Hypergraphs projection methods for community detections

Rattana Pukdee

This manuscript was compiled on October 27, 2020

Community detection is one of the first thing people do when they start working with a dataset as it gives an overview of the dataset and therefore is an important task in networks. Hypergraph is an extension of graph to capture higher order interactions and there are works in community detection algorithms for hypergraphs that extends ideas from graphs. However, (1) shows that an algorithm on hypergraph such as hypergraph laplacian (2) is equivalent to an existing algorithm in graphs applied to a graph projection of a hypergraph. The main goal of this report is to explore whether such graph projection works well in general with other algorithm for graphs. We will test with different approach such as modularity or spectral clustering on recently developed models for hypergraphs as well as on real world data.

Graph is a model for pairwise interactions in a system and is arguably one of the most important subject in the scientific world. There has been a large literature in many tasks such as community detection (3), centrality (4), epidemic process modelling (5) and real world application range from a citation network (6) to analysis of football players (7). Hypergraph is an extension of graph to capture higher order interactions and it's still an active field in research. It makes sense to study hypergraphs as there are many high order interactions in the nature. For example, in a social network, when a group of friends hang out together or when a group of proteins interact with each other. There has been some work done on extending existing ideas from graphs to hypergraphs such as hypergraph laplacian (2), hypergraph modularity (8), hypergraph centrality (9) and also works on projecting hypergraphs to graphs and apply existing tools on graphs (10). In this report, we want to focus on community detection in hypergraphs. As mention earlier, one idea is to project a hypergraph into a graph. Examples include the academic collaboration networks (11), (12), (13), which are converted from coauthorship hypergraph networks (14). Another idea is to use hypergraph laplacian (2), (15) which extend the idea of normalized laplacian clustering(16),(17) to hypergraphs in the analogue of cut problem. There is also an idea that works with tensor which can be seen as a higher dimension array that represent a hypergraph (18), (19), (20), (14) or modularity maximisation (21).

In term of theoretical guarantee, there are only few results out there, compared to the number of algorithms such as (20) for tensor method or (15) for hypergraph laplacian. (1) show that the hypergraph laplacian is ,in fact, equivalent to a normalized laplacian clustering (16), (22) on a certain graph representation of the hypergraph. With the theoretical guarantee from (15), we realise that analysing a hypergraph using its graph representation could also give a promising result. This raises a question that the author would like to explore more in this report that whether the graph representation of a hypergraph works well in general with other algorithm for graphs. We will test this hypothesis by applying different graph community detection algorithms on both synthesized hypergraphs generated from recent hypergraph models (15), (14) and on real world data.

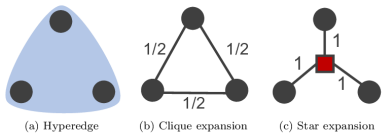

Fig. 1. Techniques for converting a hyperedge into a set of edges (23)

Preliminaries. In this section, we first introduce relevant definitions from (2),(15). For simplicity, we will consider only unweighted hypergraphs in this report. Let V denote a finite set of objects, and let E be a family of subsets e of V such that $\cup_{e\in E} = V$. We call $G = (V, E)$ a *hypergraph* with the *vertex set* V and the *hyperedge* set E. A hypergraph G can be represented by a $|V| \times |E|$ matrix H with entries $H_{ve} = 1$ if the node v is contained in the edge e, and 0 otherwise, called the *incidence matrix* of G. For a vertex $v \in V$, its *degree* is defined by $\deg(v) = \sum_{e\in E} H_{ve}$, number of edges that contain that vertex. The cardinality of an edge $e \in E$ is $|e| = \sum_{v\in V} H_{ve}$, number of vertices in that edges. A *m-uniform hypergraph* is a hypergrah that all of its hyperedges have cardinality m. Next, we introduce two popular hypergraphs projection methods (1),(10).

Clique Expansion. Construct a graph $G_x(V, E_x)$ from the original hypergraph $G(V, E)$ by replacing each hyperedge e with a clique : $E_x = \{(u, v) : u, v \in e, e \in E\}$.

Star Expansion. Constructs a graph $G^*(V^*, E^*)$ from the original hypergraph $G(V, E)$ by introducing a new vertex for every hyperedge $e \in E$, thus $V = V \cup E$. Each new vertex e connects to vertices in its corresponding hyperedge. That is $E^* = \{(u, e) : u \in e, e \in E\}$.

The visualisation is illustrated in figure 1. We can see that a triangle clique expansion represents both when a hyperedge with size three or three hyperedges with size two and this may

Significance Statement

Hypergraphs projection to graphs is a simple yet effective way to visualise higher order interaction in hypergraphs and it also provides access to existing tools in graphs for analysing hypergraphs. A good understanding of such method could benefit both researchers who work directly with hypergraphs and those who work with graph

lead to loss of information when we work with the projected graph. (2) avoid working with graph representations of hypergraphs and instead try to work directly with the hypergraphs by generalising the normalized cut approach of (17). We will introduce definitions and discuss about this in the next section.

Normalized hypergraph cut. Let $V_1 \subset V$, then $vol(V_1) = \sum_{v \in V_1} \deg(v)$ is called the *volume* of V_1. Let V_1^c be the complement of V_1, we say that an edge e is a *cut* if it contains vertices in V_1 and V_1^c simultaneously. The *boundary* of V_1 is defined as $\partial V_1 = \{e \in E : e \cap V_1 \neq \emptyset, e \cap V_1^c \neq \emptyset\}$, the set of edges that are cut. The *volume* of ∂V_1 is defined as

$$\mathrm{vol}(V_1) = \sum_{e \in \partial V_1} \frac{|e \cap V_1||e \cap V_1^c|}{|e|}$$

is number of subedges which are cut. In a cut problem, we want to obtain a partition which has a dense connections in the same cluster and sparse connection between two clusters. The volume ∂V_1 measures how strong the connection between V_1 and V_1^c in some sense. We consider a problem of partitioning the vertex set V into k disjoint sets, $V_1, \ldots, V_k$ that minimises the *normalized hypergraph cut*

$$\text{NH-Cut}(V_1, \ldots, V_k) = \sum_{j=1}^{k} \frac{\mathrm{vol}\partial V_j}{\mathrm{vol} V_j} \qquad [1]$$

According to (22), the denominator terms $\mathrm{vol}V_j$, help regulate the solution to prefer more "balance" clusters (with similar volume). We define the *normalized hypergraph laplacian matrix* $L \in \mathbb{R}^{|V| \times |V|}$ as follows

$$L = I - D^{-1/2} H \triangle^{-1} H^T D^{-1/2} \qquad [2]$$

when $D \in \mathbb{R}^{|V| \times |V|}, \triangle \in \mathbb{R}^{|E| \times |E|}$ are diagonal matrix with $D_{vv} = \deg(v)$ and $\triangle_{ee} = |e|$. With a direct calculation, the problem of minimising [1] is equivalent to the problem:

$$\text{minimize}_{V_1, \ldots, V_k} \mathrm{Trace}(\hat{X}^T L \hat{X}) \qquad [3]$$

where $\hat{X} \in \mathbb{R}^{|V| \times k}$ such that $\hat{X}_{vj} = \sqrt{\frac{\deg(v)}{\mathrm{vol}V_j}} \mathbb{1}\{v \in V_j\}$ satisfies $\hat{X}^T \hat{X} = I$. Intuitively, $\hat{X}$ is a clustering assignment matrix where entries for each row are zero apart from the entry which correspond to the vertex's cluster. The problem [3] is NP-hard and we relax constraints to only minimising over $\hat{X}^T \hat{X} = I$. The solution of this relaxed problem is given by the matrix U which columns are the first k eigenvectors of L (the eigenvectors corresponding to the k smallest eigenvalues of L)(15). As $\hat{X}$ has exactly one non-zero entry for each row, we can also identify the partitions by clustering vertices that corresponding rows in $\hat{X}$ are close to each other. This naturally suggests that we should also cluster vertices by row entries of U which could be seen as an approximated solution of $\hat{X}$. Given the matrix U, (15) suggests that we normalise rows of U to have a unit norm, called a new matrix $\bar{U}$ then run k-means on the rows of $\bar{U}$. We partition V corresponds to the clusters obtained from the k-means. By comparing this with the normalized laplacian clustering algorithm which is written explicitly in Algorithm 1 in the material and methods section, we can see that the hypergraph laplacian is a natural extension of the laplacian.

We have seen that this method successfully avoided working with graph representations by working with the normalized hypergraph laplacian instead. However, (1) shows that various formulations of learning problem on hypergraphs included the NH-Cut mentioned above are equivalent to certain graphs problems. We present key ideas and lemma from (1) below.

Lemma 1. *Let*

$$B = \begin{bmatrix} I & -A \\ -A^T & I \end{bmatrix}$$

be a block matrix with A rectangular. Consider the eigenvalue problem

$$\begin{bmatrix} I & -A \\ -A^T & I \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$$

the the following relation holds

$$AA^T \boldsymbol{x} = (1-\lambda)^2 \boldsymbol{x}$$

Proof. For the eigenvalue problem, we have

$$I\mathbf{x} - A\mathbf{y} = \lambda \mathbf{x}$$

$$-A^T \mathbf{x} + I\mathbf{y} = \lambda \mathbf{y}$$

which rearrange to

$$A\mathbf{y} = (1-\lambda)\mathbf{x}$$

$$-AA^T = A(\lambda - 1)\mathbf{y}$$

so we have

$$AA^T \mathbf{x} = (1-\lambda)^2 \mathbf{x}$$

□

Theorem 1. *NH-cut is equivalent to consructing a star expansion and using the normalized Laplacian on it*

Proof. Consider the star expansion of a hypergraph with an incidence matrix H. The adjacency matrix of the resulting graph can be written as

$$S^* = \begin{bmatrix} 0 & H \\ H^T & 0 \end{bmatrix}$$

The degree matrix of this graph is

$$D^* = \begin{bmatrix} D & 0 \\ 0 & \triangle \end{bmatrix}$$

when D and $\triangle$ are diagonal matrices with $D_{vv} = \deg(v)$ and $\triangle_{ee} = |e|$. Thus, the normalized laplacian for this graph is

$$\mathrm{D}^{*-1/2}(D^* - S^*)D^{*-1/2} = D^{*-1/2} \begin{bmatrix} D & -H \\ -H^T & \triangle \end{bmatrix} D^{*-1/2}$$
$$= \begin{bmatrix} I & -D^{-1/2} H \triangle^{-1/2} \\ -\triangle^{-1/2} H^T D^{-1/2} & I \end{bmatrix} \qquad [4]$$

(see Algorithm 1 for the normalized laplacian algorithm). Consider eigenvalue problem of this matrix, with eigenvector $\mathbf{x}^T = \begin{bmatrix} x_v & x_e \end{bmatrix}$. By Lemma 1, we have that

$$(D^{-1/2} H \triangle^{-1/2})(\triangle^{-1/2} H^T D^{-1/2}) x_v = (1-\lambda)^2 x_v$$

$$(D^{-1/2} H \triangle^{-1} H^T D^{-1/2}) x_v = (1-\lambda)^2 x_v$$

$$(I - D^{-1/2} H \triangle^{-1} H^T D^{-1/2}) x_v = (1 - (1-\lambda)^2) x_v$$

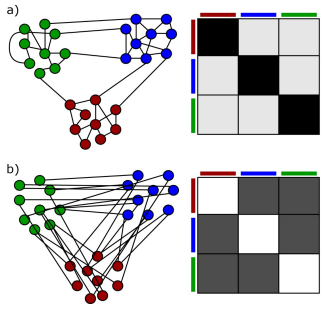

Fig. 2. Stochastic block models (29)

Compare this with [2], we can see that they have the same eigenvectors but correspond to different eigenvalues following an equation $\lambda_h = (1-(1-\lambda)^2)$ when λ_h denote the eigenvalue of [2] . Note that $(1-(1-\lambda)^2) = 2\lambda - \lambda^2$ is a non-decreasing function when $2 \geq \lambda$. (22) shows that all eigenvalues of a symmetric laplacian are in $[0, 2]$ that is $\lambda \in [0, 2]$. We can conclude that the first k eigenvectors of [2] is the same as the first k eigenvectors of this star expansion graph. □

(1) summarises that many existing hypergraph methodologies can be reduced to equivalent problems in graphs and perhaps it is graphs that lie in the heart of learning in higher order interactions.

We want to challenge this statement and explore further properties of graph projection methods, in particular, the star expansion. A natural question to ask is whether such graph projection works well in general with other algorithms for graphs. We will test this hypothesis by applying different community detection algorithms such as modularity or spectral clustering. Also, as mentioned in (14), many graphs data actually came from hypergraphs from clique expansion or star expansion and this usually happens in the pre-processing step that we may not aware of. Therefore, it would be interesting to see if many algorithms in graphs work and the pre-processing step does make sense.

Materials and Methods

We will test our algorithms on both synthesized and real data and in this section, we introduce models for our synthesized data and algorithms accordingly.

A stochastic block model is a generative model for blocks, groups, or communities in networks (24),(25). There has been a lot of work done on analysis of this model and also its application to community detection (26), (27), (28). For the rest of this report, we use the following terminology. For a graph G with n vertices, the i^{th} vertex, v_i has degree k_i and belong to the community g_i. A is the adjacency matrix of G.

The stochastic block model is constructed as follows, we define a $k \times k$ symmetric matrix B that the probability of an edge between v_i, v_j is B_{ij}.

There are extension of the stochastic block model to hypergraphs. (15) propose a planted partition model for sparse random non-uniform hypergraph. Before diving in to the model, we define related definition on tensor first. We call $B = \{B(i_1, \ldots, i_m)\}_{1 \leq i_1, \ldots, i_m \leq n}$ an *m-way tensor of dimension n*. A tensor is *symmetric* if it is invariant under a permutation of its vector arguments that is $B(i_1, \ldots, i_m) = B(j_1, \ldots, j_m)$ when $(j_1, \ldots, j_m)$ is a permutation of $(i_1, \ldots, i_m)$. We can see a tensor as a higher-dimension adjacency matrix for a hypergraph.

A. Planted partition model for hypergraph. The planted partition model for sparse random non-uniform hypergraph (15) generate a non-uniform hypergraph with n vertices with hyperedges of cardinality up to M and with k communities. Let M be an integer, represented the maximum edge cardinality in the hypergraph. For each $m = 2, \ldots, M$, every set $\{(i_1, \ldots, i_m)\} \subset V$, the probability of having an edge $e = \{(i_1, \ldots, i_m)\} \in E$ is

$$\mathbb{P}(\{(i_1, \ldots, i_m)\} \in E) = \alpha_{m,n} B^{(m)}(g_{i_1}, g_{i_2}, \ldots, g_{i_m})$$

when $B^{(m)}$ is a symmetric k-dimensional tensor of order m containing the probabilities of forming m-way edges among the different classes. While $\alpha_{m,n}$ allows for a sparsity scaling that does not depend on the partitions.

B. Degree Corrected Block Model for Hypergraph. We consider another extension of the stochastic block models. (25) argues that the simple block model does not work well to real-world network as the model is not flexible enough to generate networks with structure similar to one that found in empirical network data. (25) extends the model to take into account degree heterogeneity that we observe in the real-world data. (14) extends model in (25) to m-uniform hypergraphs with degree heterogeneity. First, we generate θ_i for each vertex v_i from the power law distribution with a parameter α to represent degree heterogeneity parameter associated with v_i. Let B be a non-negative symmetric k-dimensional tensor of order m containing the probabilities of forming m-way edges among the different classes. The probability of having an edge $e = \{(i_1, \ldots, i_m)\} \in E$ is

$$\mathbb{P}(\{(i_1, \ldots, i_m)\} \in E) = B(g_{i_1}, g_{i_2}, \ldots, g_{i_m}) \prod_{j=1}^{m} \theta_{i_j}$$

C. Normalized laplacian method. First, we present the normalized laplacian method on graph (16),(22). For a graph G with an adjacency matrix A, we have an algorithm as follows

Algorithm 1 Normalized laplacian method

Input: Adjacency matrix A

1: Compute the laplacian matrix $L = D - A$ when D is the diagonal matrix with $D_{vv} = \deg(v)$
2: Compute the normalized laplacian matrix $L_{sym} = D^{-1/2} L D^{-1/2}$
3: Compute the first k eigenvectors $v_1, \ldots, v_k$ of L_{sym} and let $V \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $v_1, \ldots, v_k$ as columns
4: Form a matrix U from V by normalising the row sums to have norm 1
5: Clustering the label by applying the k-means method to the rows of U

Output: Partitions according to k-means

As we state earlier, this will be the baseline model for our experiment.

D. Modularity method. Modularity, denoted by Q is a quantity introduced to measure the goodness of the partitioning of a network into communities (30),(31),(32).We define

$$Q = \frac{1}{2M}\sum_{i,j=1}^{n}(A_{ij} - \frac{k_i k_j}{2M})\delta(g_i, g_j)$$

where $\delta(g_i, g_j) = 1$ if $g_i = g_j$ and $\delta(g_i, g_j) = 0$ otherwise. A contribution to Q from a community C is

$$\sum_{v_i, v_j \in C}(A_{ij} - \frac{k_i k_j}{2M})$$

From the configuration model for graph (33),(32), for a vertex v_i, v_j the expected number of link between v_i, v_j is given by $A^*_{i,j} = \frac{k_i k_j}{2M}$. Intuitively, the modularity compares the number of edges in a community to the expected number of edges given by the configuration model. The higher value means edges are denser in that community and we prefer that behaviour. Therefore, we want to maximise the modularity. We will use the spectral optimization of modularity as in (32). Define a real symmetric matrix B with element

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2M}$$

From (32), we can rewrite

$$Q = \frac{1}{4M}s^T B s$$

when $s \in \mathbb{R}^{n\times 1}$ that $s_i = 1$ if the vertex i belongs to group 1 and -1 otherwise. This is a similar kind of combinatorial optimisation problem as we have seen for the NH-cut and the solution to the relaxed problem is the eigenvector u_1 which correspond to the largest eigenvalue of B. The vertices is partitioned into two part $\{v_i | u_{1i} \geq 0\}$ and $\{v_i | u_{1i} < 0\}$

E. SCORE. SCORE stands for Spectral Clustering On Ratios-of-Eigenvectors is a spectral clustering technique that has ability to deal with degree heterogeneity in graphs. According to (34), the innovation is to use entry-wise ratios between the first leading eigenvector and each of the other leading eigenvectors for clustering. This largely remove the influence of heterogeneity.

Algorithm 2 SCORE for 2 clusters

Input: Adjacency matrix A

1: Compute $\hat{\eta}_1, \hat{\eta}_2$, the two unit-norm eigenvectors of A associated with the largest and second largest eigenvalue (in magnitude) respectively

2: Compute $\hat{r}$ the $n \times 1$ vector of coordinatewise ratios:

$$\hat{r}(i) = \hat{\eta}_2(i)/\hat{\eta}_1(i)$$

for $1 \leq i \leq n$

3: Clustering the labels by applying the k-means method to the vector $\hat{r}$

Output: Partitions according to k-means

F. Asynchronous fluid communities. The idea of the Asynchronous fluid communities algorithm is based on the idea of fluids (communities) interacting in an environment, expanding and contracting as a result of the topology of the environment until a stable state is reached(35). The algorithm is given as follows

Algorithm 3 Asynchornous fluid communities

Input: A graph $G = (V, E)$

1: Initialize k fluid communities $C = \{c_1, \ldots, c_k\}$, and denote a density of each community c, $\mathrm{d}(c) = 1/|c|$

2: Iterate over all vertices of V in random order, updating the community each vertex belongs to using an update rule

$$C_v = \mathrm{argmax}_{c\in C}\sum_{w\in\{v,\Gamma(v)\}} d(c) \times \delta(g_w, c)$$

when $\Gamma(v)$ is the neighbor of v, g_w is the current community of w. C_v can have many values and if the original community of v is in C_v then v does not change a community. Otherwise, we assign the community of v randomly.

3: Return to step 2 until there is no community change in two consecutive steps.

Output: Communities C

G. Girvan-Newman and Kernighan–Lin. In this section, we discuss briefly on the Girvan-Newman (36) and Kernighan-Lin algorithm (37) as the runtime are quite high and we can't run them in many cases when our hypergraph is large in the experiment. The idea of the Girvan-Newman is to remove edges that are least central, which is measured by edge betweenness. They propose that progressively removing edges like this will reveal a community structure.

The idea of the Kernighan-Lin algorithm is to minimise a cut between two partitions (says A, B) which is the number of edges or the sum of weights of edges that cross from A to B and it proposes a greedy algorithm to do the task.

Experiment

We present our experiment results in this section. In the rest of this report, we will experiment with a hypergraph with 2 real communities and we will try to recover them back using algorithm mentioned above on different scenarios.

First, we begin with the planted partition model when $M = 3$ which generates a hypergraph with 2-way and 3-way edges. We look at 4 cases,

- **Case 1:** no signals from 2-way edges
- **Case 2:** strong signals from 2-way edges
- **Case 3:** more 3-way edges
- **Case 4:** less 3-way edges

By this we differ the tensor B and parameter α that we use for model generations. For example, we set $B^{(2)}(0,0) = 0.2, B^{(2)}(1,0) = 0.1, B^{(2)}(0,0) = 0.2$ for the case 1 while $B^{(2)}(0,0) = 0.5$ in the case 2. Also, we change the value of α to control the number of hyperedges in the generated graph. You can find more detail on parameters in the appendix. We

Table 1. Planted partition model, M = 3, n = 100

Case	# of edges	NH-cut	SCORE	Modularity	AF
Case 1	(229, 799)	**3.0±2.2**	45.1±3.3	10.9±8.0	41.5±6.1
Case 2	(346, 797)	**1.0±0.8**	41.1±10.2	13.0±15.9	32.9±10.1
Case 3	(236, 1424)	**0.7±0.9**	46.3±2.9	4.8±5.6	38.7±8.1
Case 4	(236, 329)	**11.0±3.9**	41.2±6.7	35.6±6.6	40.6±6.0

Table 2. Planted partition model, M = 3, n = 50

NH-cut	SCORE	Modularity	AF	GN	KL
9.2±10.0	32.8±10.6	31.6±17.8	36.4±7.5	23.6±17.0	38±6.3

Table 3. Planted partition model, M = 3, more realistic cases

# of edges	NH-cut	SCORE	Modularity	AF
(938, 805)	**0.2±0.4**	38.6±12.4	10.6±16.9	36.0±9.5
(1013, 256)	**0.3±0.6**	9.8±11.9	31.1±14.8	35.2±10.4

Table 4. Planted partition model, M = 4

# of edges	NH-cut	SCORE	Modularity	AF
(323, 775, 728)	**0.7±0.8**	44.9±2.7	5.8±9.9	42.2±6.6
(341, 751, 2403)	**0.1±0.3**	46.4±3.2	5.2±14.6	40.4±6.6
(354, 107, 2282)	**0.3±0.5**	45.4±3.4	**1.9±2.0**	36.6±9.3

Table 5. hDCBM model

α	# of edges	NH-cut	SCORE	Modularity	AF
3	1543	**1.3±1.9**	44.6±6.2	34.8±12.5	33.0±11.4
4	667	**4.2±3.2**	46.3±4.7	27.8±11.1	37.4±7.6
5	482	**8.2±3.8**	44.5±5.7	27.4±8.4	40.4±7.3

generated hypergraphs with 100 vertices and table 1 shows the number of 2-way edges, 3-ways edges sample from each scenarios in a pair (2-ways, 3-ways) and also error rate in percentages (mean ± standard deviation) from running each algorithm 20 times. By error rate we mean the number of nodes that are clustered to a wrong community.

As you can see, we presented only 4 algorithms above as the author does not have sufficient computing power to run the Girvan-Newman and Kernighan–Lin algorithm on $n = 100$ vertices. Therefore, we ran a simulation when $n = 50$ vertices for 5 times and the result is shown in table 2.

Next, we moved to a more realistic case when the number of 2-way edges is much higher than 3-way edges as observed in the empirical data (15). We run simulation when $n = 100$ vertices for 20 times and the result is shown in table 3. We can see that the NH-cut gave us unmatched performance compared to other algorithms. The SCORE and Asynchronous fluid communities algorithm produced a quite high error although the error decreased significantly when we have more 2-way edges in the case of SCORE. The modularity approach return a quite small error rate and it is the second best algorithm. However, we observe in the table 3 that as we decrease the number of 3-way edges, the modularity approach performance dropped sharply.

Next, we present results from the planted partition model when $M = 4$ which generates a hypergraph with 2-way, 3-way and 4-way edges. We also run the simulation with $n = 100$ vertices for 10 times. We test this with 2 cases when the number of 4-way edges is about the same as 3-way edges and when the number of 4-way edges is much higher.

We see that the NH-cut still performed best followed by the modularity approach. Notice that the results improve noticeably for the modularity approach as we have more 4-way edges and less 3-way edges. With more computing power, the author would be interested to see what would happen when we have hyperedges of higher order.

Next, we also consider a hDCBM model which could capture the degree heterogeneity behaviour. We ran a simulation on a 3-uniform hypergraph with $n = 100$, $\alpha = 3, 4, 5$ which control the power law distribution where the lower means the higher probability for a vertex to have a higher degree. After 20 simulations, the result is shown in the table 5 and as expected, the high probability Although, we expected to see a better performance when we have a more uniform hypergraph (the higher value of α), the number of edges decreased accordingly and the performance when α is low is better. We varied the tensor matrix B so that each α give about the same number of edges and compare algorithms again in the table 6. As expected, the less the degree heterogeneity, the better results.

Lastly, we conduct experiment with real world datasets. We choose the Mushroom, Nursery and Congressional Voting Records datasets from the UCI Machine Learning Repository (38). Mushroom dataset contains 8123 instances of mushrooms where each is identified as edible or poisonous and described by 22 attributes such as size, colour. Nursery dataset contains 12960 instances of family nursery-school applications during several years in 1980s where each family has 8 attributes such as financial standing of the family, health condition. Each family has a decision result but for simplicity, we assume that there is 2 communities : not recommended and the rest. Congressional Voting Records datasets. This data set includes votes from politicians for each of the U.S. House of Representatives Congressmen on the 16 key votes and there are 2 classes of the politician, Republican and Democrat. We constructed a hypergraph for each dataset where the same attribute values are hyperedges. The result is shown in table 7. With the limitation of computing power, we can't run the Girvan-Newman and Kernighan–Lin algorithm and for the Mushroom and Nursery data. The NH-cut and modularity approach have similar result in both Mushroom and Voting dataset. Surprisingly, SCORE gave the lowest error in the Nursery data set.

To sum up, according to our simulations the NH-cut al-

Table 6. hDCBM model

α	# of edges	NH-cut	SCORE	Modularity	AF
3	1315	**1.3±1.6**	46.6±2.4	35.9±13.7	36.6±11.0
4	1348	**0.9±1.0**	42.5±7.1	33.5±13.7	39.5±6.6
5	1336	**0.9±1.2**	41.18.5	29.6±14.4	38.4±8.7

Table 7. Realworld data

Dataset	NH-cut	SCORE	Modularity	AF	GN	KL
Mushsroom	**10.9**	48.2	**12.1**	21.1	-	-
Nursery	39.6	**33.3**	50.0	40.8	-	-
Voting	**17.1**	38.5	**17.1**	**16.8**	38.5	**16.4**

gorithm has an ability to recover real communities from a hypergraph with a very high accuracy. The modularity approach managed to deliver a reasonably good result compared to other algorithms and the accuracy improved as we have edges with a higher order. In the real world data, NH-cut and modularity approach provide a comparable result. Moreover, not every community detection algorithm in graph does work well with the star expansion which may indicate that we have to be more careful with this type of graph projection.

Discussion and Conclusion

Discussion. In this section, we want to discuss the modelling issues. First, current models for generating a hypergraph is computationally expensive. A runtime for the planted partition model for hypergraph when $M = 4$ is $O(n^4)$ and in general is $O(n^M)$ as we have to run the algorithm over $\binom{n}{M}$ combinations of vertices. A hypergraph in the real world can have a hyperedge that contains a large portion of vertices and current models could not generate a hypergraph to mimic this behaviour. Next, we note that in the hDCBM model the number of m-way edges is $O(n^m)$ which is too large for the author to run a simulation on so the author multiply a factor of $O(n^{-(m-2)})$ to the probability of generating an edge to control the number of edges in the hypergraph. Also, we only considered a limited number of algorithms as well as test cases (due to runtime and time constraint). Also, note that we only look at the problem in the community detection context and properties of graph representations for other tasks such as centrality or dynamics would also be an interesting area to explore. Another interesting question is that for any algorithm on graphs A, is there a projection of a hypergraph G to its graph representation G' that when applying the algorithm A on G' it gives us a satisfactory result or insight of G on a certain task (e.g. Star expansion for normalized laplacian clustering) or is there a projection of a hypergraph that preserve a certain property such as dynamics of it.

Conclusion. In this report, we try to get an insight to a question whether a hypergraph works well in general with other algorithm for graphs. We conclude that the answer is not necessarily, as we can see from results of SCORE and AF algorithms. However, the representation definitely capture meaningful properties of the hypergraph as we can see good results from both NH-cut and modularity approach even if these methods are very different (one minimises cut, one maximises modularity). As in (14), hypergraph projection usually happens at the pre-processing level and we might not be aware of it, our result implies that we have to be more careful on a pre-processed graph whether it comes from a hypergraph and what graph projection method is applied to it, knowing this could give us useful prior information on tools to analyse the graph.

1. S Agarwal, K Branson, S Belongie, Higher order learning with graphs in *Proceedings of the 23rd international conference on Machine learning*. pp. 17–24 (2006).
2. D Zhou, J Huang, B Schölkopf, Learning with hypergraphs: Clustering, classification, and embedding in *Advances in neural information processing systems*. pp. 1601–1608 (2007).
3. S Fortunato, Community detection in graphs. *Phys. reports* **486**, 75–174 (2010).
4. A Landherr, B Friedl, J Heidemann, A critical review of centrality measures in social networks. *Bus. & Inf. Syst. Eng.* **2**, 371–385 (2010).
5. I Gertsbakh, Epidemic process on a random graph: some preliminary results. *J. Appl. Probab.* **14**, 427–438 (1977).
6. NP Hummon, P Dereian, Connectivity in a citation network: The development of dna theory. *Soc. networks* **11**, 39–63 (1989).
7. R Davies, GF Royle, Graph domination, tabu search and the football pool problem. *Discret. Appl. Math.* **74**, 217–228 (1997).
8. W Yang, G Wang, MZA Bhuiyan, KKR Choo, Hypergraph partitioning for social networks based on information entropy modularity. *J. Netw. Comput. Appl.* **86**, 59–71 (2017).
9. E Estrada, JA Rodríguez-Velázquez, Subgraph centrality and clustering in complex hyper-networks. *Phys. A: Stat. Mech. its Appl.* **364**, 581–594 (2006).
10. JY Zien, MD Schlag, PK Chan, Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. *IEEE Transactions on computer-aided design integrated circuits systems* **18**, 1389–1399 (1999).
11. JW Grossman, The evolution of the mathematical research collaboration graph. *Congr. Numerantium*, 201–212 (2002).
12. ME Newman, The structure of scientific collaboration networks. *Proc. national academy sciences* **98**, 404–409 (2001).
13. P Ji, J Jin, , et al., Coauthorship and citation networks for statisticians. *The Annals Appl. Stat.* **10**, 1779–1812 (2016).
14. ZT Ke, F Shi, D Xia, Community detection for hypergraph networks via regularized tensor power iteration. *arXiv preprint arXiv:1909.06503* (2019).
15. D Ghoshdastidar, A Dukkipati, , et al., Consistency of spectral hypergraph partitioning under planted partition model. *The Annals Stat.* **45**, 289–315 (2017).
16. AY Ng, MI Jordan, Y Weiss, On spectral clustering: Analysis and an algorithm in *Advances in neural information processing systems*. pp. 849–856 (2002).
17. J Shi, J Malik, Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis machine intelligence* **22**, 888–905 (2000).
18. C Kim, AS Bandeira, MX Goemans, Community detection in hypergraphs, spiked tensor models, and sum-of-squares in *2017 International Conference on Sampling Theory and Applications (SampTA)*. (IEEE), pp. 124–128 (2017).
19. A Anandkumar, H Sedghi, Learning mixed membership community models in social tagging networks through tensor methods. *arXiv preprint arXiv:1503.04567* (2015).
20. D Ghoshdastidar, A Dukkipati, A provable generalized tensor spectral method for uniform hypergraph partitioning in *International Conference on Machine Learning*. pp. 400–409 (2015).
21. T Kumar, S Vaidyanathan, H Ananthapadmanabhan, S Parthasarathy, B Ravindran, Hypergraph clustering: A modularity maximization approach. *arXiv preprint arXiv:1812.10869* (2018).
22. U Von Luxburg, A tutorial on spectral clustering. *Stat. computing* **17**, 395–416 (2007).
23. N Veldt, AR Benson, J Kleinberg, Hypergraph cuts with general splitting functions. *arXiv preprint arXiv:2001.02817* (2020).
24. PW Holland, KB Laskey, S Leinhardt, Stochastic blockmodels: First steps. *Soc. networks* **5**, 109–137 (1983).
25. B Karrer, ME Newman, Stochastic blockmodels and community structure in networks. *Phys. review E* **83**, 016107 (2011).
26. K Rohe, S Chatterjee, B Yu, , et al., Spectral clustering and the high-dimensional stochastic blockmodel. *The Annals Stat.* **39**, 1878–1915 (2011).
27. DL Sussman, M Tang, DE Fishkind, CE Priebe, A consistent adjacency spectral embedding for stochastic blockmodel graphs. *J. Am. Stat. Assoc.* **107**, 1119–1128 (2012).
28. E Abbe, AS Bandeira, G Hall, Exact recovery in the stochastic block model. *IEEE Transactions on Inf. Theory* **62**, 471–487 (2015).
29. T Funke, T Becker, Stochastic block models: A comparison of variants and inference methods. *PloS one* **14** (2019).
30. U Brandes, et al., On modularity clustering. *IEEE transactions on knowledge data engineering* **20**, 172–188 (2007).
31. ME Newman, Modularity and community structure in networks. *Proc. national academy sciences* **103**, 8577–8582 (2006).
32. R Lambiotte, C5.4 networks (year?).
33. M Newman, *Networks*. (Oxford university press), (2018).
34. J Jin, , et al., Fast community detection by score. *The Annals Stat.* **43**, 57–89 (2015).
35. F Parés, et al., Fluid communities: a competitive, scalable and diverse community detection algorithm in *International Conference on Complex Networks and their Applications*. (Springer), pp. 229–240 (2017).
36. M Girvan, ME Newman, Community structure in social and biological networks. *Proc. national academy sciences* **99**, 7821–7826 (2002).
37. BW Kernighan, S Lin, An efficient heuristic procedure for partitioning graphs. *The Bell Syst. Tech. J.* **49**, 291–307 (1970).
38. D Dua, C Graff, UCI machine learning repository (2017).

Appendix

Additional figures. We include example figures of hypergraphs that we generated from planted model and hDCBM model. Each figure is a star expansion graph where red nodes are vertices and blue nodes are edges in the original hypergraph.

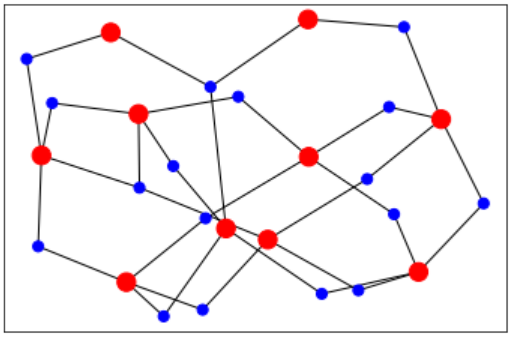

Fig. 3. Planted partition model, M = 3

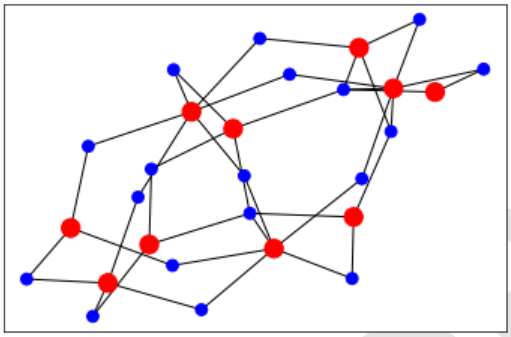

Fig. 4. Planted partition model, M = 4

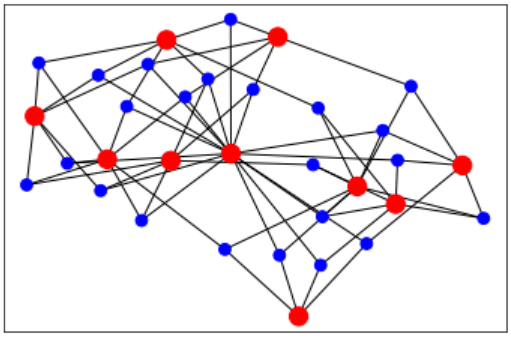

Fig. 5. hDCBM model

Parameters. For simplicity we represent a symmetric tensor $B^{(2)}$ with $(B^{(2)}_{00}, B^{(2)}_{10}, B^{(2)}_{11})$ and $B^{(3)}$ with $(B^{(3)}_{000}, B^{(3)}_{100}, B^{(3)}_{110}, B^{(3)}_{111})$ in this order and so on. The following are parameters that we used to generate hypergraphs in the **table 1**

$\alpha_2 = 1/n^{0.25}, \alpha_3 = 1/n^{0.75}$

- Case 1: $B : (0.2, 0.1, 0.2), (0.3, 0.1, 0.1, 0.3)$
- Case 2: $B : (0.2, 0.1, 0.5), (0.3, 0.1, 0.1, 0.3)$
- Case 3: $B : (0.2, 0.1, 0.5), (0.5, 0.2, 0.2, 0.5)$
- Case 4: $B : (0.2, 0.1, 0.5), (0.1, 0.05, 0.05, 0.1)$

Table 2

$\alpha_2 = 1/n^{0.25}, \alpha_3 = 1/n^{0.75}$

- $B : (0.2, 0.1, 0.5), (0.5, 0.2, 0.2, 0.5)$

Table 3

$B : (0.2, 0.1, 0.5), (0.3, 0.1, 0.1, 0.3)$

- $\alpha_2 = 1, \alpha_3 = 1/n^{0.75}$
- $\alpha_2 = 1, \alpha_3 = 1/n$

Table 4

$B : (0.2, 0.1, 0.5), (0.1, 0.05, 0.05, 0.1), (0.4, 0.2, 0.1, 0.2, 0.4)$

- $\alpha_2 = 1/n^{0.25}, \alpha_3 = 1/n^{0.55}, \alpha_4 = 1/n^{1.5}$
- $\alpha_2 = 1/n^{0.25}, \alpha_3 = 1/n^{0.55}, \alpha_4 = 1/n^{1.25}$
- $\alpha_2 = 1/n^{0.25}, \alpha_3 = 1/n^{1}, \alpha_4 = 1/n^{1.25}$

Table 5

$B : (0.3, 0.1, 0.1, 0.3)$

- $\alpha = 3$
- $\alpha = 4$
- $\alpha = 5$

Table 6

In this case we set $B = f \times (0.3, 0.1, 0.1, 0.3)$ to get similar number of edges for different values of alpha.

- $\alpha = 3, f = 0.8$
- $\alpha = 4, f = 1.7$
- $\alpha = 5, f = 2.3$