sign-based partitioning using the Fiedler vector computed from the Laplacian tensor of the hypergraph. It should be noted that computation of tensor eigenvalues in NP-hard and cannot be approximated unless P = NP [22]. A histogram plot summarizing the results is shown in Fig 6.

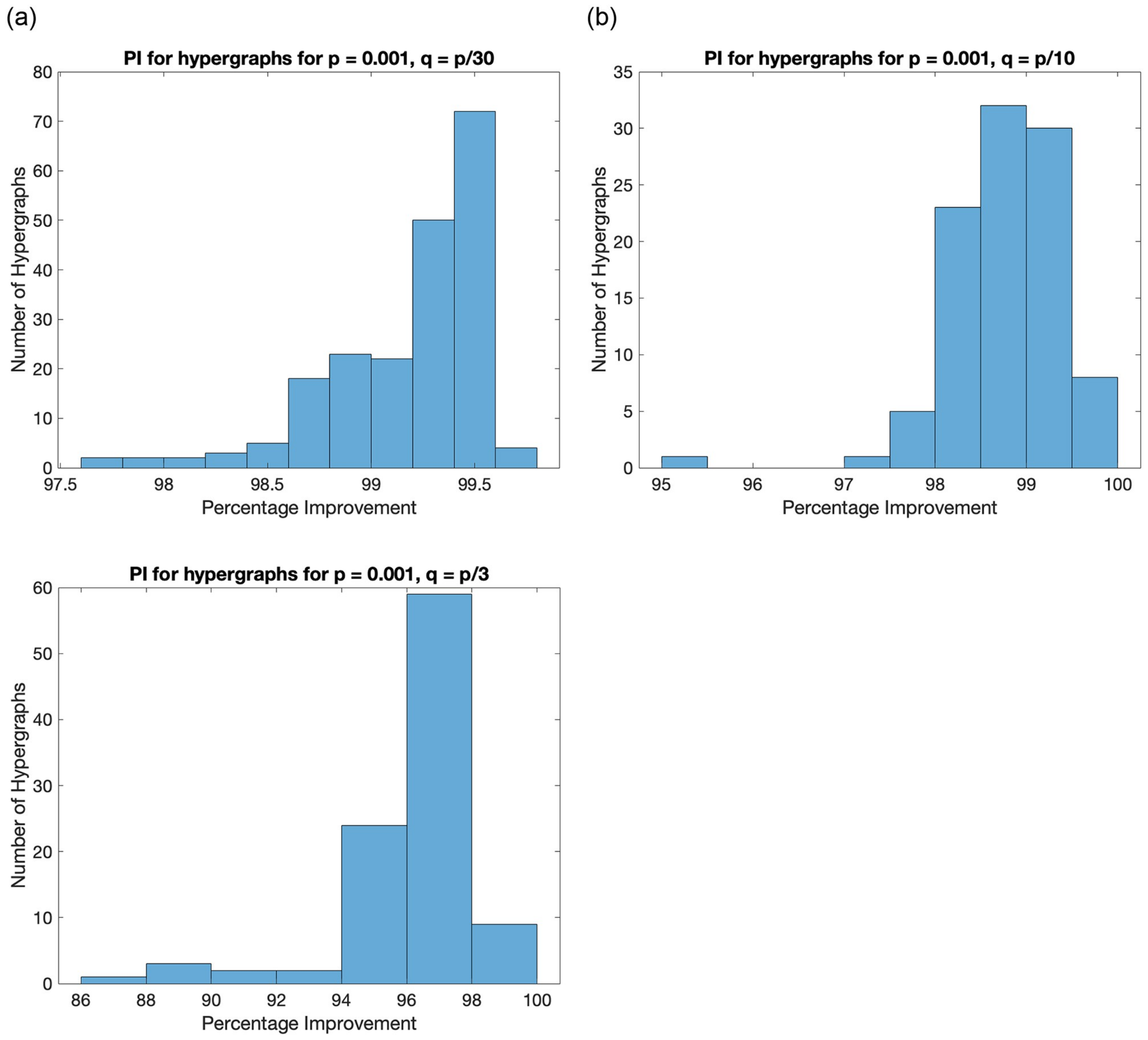
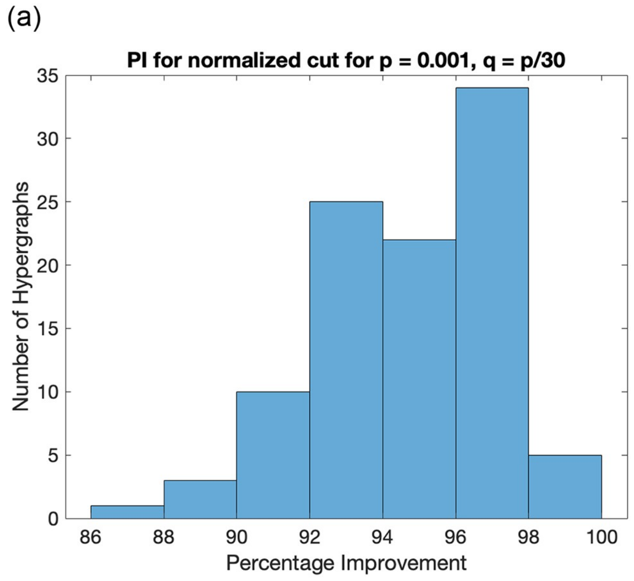


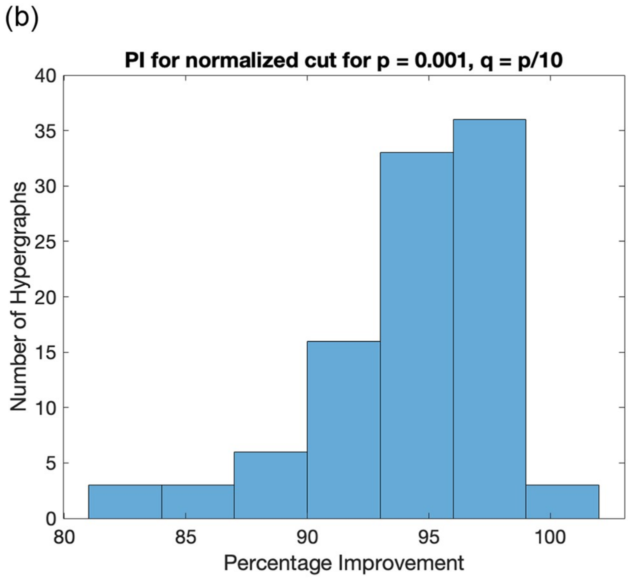
Fig 6. Histogram plot for percentage improvement on ratio cut value by the proposed method for hypergraphs generated by the SBM for different values of *q*. It shows that the proposed algorithm performs significantly better as compared to sign-based partitioning for all generated hypergraphs.

https://doi.org/10.1371/journal.pone.0288457.g006

We perform a similar analysis on the comparison of proposed algorithm and sign-based partitioning using Fiedler vector of normalized Laplacian tensor of the hypergraph. This is done to study the behaviour of algorithm for normalized cut defined in Eq (16). The histogram plot is presented in Fig 7.

It can be observed that the proposed algorithm has improved the ratio-cut value (defined in Eq (15)) and normalized cut value (defined in Eq (16)) significantly as compared to the traditional sign-based partitioning. This is primarily because cutting a few hyperedges does not necessarily produce only two components, unlike the case of graphs. For example, if we cut a hyperedge having 3 nodes in a hypergraph with one hyperedge only, we get 3 disconnected





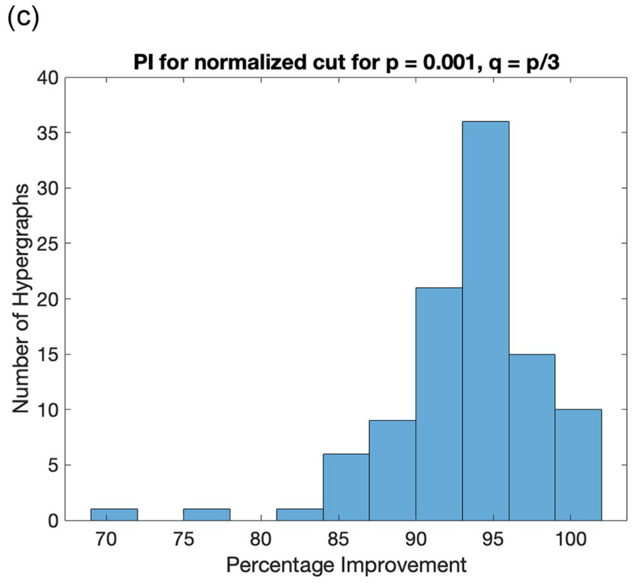


Fig 7. Histogram plot for percentage improvement by the proposed method for normalized cut value on hypergraphs generated by the SBM for different values of *q*. It shows that the proposed algorithm performs significantly better as compared to sign-based partitioning for all generated hypergraphs.

https://doi.org/10.1371/journal.pone.0288457.g007

components, and there is no possibility of obtaining two connected components. Hence, we may get 3 connected components, even if we desired only 2 connected components.

Any partitioning algorithm producing many small connected components (like singletons) is likely to have a higher ratio-cut value. We observe that the sign-based partitioning approach using the Fiedler vector of the Laplacian tensor is more prone to producing many small connected components as compared to the results by proposed algorithms. Hence, the ratio-cut value or normalized cut value by sign-based partitioning is significantly higher.

5 Conclusions & future work

In this work, we propose a hypergraph partitioning algorithm using tensor eigenvalue framework which removes the hyperedges directly without performing reduction to a graph like existing methods. This was done by using the novel "hyperedge score" metric. To do this, we extended the definition of ratio-cut and normalized cut from graphs to hypergraph and showed the equivalence of relaxed optimization problem to a tensor eigenvalue problem. Further, we derived a tighter upper bound for the approximation of normalized-cut problem. The future directions of this work is along the lines of similar analysis for non-uniform and directed hypergraphs.

Supporting information

S1 File.
(ZIP)

Author Contributions

Conceptualization: Deepak Maurya, Balaraman Ravindran.

Data curation: Deepak Maurya.

Formal analysis: Deepak Maurya.

Funding acquisition: Balaraman Ravindran.

Methodology: Deepak Maurya, Balaraman Ravindran.

Project administration: Balaraman Ravindran.

Resources: Balaraman Ravindran.

Supervision: Balaraman Ravindran.

Validation: Deepak Maurya, Balaraman Ravindran.

Writing – original draft: Deepak Maurya.

Writing – review & editing: Balaraman Ravindran.

References

1. Aittokallio T, Schwikowski B. Graph-based methods for analysing networks in cell biology. Briefings in bioinformatics. 2006; 7(3):243–255. https://doi.org/10.1093/bib/bbl022 PMID: 16880171
2. Bhatt SN, Leighton FT. A framework for solving VLSI graph layout problems. Journal of Computer and System Sciences. 1984; 28(2):300–343. https://doi.org/10.1016/0022-0000(84)90071-0
3. Veksler O. Star shape prior for graph-cut image segmentation. In: European Conference on Computer Vision. Springer; 2008. p. 454–467.
4. Gao J, Buldyrev SV, Stanley HE, Havlin S. Networks formed from interdependent networks. Nature physics. 2012; 8(1):40.

5. Dhillon IS, Guan Y, Kulis B. Kernel k-means: spectral clustering and normalized cuts. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM; 2004. p. 551–556.

6. Wang M, Fu W, Hao S, Tao D, Wu X. Scalable semi-supervised learning by efficient anchor graph regularization. IEEE Transactions on Knowledge and Data Engineering. 2016; 28(7):1864–1877. https://doi.org/10.1109/TKDE.2016.2535367

7. Chung FR, Graham FC. Spectral graph theory. 92. American Mathematical Soc.; 1997.

8. Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems. 2016; 29.

9. Zhou D, Huang J, Scholkopf B. Beyond pairwise classification and clustering using hypergraphs. In: Proceedings of the Neural Information Processing Systems; 2005.

10. Karypis G, Aggarwal R, Kumar V, Shekhar S. Multilevel hypergraph partitioning: applications in VLSI domain. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 1999; 7(1):69–79. https://doi.org/10.1109/92.748202

11. Agarwal S, Lim J, Zelnik-Manor L, Perona P, Kriegman D, Belongie S; IEEE. Beyond pairwise clustering. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). 2005;2:838–845.

12. Huang Y, Liu Q, Metaxas D. Video object segmentation by hypergraph cut. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE; 2009. p. 1738–1745.

13. Zhou D, Huang J, Schölkopf B. Learning with hypergraphs: Clustering, classification, and embedding. In: Advances in neural information processing systems; 2007. p. 1601–1608.

14. Yadati N, Nimishakavi M, Yadav P, Nitin V, Louis A, Talukdar P. HyperGCN: A new method of training graph convolutional networks on hypergraphs. arXiv preprint arXiv:180902589. 2018;.

15. Agarwal S, Branson K, Belongie S. Higher order learning with graphs. In: Proceedings of the 23rd International Conference on Machine learning. ACM; 2006. p. 17–24.

16. Kumar T, Darwin K, Parthasarathy S, Ravindran B. HPRA: Hyperedge prediction using resource allocation. In: 12th ACM conference on web science; 2020. p. 135–143.

17. Li L, Li T. News recommendation via hypergraph learning: encapsulation of user behavior and news content. In: Proceedings of the sixth ACM international conference on Web search and data mining; 2013. p. 305–314.

18. Gao Y, Wang M, Tao D, Ji R, Dai Q. 3-D object retrieval and recognition with hypergraph analysis. IEEE Transactions on Image Processing. 2012; 21(9):4290–4303. https://doi.org/10.1109/TIP.2012.2199502 PMID: 22614650

19. Veldt N, Benson AR, Kleinberg J. Minimizing localized ratio cut objectives in hypergraphs. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2020. p. 1708–1718.

20. Ghoshdastidar D, Dukkipati A. A provable generalized tensor spectral method for uniform hypergraph partitioning. In: International Conference on Machine Learning; 2015. p. 400–409.

21. Ihler E, Wagner D, Wagner F. Modeling hypergraphs by graphs with the same mincut properties. Information Processing Letters. 1993; 45(4):171–175. https://doi.org/10.1016/0020-0190(93)90115-P

22. Hillar CJ, Lim LH. Most tensor problems are NP-hard. Journal of the ACM (JACM). 2013; 60(6):1–39. https://doi.org/10.1145/2512329

23. Hu S, Qi L. The eigenvectors associated with the zero eigenvalues of the Laplacian and signless Laplacian tensors of a uniform hypergraph. Discrete Applied Mathematics. 2014; 169:140–151. https://doi.org/10.1016/j.dam.2013.12.024

24. Shashua A, Zass R, Hazan T. Multi-way clustering using super-symmetric non-negative tensor factorization. In: European conference on computer vision. Springer; 2006. p. 595–608.

25. Benson AR. Three hypergraph eigenvector centralities. SIAM Journal on Mathematics of Data Science. 2019; 1(2):293–312. https://doi.org/10.1137/18M1203031

26. Von Luxburg U. A tutorial on spectral clustering. Statistics and computing. 2007; 17(4):395–416. https://doi.org/10.1007/s11222-007-9033-z

27. Chen Y, Qi L, Zhang X. The Fiedler Vector of a Laplacian Tensor for Hypergraph Partitioning. SIAM Journal on Scientific Computing. 2017; 39(6):A2508–A2537. https://doi.org/10.1137/16M1094828

28. Li P, Milenkovic O. Submodular Hypergraphs: p-Laplacians, Cheeger Inequalities and Spectral Clustering. In: Dy J, Krause A, editors. Proceedings of the 35th International Conference on Machine Learning. vol. 80 of Proceedings of Machine Learning Research. PMLR; 2018. p. 3014–3023.

29. Zien JY, Schlag MD, Chan PK. Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. IEEE Transactions on computer-aided design of integrated circuits and systems. 1999; 18(9):1389–1399. https://doi.org/10.1109/43.784130

30. Kumar T, Vaidyanathan S, Ananthapadmanabhan H, Parthasarathy S, Ravindran B. Hypergraph Clustering: A Modularity Maximization Approach. arXiv preprint arXiv:181210869. 2018;.

31. Qi L, Luo Z. Tensor analysis: spectral theory and special tensors. vol. 151. SIAM; 2017.

32. Banerjee A, Char A, Mondal B. Spectra of general hypergraphs. Linear Algebra and its Applications. 2017; 518:14–30. https://doi.org/10.1016/j.laa.2016.12.022

33. Hagen L, Kahng AB. New spectral methods for ratio cut partitioning and clustering. IEEE transactions on computer-aided design of integrated circuits and systems. 1992; 11(9):1074–1085. https://doi.org/10.1109/43.159993

34. Shi J, Malik J. Normalized cuts and image segmentation. Departmental Papers (CIS). 2000; p. 107.

35. Chung F. Four proofs for the Cheeger inequality and graph partition algorithms. Proceedings of ICCM. 2007; 2:378.

36. Bühler T, Hein M. Spectral clustering based on the graph p-Laplacian. In: Proceedings of the 26th Annual International Conference on Machine Learning; 2009. p. 81–88.

37. Lee JR, Gharan SO, Trevisan L. Multiway spectral partitioning and higher-order cheeger inequalities. Journal of the ACM (JACM). 2014; 61(6):1–30. https://doi.org/10.1145/2665063

38. Chandrasekaran K, Xu C, Yu X. Hypergraph k-cut in randomized polynomial time. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics; 2018. p. 1426–1438.

39. Chekuri C, Li S. A note on the hardness of approximating the k-way Hypergraph Cut problem. Manuscript, http://chekuri.cs.illinois.edu/papers/hypergraph-kcut.pdf. 2015;.

40. Ghoshdastidar D, Dukkipati A. Spectral Clustering Using Multilinear SVD: Analysis, Approximations and Applications. In: AAAI; 2015. p. 2610–2616.

41. Chekuri C, Li S. On the Hardness of Approximating the k-WAY HYPERGRAPH CUT Problem. Theory of Computing. 2020; 16(1).

42. Chandrasekaran K, Chekuri C. Min-max partitioning of hypergraphs and symmetric submodular functions. In: Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA). SIAM; 2021. p. 1026–1038.

43. Goldschmidt O, Hochbaum DS. A polynomial algorithm for the k-cut problem for fixed k. Mathematics of operations research. 1994; 19(1):24–37. https://doi.org/10.1287/moor.19.1.24

44. Qi L. Eigenvalues of a real supersymmetric tensor. Journal of Symbolic Computation. 2005; 40 (6):1302–1324. https://doi.org/10.1016/j.jsc.2005.05.007

45. Qi L, Wang F, Wang Y. Z-eigenvalue methods for a global polynomial optimization problem. Mathematical Programming. 2009; 118(2):301–316. https://doi.org/10.1007/s10107-007-0193-6

46. Kolda TG, Mayo JR. Shifted power method for computing tensor eigenpairs. SIAM Journal on Matrix Analysis and Applications. 2011; 32(4):1095–1124. https://doi.org/10.1137/100801482

47. Hu S, Huang ZH, Qi L. Finding the extreme Z-eigenvalues of tensors via a sequential semidefinite programming method. Numerical Linear Algebra with Applications. 2013; 20(6):972–984. https://doi.org/10.1002/nla.1884

48. Chen L, Han L, Zhou L. Computing tensor eigenvalues via homotopy methods. SIAM Journal on Matrix Analysis and Applications. 2016; 37(1):290–319. https://doi.org/10.1137/15M1010725

49. Cui CF, Dai YH, Nie J. All real eigenvalues of symmetric tensors. SIAM Journal on Matrix Analysis and Applications. 2014; 35(4):1582–1601. https://doi.org/10.1137/140962292

50. Lasserre JB. Global optimization with polynomials and the problem of moments. SIAM Journal on optimization. 2001; 11(3):796–817. https://doi.org/10.1137/S1052623400366802

51. Ghoshdastidar D, Dukkipati A. Consistency of spectral partitioning of uniform hypergraphs under planted partition model. In: Advances in Neural Information Processing Systems; 2014. p. 397–405.

52. Hein M, Setzer S, Jost L, Rangapuram SS. The total variation on hypergraphs-learning on hypergraphs revisited. In: Advances in Neural Information Processing Systems; 2013. p. 2427–2435.

53. Li P, Milenkovic O. Inhomogeneous hypergraph clustering with applications. In: Advances in Neural Information Processing Systems; 2017. p. 2308–2318.

54. Zhang C, Hu S, Tang ZG, Chan T. Re-revisiting learning on hypergraphs: confidence interval and subgradient method. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org; 2017. p. 4026–4034.

55. Chan THH, Louis A, Tang ZG, Zhang C. Spectral properties of hypergraph laplacian and approximation algorithms. Journal of the ACM (JACM). 2018; 65(3):1–48. https://doi.org/10.1145/3178123

56. Chan THH, Liang Z. Generalizing the hypergraph laplacian via a diffusion process with mediators. Theoretical Computer Science. 2020; 806:416–428. https://doi.org/10.1016/j.tcs.2019.07.024

57. Erdős P, Rényi A, et al. On the evolution of random graphs. Publ Math Inst Hung Acad Sci. 1960; 5 (1):17–60.

58. Holland PW, Laskey KB, Leinhardt S. Stochastic blockmodels: First steps. Social networks. 1983; 5 (2):109–137. https://doi.org/10.1016/0378-8733(83)90021-7