PLOS ONE

RESEARCH ARTICLE

Clustering via hypergraph modularity

Bogumił Kamiński[1], Valérie Poulin[2], Paweł Prałat[3]*, Przemysław Szufel[1], François Théberge[2]

1 SGH Warsaw School of Economics, Warsaw, Poland, **2** The Tutte Institute for Mathematics and Computing, Ottawa, ON, Canada, **3** Department of Mathematics, Ryerson University, Toronto, ON, Canada

* pralat@ryerson.ca

Citation: Kamiński B, Poulin V, Prałat P, Szufel P, Théberge F (2019) Clustering via hypergraph modularity. PLoS ONE 14(11): e0224307. https://doi.org/10.1371/journal.pone.0224307

Editor: Sabrina Gaito, Universitá degli Studi di Milano, ITALY

Received: February 21, 2019

Accepted: October 10, 2019

Published: November 6, 2019

Data Availability Statement: We included 2 examples in our paper. The first one is a synthetic hypergraph that can be regenerated by the reader (details are provided in [26]). The second one is a DBLP hypergraph - all information and our preprocessed data is available through the GitHub gist repository that we referenced in the paper, that is, https://gist.github.com/pszufe/02666497d2c138d1b2de5b7f67784d2b.

Funding: The presented research was partially financed with the support of financed by the Polish National Agency for Academic Exchange (NAWA), on the basis of grant agreement no. PPI/APM/

Abstract

Despite the fact that many important problems (including clustering) can be described using hypergraphs, theoretical foundations as well as practical algorithms using hypergraphs are not well developed yet. In this paper, we propose a hypergraph modularity function that generalizes its well established and widely used graph counterpart measure of how clustered a network is. In order to define it properly, we generalize the Chung-Lu model for graphs to hypergraphs. We then provide the theoretical foundations to search for an optimal solution with respect to our hypergraph modularity function. A simple heuristic algorithm is described and applied to a few illustrative examples. We show that using a strict version of our proposed modularity function often leads to a solution where a smaller number of hyperedges get cut as compared to optimizing modularity of 2-section graph of a hypergraph.

1 Introduction

An important property of complex networks is their community structure, that is, the organization of vertices in clusters, with many edges joining vertices of the same cluster and comparatively few edges joining vertices of different clusters [1, 2]. In social networks, communities may represent groups by interest (practical application include collaborative tagging—[3]), in citation networks they correspond to related papers (see [4]), similarly in the web communities are formed by pages on related topics.

Yet another example could be financial markets where we have several groups of financial instruments that might be correlated with each other in several different groups. Such groups can be represented as hyperedges and hence detection of communities in such hypergraph could lead to better understanding of dependencies between financial instruments.

Hypergraphs can also be used to model transportation systems. For example in [5] the authors consider transportation system represented as a directed hypergraph. A hyperedge can represent a situation where a single stop (starting or destination point) is being serviced by several public transportation lines and vehicle types that can be differently chosen by an agent traveling within the transportation grid and communities can represent paths taken often together. Another application was presented in [6]—the authors suggest using hypergraphs to model interactions between biological cells in computational biology. Finally, in [7] one can find a discussion on how hypergraphs can be used for modeling telecommunication systems

2018/1/00037. It is also related to the NSERC Discovery grant entitled "Modelling and Mining Complex Networks" received by Paweł Prałat.

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

and it is argued that using hypergraphs to represent communication within a mobile grid conveys more information that using regular graphs. Again, this information can be used for a social community detection.

Being able to identify communities in a network could help us to exploit this network more effectively. For example, clusters in citation graphs may help to find similar scientific papers, discovering users with similar interests is important for targeted advertisement, clustering can also be used for network compression and visualization.

The key ingredient for many clustering algorithms is *modularity*, which is at the same time a global criterion to define communities, a quality function of community detection algorithms, and a way to measure the presence of community structure in a network. Modularity for graphs was introduced by Newman and Girvan [8] and it is based on the comparison between the actual density of edges inside a community and the density one would expect to have if the vertices of the graph were attached at random regardless of community structure, while respecting the vertices' degree on average. This random family of graphs is known as the Chung-Lu random model [9].

Myriad of problems can be described in hypergraph terms, however, despite being formally defined in the 1960s (and various realizations studied long before that) hypergraph theory is patchy and often not sufficiently general. The result is a lack of machinery for investigating hypergraphs, leading researchers and practitioners to create the 2-section graph of a hypergraph of interest [10–16] or to restrict their study to d-uniform hypergraphs [17, 18]. In taking the 2-section (that is, making each hyperedge a clique) we lose some information about edges of size greater than two. Sometimes losing this information does not affect our ability to answer questions of interest, but in other cases it has a profound impact. In particular, an important scenario when hypergraph-based approach can be preferred, is when a large hyperedge connecting some vertices is a strong indicator that they all belong to the same community. Such situations occur often in practice. Let us briefly discuss two simple examples. First consider e-mails as hyperedges of a hypergraph whose vertices are e-mail addresses. Multiple addresses in an e-mail (group e-mails) most likely are not sent to random people, but rather to some community of common interests. As a second example consider a platform like GitHub, where vertices are users and hyperedges are repositories linking users that committed to them. Again, if a group of users commits to the same repository it is most likely a strong indicator that they form some community. In both cases, as indicated above, replacing a hyperedge by a clique would lose valuable information.

The paper is organized as follows. In Section 2, we review the Chung-Lu model for graphs and its link to the modularity function. We then propose a generalization of the Chung-Lu model for hypergraphs, as well as a hypergraph modularity function. In Section 3, we provide the framework to develop algorithms using our hypergraph modularity function. We propose an hypergraph partitioning algorithm and a few illustrative examples in Section 4. This is a new measure we are proposing, and there is plenty of future work to do, which we summarize in Section 5. Additionally, we made the source codes available on-line [19] allowing to reproduce the analyses presented in the paper.

2 Hypergraph modularity

In this section, we recall the definition of modularity function for graphs, and we propose its generalization for hypergraphs. Throughout the paper we will use n for the number of vertices. We will use $\binom{X}{k}$ for the set consisting of all k-element subsets of X. Finally, $[n] \coloneqq \{1, \ldots, n\}$.

2.1 Chung-Lu model for graphs

Let $G = (V, E)$ be a graph, where $V = \{v_1, \ldots, v_n\}$ are the vertices, the edges E are multisets of V of cardinality 2 (loops allowed), and $deg_G(v)$ is the degree of v in G (with a loop at v contributing 2 to the degree of v). For $A \subseteq V$, let the *volume* of A be $vol_G(A) = \sum_{v \in A} deg_G(v)$; in particular $vol_G(V) = \sum_{v \in V} deg_G(v) = 2|E|$. We will omit the subscript G when the context is clear.

We define $\mathcal{G}(G)$ to be the probability distribution on graphs on the vertex set V following the well-known Chung-Lu model [20–23]. In this model, each set $e = \{v_i, v_j\}$, $v_i, v_j \in V$, is independently sampled as an edge with probability given by:

$$P(v_i, v_j) = \begin{cases} \frac{deg(v_i)deg(v_j)}{2|E|}, & i \neq j \\ \frac{deg^2(v_i)}{4|E|}, & i = j. \end{cases}$$

(Let us mention about one technical assumption. Note that it might happen that $P(v_i, v_j)$ is greater than one and so it should really be regarded as the expected number of edges between i and j; for example, as suggested in [24], one can introduce a Poisson-distributed number of edges with mean $P(v_i, v_j)$ between each pair of vertices i, j. However, since typically the maximum degree Δ satisfies $\Delta^2 \leq 2|E|$ it rarely creates a problem and so we may assume that $P(v_i, v_j) \leq 1$ for all pairs.)

This model is a function of the degree sequence of G. One desired property of this random model is that it yields a distribution that preserves the expected degree for each vertex, namely: for any $i \in [n]$,

$$\begin{aligned} \mathbb{E}_{G' \sim \mathcal{G}(G)}[deg_{G'}(v_i)] &= \sum_{j \in [n] \setminus \{i\}} \frac{deg(v_i)deg(v_j)}{2|E|} + 2 \cdot \frac{deg^2(v_i)}{4|E|} \\ &= \frac{deg(v_i)}{2|E|} \sum_{j \in [n]} deg(v_j) = deg(v_i), \end{aligned}$$

where all degrees are with respect to graph G. This model will be useful to understand the graph modularity definition and its generalization to hypergraphs.

2.2 Review of graph modularity

The definition of modularity for graphs was first introduced by Newman and Girvan in [8]. Despite some known issues with this function such as the "resolution limit" reported in [25], many popular algorithms for partitioning large graph data sets use it [26–28]. It was also recently studied for some models of complex networks [29–32]. The modularity function favours partitions in which a large proportion of the edges fall entirely within the parts (note that throughout the paper, we use the term "part" and "partition" that are more common in mathematical literature; in the information sciences the equivalent term is "cluster") and biases against having too few or too unequally sized parts.

For a graph $G = (V, E)$ and a given partition $\mathbf{A} = \{A_1, \ldots, A_k\}$ of V, the modularity function is defined as follows:

$$\begin{aligned} q_G(\mathbf{A}) &= \frac{1}{|E|} \sum_{A_i \in \mathbf{A}} \left(e_G(A_i) - \mathbb{E}_{G' \sim \mathcal{G}(G)}[e_{G'}(A_i)] \right) \\ &= \sum_{A_i \in \mathbf{A}} \frac{e_G(A_i)}{|E|} - \sum_{A_i \in \mathbf{A}} \frac{\mathbb{E}_{G' \sim \mathcal{G}(G)}[e_{G'}(A_i)]}{|E|}, \end{aligned} \tag{1}$$

where $e_G(A_i) = |\{\{v_j, v_k\} \in E: v_j, v_k \in A_i\}|$ is the number of edges in the subgraph of G induced by the set A_i. The modularity measures the deviation of the number of edges of G that lie inside parts (clusters) of $\mathbf{A}$ from the corresponding expected value based on the Chung-Lu distribution $\mathcal{G}(G)$. The expected value for part A_i is

$$\begin{aligned}\mathbb{E}_{G' \sim \mathcal{G}(G)}[e_{G'}(A_i)] &= \sum_{\{v_j,v_k\} \in \binom{A_i}{2}} \frac{deg(v_j)deg(v_k)}{2|E|} + \sum_{v_j \in A_i} \frac{deg^2(v_j)}{4|E|} \\ &= \frac{1}{4|E|}\left(\sum_{v_j \in A_i} deg(v_j)\right)^2 = \frac{(vol(A_i))^2}{4|E|}.\end{aligned}$$

The first term in (1), $\sum_{A_i \in \mathbf{A}} e_G(A_i)/|E|$, is called the *edge contribution*, whereas the second one, $\sum_{A_i \in \mathbf{A}} (vol(A_i))^2/4|E|^2$, is called the *degree tax*. It is easy to see that $q_G(\mathbf{A}) \leq 1$. Also, if $\mathbf{A} = \{V\}$, then $q_G(\mathbf{A}) = 0$, and if $\mathbf{A} = \{\{v_1\}, \ldots, \{v_n\}\}$, then $q_G(\mathbf{A}) = -\frac{\sum deg(v)^2}{4|E|^2} < 0$.

The maximum *modularity* $q^*(G)$ is defined as the maximum of $q_G(\mathbf{A})$ over all possible partitions $\mathbf{A}$ of V; that is, $q^*(G) = \max_{\mathbf{A}} q_G(\mathbf{A})$. In order to maximize $q_G(\mathbf{A})$ one wants to find a partition with large edge contribution subject to small degree tax. If $q^*(G)$ approaches 1 (which is the trivial upper bound), we observe a strong community structure; conversely, if $q^*(G)$ is close to zero (which is the trivial lower bound), there is no community structure. The definition in (1) can be generalized to weighted edges by replacing edge counts with sums of edge weights.

2.3 Generalization of the Chung-Lu model to hypergraphs

Consider a hypergraph $H = (V, E)$ with $V = \{v_1, \ldots, v_n\}$, where hyperedges $e \in E$ are subsets of V of cardinality greater than one. Since we are concerned with not necessarily simple hypergraphs, hyperedges are multisets. Such hyperedges can be described using distincts sets of pairs $e = \{(v, m_e(v)): v \in V\}$ where $m_e(v) \in \mathbb{N} \cup \{0\}$ is the multiplicity of the vertex v in e (including zero which indicates that v is not present in e). Then $|e| = \sum_v m_e(v)$ is the *size* of hyperedge e and the *degree of a vertex* v in H is defined as $deg_H(v) = \sum_{e \in E} m_e(v)$. When the reference to the hyperedge is clear from the context, we simply use m_i to denote $m_e(v_i)$.

A hypergraph is said to be *d-uniform* if all its hyperedges have size d. In particular, a 2-uniform hypergraph is simply a graph. All hypergraphs H can be expressed as the disjoint union of d-uniform hypergraphs $H = \bigcup H_d$, where $H_d = (V, E_d)$, $E_d \subseteq E$ are all hyperedges of size d, and $deg_{H_d}(v)$ is the d-degree of vertex v. As for graphs, the volume of a vertex subset $A \subseteq V$ is $vol_H(A) = \sum_{v \in A} deg_H(v)$.

Similarly to what we did for graphs, we define a random model on hypergraphs, $\mathcal{H}(H)$, where the expected degrees of all vertices are the corresponding degrees in H. To simplify the notation, we omit the explicit references to H in the remaining of this section; in particular, $deg(v)$ denotes $deg_H(v)$, $\mathcal{H}$ denotes $\mathcal{H}(H)$, E_d denotes the edges of H of size d. Moreover, we use E' to denote the edge set of H'.

Let F_d be the family of multisets of size d; that is,

$$F_d := \left\{ \{(v_i, m_i) : 1 \leq i \leq n\} : \sum_{i=1}^{n} m_i = d \right\}.$$

The hypergraphs in the random model are generated via independent random experiments.

For each d such that $|E_d| > 0$, the probability of generating the edge $e \in F_d$ is given by:

$$P_{\mathcal{H}}(e) = |E_d| \cdot \binom{d}{m_1, \ldots, m_n} \prod_{i=1}^{n} \left(\frac{deg(v_i)}{vol(V)}\right)^{m_i}. \tag{2}$$

(Recall that $m_i = m_e(v_i)$.) Let $(X_1^{(d)}, \ldots, X_n^{(d)})$ be the random vector following a multinomial distribution with parameters $d, p_{\mathcal{H}}(1), \ldots, p_{\mathcal{H}}(n)$, where $p_{\mathcal{H}}(i) = deg(v_i)/vol(V)$ and $\sum_{i\in[n]} p_{\mathcal{H}}(i) = 1$; that is,

$$s_{\mathcal{H}}(e) \coloneqq \mathbb{P}\left((X_1^{(d)}, \ldots, X_n^{(d)}) = (m_1, \ldots m_n)\right) = \binom{d}{m_1, \ldots, m_n} \prod_{i=1}^{n} (p_{\mathcal{H}}(i))^{m_i}.$$

Note that this is the expression found in (2); that is, $P_{\mathcal{H}}(e) = |E_d| \cdot s_{\mathcal{H}}(e)$. As a result, alternatively one can think about the following auxiliary process. Select a random multiset consisting of d vertices (counting possible repetitions); in d independent rounds, vertex v_i is selected with probability $p_{\mathcal{H}}(i)$. Repeat this experiment $|E_d|$ times and use the expected number of times edge e occurred in this process for the value of $P_{\mathcal{H}}(e)$. An immediate consequence of this coupling between the two processes is that the expected number of edges of size d is $|E_d|$. Finally, as with the graph Chung-Lu model, if $P_{\mathcal{H}}(e) > 1$, then it should be regarded as the expectation and multi-hypergraph should be considered instead. However, as before, from practical point of view it is safe to assume that all $P_{\mathcal{H}}(e) \leq 1$.

In order to compute the expected d-degree of a vertex $v_i \in V$, note that

$$deg_{H'_d}(v_i) = \sum_{e\in F_d} m_e(v_i) \cdot \mathbb{I}_{\{e\in E'\}},$$

where $\mathbb{I}_{\{\}}$ is the indicator random variable. Hence, using the linearity of expectation, then splitting the sum into $d + 1$ partial sums for different multiplicities of v_i, we get:

$$\begin{aligned}
\mathbb{E}_{H'\sim\mathcal{H}}(deg_{H'_d}(v_i)) &= \sum_{e\in F_d} m_e(v_i) \cdot P_{\mathcal{H}}(e) = |E_d| \sum_{e\in F_d} m_e(v_i) \cdot s_{\mathcal{H}}(e) \\
&= |E_d| \sum_{m=0}^{d} m \sum_{e\in F_d; m_e(v_i)=m} s_{\mathcal{H}}(e) \\
&= |E_d| \sum_{m=0}^{d} m \cdot \mathbb{P}(X_i^{(d)} = m) \\
&= |E_d| \sum_{m=0}^{d} m \cdot \binom{d}{m} (p_{\mathcal{H}}(i))^m (1 - p_{\mathcal{H}}(i))^{d-m} \\
&= |E_d| \cdot d \cdot p_{\mathcal{H}}(i).
\end{aligned}$$

The second last equality follows from the fact that we obtained the expected value of a random variable with binomial distribution. One can compute the expected degree as follows:

$$\mathbb{E}_{H'\sim\mathcal{H}}[deg_{H'}(v_i)] = \sum_{d\geq 2} \frac{d \cdot |E_d| \cdot deg(v_i)}{vol(V)} = deg(v_i),$$

since $vol(V) = \sum_{d\geq 2} d \cdot |E_d|$.

We will use the generalization of the Chung-Lu model to hypergraphs as a null model allowing us to define hypergraph modularity.

2.4 Hypergraph modularities

Consider a hypergraph $H = (V, E)$ and $\mathbf{A} = \{A_1, \ldots, A_k\}$, a partition of V. For edges of size greater than 2, several definitions can be used to quantify the edge contribution given $\mathbf{A}$, such as:

(a). all vertices of an edge have to belong to one of the parts (clusters) to contribute; this is a *strict* definition that we focus on in this paper;

(b). the *majority* of vertices of an edge belong to one of the parts;

(c). at least 2 vertices of an edge belong to the same part; this is implicitly used when we replace a hypergraph with its 2-section graph representation.

We see that the choice of hypergraph modularity function is not unique; in fact, it depends on how strongly we believe that a hyperedge is an indicator that vertices belonging to it fall into one community. More importantly, one needs to decide how often vertices in one community "blend" together with vertices from other community; that is, how hermetic the community is. In particular, option (c) is the *softest* one and leads to standard 2-section graph modularity. In order to illustrate how one can obtain formulas for a given variant, we concentrate on the second extreme, option (a), that we call *strict*. However, it is easy to repeat calculations for other variants. We will skip details but differences will be highlighted and the final formula for option (b) will be presented next. Comparison of various variants will be done in the forthcoming paper.

2.4.1 Strict hypergraph modularity. In this case, the definition of edge contribution for $A_i \subseteq V$ is:

$$e_H(A_i) = |\{e \in E;\ e \subseteq A_i\}|. \tag{3}$$

The strict modularity of $\mathbf{A}$ on H is then defined as a natural extension of standard modularity in the following way:

$$q_H(\mathbf{A}) = \frac{1}{|E|}\sum_{A_i \in \mathbf{A}} \left(e_H(A_i) - \mathbb{E}_{H' \sim \mathcal{H}}[e_{H'}(A_i)]\right). \tag{4}$$

Consider any $A \subseteq V$. We want to compute the expected edge contribution of A over $\mathcal{H}$. Let $F_d(A) \subseteq F_d$ be the family of multisets of size d with all members in A; that is,

$$F_d(A) \coloneqq \left\{ \{(v_i, m_i) : 1 \le i \le n\} : \sum_{i=1}^{n} m_i = \sum_{i:v_i \in A} m_i = d \right\}.$$

First, note that

$$\begin{aligned} \mathbb{E}_{H' \sim \mathcal{H}}[e_{H'}(A)] &= \sum_{d \ge 2} \sum_{e \in F_d(A)} P_{\mathcal{H}}(e) = \sum_{d \ge 2} |E_d| \sum_{e \in F_d(A)} s_{\mathcal{H}}(e) \\ &= \sum_{d \ge 2} |E_d| \cdot \mathbb{P}\left(\sum_{i:v_i \in A} X_i^{(d)} = d \right) = \sum_{d \ge 2} |E_d| \cdot \left(p_{\mathcal{H}}(A)\right)^d \end{aligned} \tag{5}$$

where $p_{\mathcal{H}}(A) = \sum_{i:v_i \in A} p_{\mathcal{H}}(i)$, therefore $p_{\mathcal{H}}(A) = vol(A)/vol(V)$, so

$$\mathbb{E}_{H' \sim \mathcal{H}}[e_{H'}(A)] = \sum_{d \ge 2} |E_d| \cdot \left(vol(A)/vol(V)\right)^d. \tag{6}$$

Putting (6) properly into (4), we get the strict modularity function of a hypergraph partition:

$$q_H(\mathbf{A}) = \frac{1}{|E|}\left(\sum_{A_i\in\mathbf{A}} e(A_i) - \sum_{d\geq 2}|E_d|\sum_{A_i\in\mathbf{A}}\left(\frac{vol(A_i)}{vol(V)}\right)^d\right). \quad (7)$$

Just as for graphs, the corresponding *modularity* q_H^* is defined as the maximum of $q_H(\mathbf{A})$ over all possible partitions $\mathbf{A}$ of V.

2.4.2 Generalizations. As with graphs, one can easily generalize the modularity function to allow for weighted hyperedges. As we already mentioned, we focused on the strict definition of modularity but it is straightforward to adjust the degree tax to many natural definitions of edge contribution. In particular, for the majority definition (see option (b) at the beginning of this section), one can simply replace $\mathbb{P}(\sum_{i;v_i\in A} X_i^{(d)} = d)$ with $\mathbb{P}(\sum_{i;v_i\in A} X_i^{(d)} > d/2)$ in (5), and so $(vol(A)/vol(V))^d$ in (6) (that is equivalent to $\mathbb{P}(\mathrm{Bin}(d, vol(A)/vol(V)) = d)$ becomes $\mathbb{P}(\mathrm{Bin}(d, vol(A)/vol(V)) > d/2)$. The majority modularity function of a hypergraph partition is then:

$$q_H(\mathbf{A}) = \frac{1}{|E|}\left(\sum_{A_i\in\mathbf{A}} e(A_i) - \sum_{d\geq 2}|E_d|\sum_{A_i\in\mathbf{A}}\mathbb{P}\left(\mathrm{Bin}\left(d, \frac{vol(A_i)}{vol(V)}\right) > d/2\right)\right).$$

We can also consider the modularity independently over hyperedges of different sizes. Decomposing H into d-uniform hypergraphs H_d, we get the following degree-independent modularity function:

$$q_H^{DI}(\mathbf{A}) = \sum_{d\geq 2}\frac{|E_d|}{|E|}q_{H_d}(\mathbf{A}).$$

This corresponds to (7) replacing the volumes computed over H with volumes computed over H_d for each d where $|E_d| > 0$.

3 Searching the solution space

In this section, we show that the solution that maximizes (7) lies in a subset of $\mathcal{P}(V)$ of size at most $2^{|E|}$ avoiding the search of the full set $\mathcal{P}(V)$. Of course, it is still not feasible to check all subsets of the edge set but observations made in this section will be useful for designing efficient heuristic algorithms. For example, in this paper we already used a simple CNM-like algorithm (see Algorithm 1) in which edges overlapping with at least two parts clusters) have to be considered. Using the results from this section, we can significantly reduce the search space by ignoring edges that fall into one part. More sophisticated (and, hopefully, much faster) algorithms utilizing properties investigated in this section will be developed in the forthcoming paper.

Let $\mathcal{S}(H)$ denote the set of all sub-hypergraphs of $H = (V, E)$ on the vertex set V: $\mathcal{S}(H) = \{H' = (V, E') \mid E' \subseteq E\}$. We use $|H'|$ to denote $|E'|$, the number of edges in H'. Moreover, let $p : \mathcal{S}(H) \to \mathcal{P}(V)$ denote the function that sends a sub-hypergraph of H to the partition its connected components induce on V. We define a relation on $\mathcal{S}(H)$:

$$H_1 \equiv_p H_2 \Leftrightarrow p(H_1) = p(H_2)$$

that puts two sub-hypergraphs in relation if they have identical connected components. Since $\equiv_p$ is an equivalence relation (based on equality), we can define the quotient set $\mathcal{S}(H)/_{\equiv_p}$. This quotient set contains equivalence classes that are in bijection with the set of all different vertex

partitions that can be induced by the union of elements of E. Its cardinality depends on E but is at most $2^{|E|}$; however, it is typically much smaller than this trivial upper bound.

Now, let us define the *canonical representative mapping* which identifies a natural representative member for each equivalence class. The canonical representative mapping $f : \mathcal{S}(H)/_{\equiv_p} \rightarrow \mathcal{S}(H)$ maps an equivalence class to the largest member of this class: $f([H']) = H^*$ where $H^* \in [H']$ and $|H^*| \geq |H''|$ for all $H'' \in [H']$. This function is well-defined; indeed, if H_1, $H_2 \in [H']$, then the union of H_1 and H_2 is also in $[H']$ and so it is impossible that two members have the largest size. Its outcome is the subgraph $H^* = (V, E^*)$ whose edge set is the union of edges of all members of the equivalence class. The following lemma explains why the canonical representative is *natural* with respect to the definition of strict modularity. As this observation follows easily from definitions, the proof is omitted.

Lemma 3.1. *Let $H = (V, E)$ be a hypergraph and $\mathbf{A} = \{A_1, \ldots, A_k\}$ be any partition of V. If there exists $H' \in \mathcal{S}(H)$ such that $\mathbf{A} = p(H')$, then the edge contribution of the strict modularity of $\mathbf{A}$ is $\frac{|E^*|}{|E|}$, where E^* is the edge set of the canonical representative of $[H']$.*

The set of canonical representatives, the image of f, is a subset of $\mathcal{S}(H)$. We denote this set by $\mathcal{S}^*(H)$ and the image of p restricted to $\mathcal{S}^*(H)$ by $\mathcal{P}^*(V)$.

The next Lemma shows how the degree tax behaves on partition refinement.

Lemma 3.2. *Let $H = (V, E)$ be a hypergraph and $\mathbf{A}$ be any partition of V. If $\mathbf{B}$ is a refinement of $\mathbf{A}$, then the degree tax of $\mathbf{A}$ is larger than or equal to the degree tax of $\mathbf{B}$ and it is equal if and only if $\mathbf{A} = \mathbf{B}$.*

Proof. Let $\mathbf{A} = \{A_1, \ldots, A_k\}$. Since $\mathbf{B}$ is a refinement of $\mathbf{A}$, for each part (cluster) of $\mathbf{A}$, A_i, there exists $\mathbf{B}_i$, a subset of parts of $\mathbf{B}$, such that $A_i = \bigcup_{B \in \mathbf{B}_i} B$ and $\mathbf{B} = \bigcup_i \mathbf{B}_i$. Hence, for each A_i and for each d, we have that $vol_d(A_i) = \sum_{B \in \mathbf{B}_i} vol_d(B)$ and so

$$vol_d(A_i)^d = \left(\sum_{B \in \mathbf{B}_i} vol_d(B) \right)^d \geq \sum_{B \in \mathbf{B}_i} vol_d(B)^d.$$

The equality holds if and only if $|\mathbf{B}_i| = 1$ for all i. The result follows.

The next result, the main result of this section, shows that one can restrict the search space to canonical representatives from $\mathcal{S}^*(H)$.

Theorem 3.3. *Let $H = (V, E)$ be a hypergraph. If $\mathbf{A} \in \mathcal{P}(V)$ maximizes the strict modularity function $q_H(\cdot)$, then $\mathbf{A} \in \mathcal{P}^*(V)$.*

Proof. Assume that $\mathbf{A} = \{A_1, \ldots, A_k\}$ maximizes the strict modularity function $q_H(\cdot)$. We will show that there exists $H^* = (V, E^*) \in \mathcal{S}^*(H)$ such that $q_H(p(H^*)) \geq q_H(\mathbf{A})$. Let $E^* = \{e \in E: e \subseteq A_i \text{ for some } i\}$. By construction of H^*, the (strict) edge contribution of partitions $\mathbf{A}$ and $p(H^*)$ are identical. Again, from construction, note that the partition $p(H^*)$ is a refinement of $\mathbf{A}$. Hence, the previous Lemma states that the degree tax of $\mathbf{A}$ is larger than or equal to the degree tax of $p(H^*)$. With equal edge contribution, this means that $q_H(p(H^*)) \geq q_H(\mathbf{A})$. Since $\mathbf{A}$ is an optimal solution, the equality must hold which is only possible if $\mathbf{A} = p(H^*)$.

4 Examples

In this section, we first illustrate the correlation between our hypergraph modularity and the Hcut measure, which counts the number of edges touching more than one part (cluster). We then propose an algorithm for hypergraph partitioning based on our hypergraph modularity function, which we apply to a real dataset. However, let us stress again that the aim of this paper is to introduce a generalization of the modularity to hypergraphs, not to introduce new algorithms to actually find good partitions. We currently work on designing and testing

algorithms for the hypergraph counterpart and, after that, we plan to do extensive experiments. The results will be included in the forthcoming paper.

4.1 Synthetic hypergraphs

We generate hyperedges following the process described as *line clustering* in section 5.1 of [33]. We first randomly generate points from 3 lines in the range $[-.5, .5]^2$, 30 points per line, perturbed with Gaussian noise $N(0, \sigma^2)$ where $\sigma = 0.01$. Each line cuts the origin, and the respective slopes are -1, 0.02 and 0.8. We also generate 60 outlying points chosen at random over the same range, for a total of 150 points.

We build hyperedges of size 3 (3-edges) by sampling sets of 3 points $\{i, j, k\}$ for which $\exp(-d(i,j,k)^2/\sigma_d^2) > 0.999$, where $d()$ is the mean distance of the points to their best fitting line, and $\sigma_d = 0.02$. This amounts to selecting 3-edges consisting of sets of 3 well aligned points. We do the same with sets of 4 points to generate the 4-edges.

The hyperedges can either consist of points all coming from the same line (which we call "signal") or not (which we call "noise"). We sample hyperedges so that the expected proportion of signal vs. noise is 2:1, and we consider 3 different regimes for the mix of edge sizes: (i) 75% 3-edges, (ii) 75% 4-edges or (iii) balanced between 3 and 4-edges. For the 3 regimes, we generate 100 hypergraphs and for each hypergraph, we apply the fast Louvain clustering algorithm (see [34]) on the weighted 2-section graph. In most cases, vertices coming from the same line are correctly put in the same part (cluster). The complete source code for this example along with instructions is available online [19].

In the left plot of Fig 1, we plot the standard graph modularity vs. the Hcut value, which is simply the proportion of hyperedges that fall in two or more parts. The Louvain algorithm is not explicitly aiming at preserving the hyperedges, so we do not expect a high correlation between the two measures. In fact, fitting a regression line to the points from the balanced regime, we get a slope of 0.0061 with R^2 value of 0.0008 (for majority of 4-edges the slope is 0.0768 and R^2 0.0734, while for majority of 3-edges the slope is 0.0061 and R^2 0.0004).

In the right plot of Fig 1, we do the same, this time comparing our hypergraph modularity with the Hcut values for the same partitions as in the left plot. The correlation here is very high. For the balanced regime, linear regression yields a slope of -0.6364 with R^2 value of 0.9693 (for majority of 4-edges the slope is -0.7079 and R^2 0.9696, while for majority of 3-edges the slope is -0.7079 and R^2 0.9696). This is an illustration of the fact that when we measure our

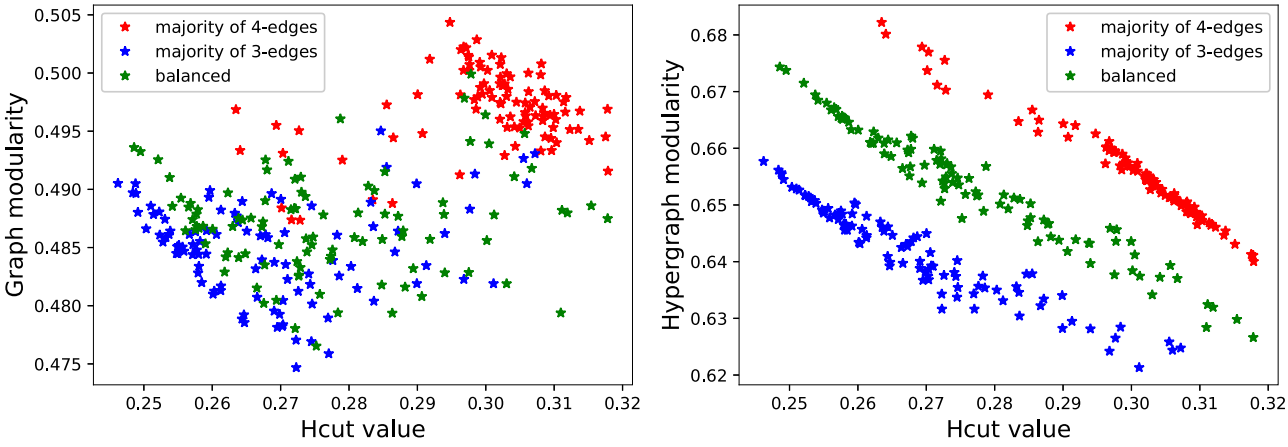

Fig 1. Modularity vs Hcut. Comparing graph and hypergraph modularity.

https://doi.org/10.1371/journal.pone.0224307.g001

proposed hypergraph modularity for different partitions, we are favouring keeping hyperedges in the same parts (clusters).

4.2 Estimating the modularity

While we can compute the modularity for very small hypergraphs by exhausting over all possible partitions, this is generally not a viable option. We propose a generalization to hypergraphs of the CNM algorithm for graph partitioning [26]. In the CNM algorithm, we start with every vertex in its own part. At each step, we merge the two parts (clusters) that yield the largest increase in modularity, and we repeat until no such move exists. The algorithm comes in two versions. In the full version at each step all hyperedges are searched and evaluated for merging. Since for larger hypergraphs this would be prohibitively numerically expensive, in the stochastic version at each step we evaluate just one randomly chosen hyperedge.

Our proposed algorithm for hypergraphs is presented in Algorithm 1. The idea is that we start with a partition where each node is in its own part. Then in each step, we loop through every hyperedge touching two or more parts, and we select (or we just randomly chose a single hyperedge) the one which, when we merge all the parts it touches, yields the best modularity, provided it is at least as high as the modularity from the previous step. We stop when no such edge exists. We use this algorithm in the next example.

Algorithm 1: **Simple CNM-like algorithm on a hypergraph** H

Data: hypergraph $H = (V, E)$
Result: $\mathbf{A}_{opt}$, a partition of V with modularity q_{opt}
Initialize $E^* = E$ and $\mathbf{A}_{opt}$ the partition with all $v \in V$ in its own part with q_{opt} the corresponding modularity;
repeat
  if *(using simplified stochastic algorithm version)* **then**
    $e^* = rand(E^*)$ #randomly select an edge;
    compute the partition A_{e^*} obtained when merging all parts in $\mathbf{A}_{opt}$ touched by e^*, and compute its modularity q_{e^*};
  end
  else
    foreach $e \in E^*$ **do**
      compute the partition A_e obtained when merging all parts in $\mathbf{A}_{opt}$ touched by e, and compute its modularity q_e;
    end
    select edge $e^* \in E^*$ with highest q_{e^*};
  end
  if $q_{e^*} \geq q_{opt}$ **then**
    $A_{opt} = A_{e^*}$ and $q_{opt} = q_{e^*}$;
    update E^*, the set of edges touching two or more parts in $\mathbf{A}_{opt}$;
  end
until ($q_{e^*} < q_{opt}$) ***or*** $E^* = \emptyset$ ***or*** *computational time budget exceeded*;
output: $\mathbf{A}_{opt}$ and q_{opt}

We have implemented the simplified stochastic version of the algorithm in Julia and use it on the hypergraph generated with the forementioned regime (i) that has 75% edges of degree 3 and 25% edges of the degree 4. In our implementation of Algorithm 1, we choose hyperedges to consider at random order (that is, at each step one hyperedge is randomly selected). In order to validate the algorithm, we have run it 1920 times with 500 steps at each run. A single run on a hypergraph consisting of 150 vertices and 5094 hyperedges, using a modern CPU and a single thread, took around 7 seconds. Note that the computational complexity of the proposed algorithm is $O(n)$ and hence it can be practically used for real-world hypergraphs.

The results presented in Fig 2 show that the heuristics presented in Algorithm 1 leads to practically reasonable and useful communities. Firstly, on average, the modularity level after

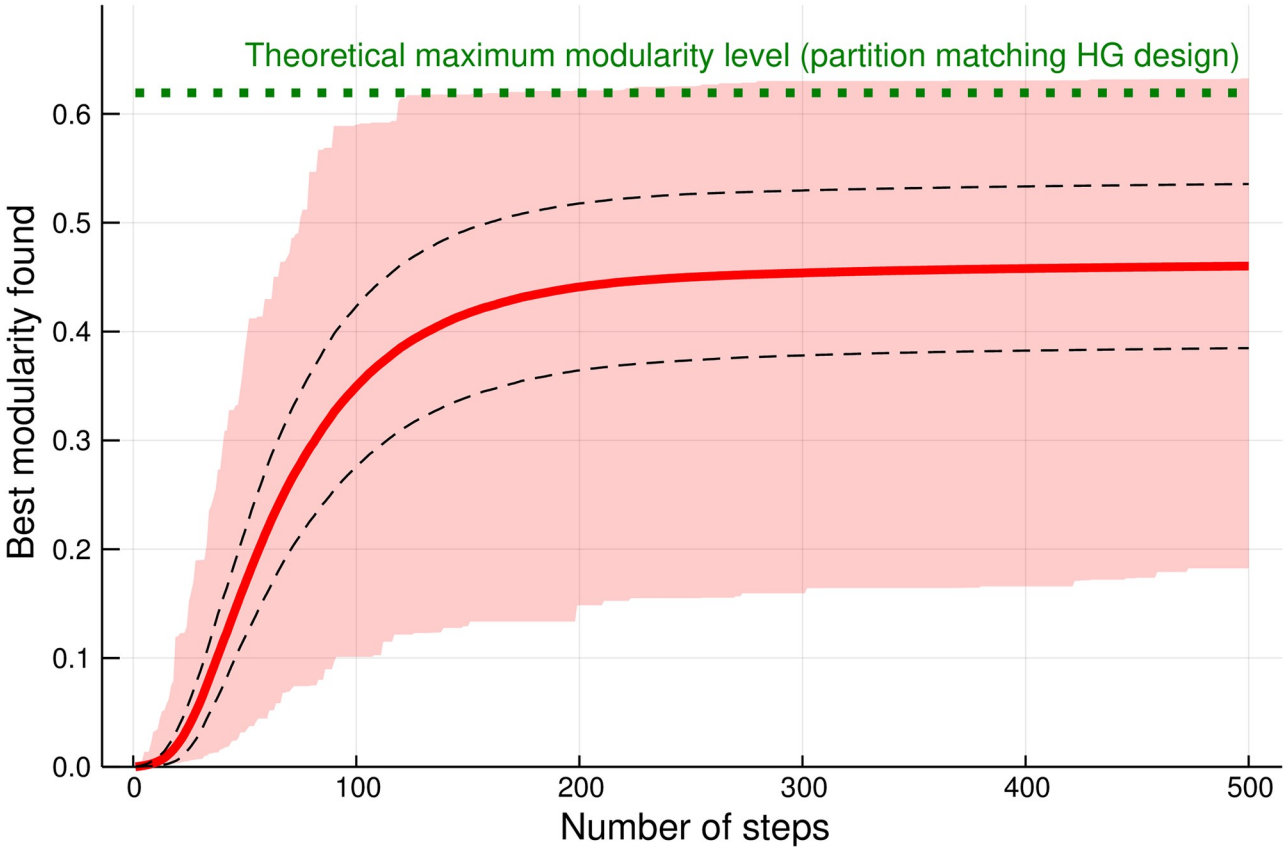

Fig 2. Performance of the Algorithm 1. The thick red line represents the average performance, the dashed line represent standard deviation for the performance and the wide pink area represents the smallest and the biggest performance found in all experiments.

https://doi.org/10.1371/journal.pone.0224307.g002

500 steps is around 75% on modularity for an "optimal" partition of the hypergraph (that is, the partition that uses the knowledge on how the hypergraph was generated). Secondly, this result can be improved by rerunning the Algorithm 1 several times (and this process can be parallelized in high performance computing environments). In particular, we have noticed that for 1.2% of all runs a partition having the optimal modularity (at least 0.6194 for the partition of the partition that matched the process of data generation described in the section 4.1 where we have three groups of points (30 points in group) sharing common hyperedges in each group and an additional 60 points) have been found. Please also note that a higher observed empirical value was observed in some cases—this arises due to the fact that the for last group of synthetic hypergraph having 60 vertices the hyperedges have been generated randomly.

4.3 DBLP hypergraph

The DBLP computer science bibliography database contains open bibliographic information on major computer science journals and proceedings. The DBLP database is operated jointly by University of Trier and Schloss Dagstuhl. The DBLP Paper data is available at http://dblp.uni-trier.de/xml/.

We consider a hypergraph of citations where each vertex represents an author and hyperedges are papers. In order to properly match author names across papers we enhance the data

Table 1. Partitioning DBLP dataset.

algorithm	$q_H()$	$q_G()$	Hcut	#parts
Louvain	0.8613	0.8805	0.1181	40
CNM	0.8671	0.8456	0.0945	92

https://doi.org/10.1371/journal.pone.0224307.t001

Table 2. Proportion of edges of size 2, 3 or 4 cut by the algorithms.

Algorithm	2-edges	3-edges	4-edges
Louvain	0.0382	0.1815	0.3158
CNM	0.0590	0.1277	0.1842

https://doi.org/10.1371/journal.pone.0224307.t002

with information scraped from journal web pages. The DBLP database contains the doi.org identifier. We use this information to obtain the journal name and retrieve paper author data directly from journal—we update available author name data using ACM, IEEE Xplore, Springer and Elsevier/ScienceDirect databases. Since the same author names can be written differently we match author names of the paper across all three data sources. This can give good representation of author names for later matching. For the analysis, we only kept the (single) large connected component. We obtained a hypergraph with 1637 nodes, 865 edges of size 2, 470 of size 3, 152 or size 4 and 37 of size 5 to 7. The complete source code for this example along with instructions and data files is available online [19].

In Table 1, we show our results with the Louvain algorithm on the 2-section graph using modularity function $q_G()$, as well as the results with our CNM algorithm on hypergraphs. Comparing the Louvain and CNM algorithms, we see that there is a tradeoff between q_H and q_G and moreover, the Hcut value is lower with the CNM algorithm. The increased number of parts with our algorithms is mainly due to the presence of singletons.

Another observation is that the actual partitions obtained with objective function $q_G()$ (Louvain) and $q_H()$ (CNM) are different. For the Louvain and CNM algorithms, we found values of 0.4355 for the adjusted Rand index (ARI), 0.4416 for its graph-aware counterpart (see [35]) and 0.684 for the adjusted mutual information, which are commonly used measures of comparisons for partitions. Similar partitions would show values close to 1. We used adjusted measures which are preferable to non-adjusted ones (such as NMI, the normalized mutual information) as they correct for random chance.

One of the difference lies in the number of edges of size 2, 3 and 4 that are cut with the different algorithms, as we see in Table 2. The algorithms based on $q_H()$ will tend to cut less of the larger edges, as compared to the Louvain algorithm, at expense of cutting more size-2 edges.

5 Conclusion

In this paper, we presented a generalization of the Chung-Lu model for hypergraphs, which we used to define a modularity function on hypergraphs. Interestingly, in hypergraph modularity case there is no one unique way to define modularity and we show that it depends on how strongly we think that a hyperedge indicates members of the same community. If the belief is soft this leads to standard 2-section graph modularity. However, if it is strong, a natural definition is strict hypergraph modularity, which we tested on numerical examples. We also proposed the in-between majority-based modularity function.

The objective of this paper is to develop a definition of hypergraph modularity. However, in order to show that this notion is numerically traceable, at least approximately, we provided the

theoretical foundations for the development of algorithms using this modularity function that greatly reduce the solution search space.

A key natural question with any new measure is if it provides qualitatively different outcomes than existing ones. Therefore we have compared strict hypergraph modularity with a standard 2-section graph modularity. For this we proposed a simple heuristic algorithm. We illustrated the fact that in comparison to 2-section graph modularity (optimized using Louvain algorithm) optimization using strict modularity function tends to cut a smaller number of hyperedges. Therefore the proposed measure is potentially highly valuable in application scenarios, where a hyperedge is a strong indicator that vertices it contains belong to the same community.

Hypergraph modularity is a new measure, and there is still a lot of work that should be done. First of all, the development of good, efficient heuristic algorithms would allow to look at larger hypergraphs. Such algorithms would allow us to perform a study over hypergraphs with different edge size distributions, comparing the hypergraph modularity function with other definitions such as graph modularity over the 2-section representation of the hyperedges, and hypergraph modularity using the less strict majority rule.

Finally, let us mention that the method of modularity maximization (in its generalized form which incorporates a resolution parameter controlling the size of the communities discovered) is equivalent to another widely used methods of community detection in networks, the method of maximum likelihood applied to the special case of the stochastic block model known as the planted partition model, in which all communities in a network are assumed to have statistically similar properties [36] (see also [37]). It would be interesting (and potentially useful) to investigate if there is a natural counterpart of our hypergraph modularity measure.

Acknowledgments

The authors would like to thank Claude Gravel for useful discussions while developing the algorithm.

Author Contributions

Conceptualization: Valérie Poulin, Paweł Prałat, François Théberge.

Formal analysis: Valérie Poulin, Paweł Prałat, François Théberge.

Funding acquisition: Bogumił Kamiński, Paweł Prałat, Przemysław Szufel.

Investigation: Bogumił Kamiński, Valérie Poulin, François Théberge.

Methodology: Valérie Poulin, Paweł Prałat, François Théberge.

Project administration: Paweł Prałat.

Software: Bogumił Kamiński, Valérie Poulin, Przemysław Szufel, François Théberge.

Validation: Bogumił Kamiński, Valérie Poulin, Paweł Prałat, Przemysław Szufel, François Théberge.

Writing – original draft: Bogumił Kamiński, Valérie Poulin, Paweł Prałat, François Théberge.

Writing – review & editing: Bogumił Kamiński, Valérie Poulin, Paweł Prałat, François Théberge.

References

1. Fortunato S. Community detection in graphs, Physics Reports. 2010; 486: 75–174. https://doi.org/10.1016/j.physrep.2009.11.002
2. Girvan M, Newman MEJ. Community structure in social and biological networks. Proceedings of the National Academy of Sciences. 2002; 99: 7821–7826. https://doi.org/10.1073/pnas.122653799
3. Zhang Z, Liu C. A hypergraph model of social tagging networks. Journal of Statistical Mechanics: Theory and Experiment, vol. 2010, no 10, 2010. https://doi.org/10.1088/1742-5468/2010/10/P10005
4. Shepherd MA, Watters CR., Cai Y. Transient hypergraphs for citation networks. Information Processing & Management, 26(3), 395–412, 2010. https://doi.org/10.1016/0306-4573(90)90099-N
5. Gallo G, Longo G, Pallottino Stefano, Nguyen S. Directed hypergraphs and applications. Discrete applied mathematics, vol. 42, no 2-3, pp 177–201, 1993. https://doi.org/10.1016/0166-218X(93)90045-P
6. Samaga R, Klamt S. Modeling approaches for qualitative and semi-quantitative analysis of cellular signaling networks. Cell communication and signaling, vol. 11 no 1, 2013. https://doi.org/10.1186/1478-811X-11-43
7. Sarkar S, Sivarajan KN. Hypergraph models for cellular mobile communication systems. IEEE Transactions on Vehicular Technology, 47(2), 460–471, 1998. https://doi.org/10.1109/25.669084
8. Newman MEJ, Girvan M. Finding and evaluating community structure in networks. Phys. Rev. E. 2004; 69: 026–113. https://doi.org/10.1103/PhysRevE.69.066133
9. Chung F, Lu L, Connected components in random graphs with given expected degree sequence. Annals of Combinatorics. 2002; 6: 125–145. https://doi.org/10.1007/PL00012580
10. Zhou D, Huang J, Scholkopf B. Learning with hypergraphs: clustering, classification and embedding. Advances in neural information processing systems. 2006; 19:1601–1608.
11. Shi J, Malik J. Normalized cuts and image segmentation. Pattern Analysis and Machine Intelligence. 2000; 22: 888–905. https://doi.org/10.1109/34.868688
12. Rodriguez JA. On the Laplacian spectrum and walk-regular hypergraphs. Linear and Multi-linear Algebra. 2003; 51: 285–297. https://doi.org/10.1080/0308108031000084374
13. Zien J, Schlag M, Chan P. Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, 1999; 18: 1389–1399. https://doi.org/10.1109/43.784130
14. Agarwal S, Lim J, Zelnik-Manor L, Perona P, Kriegman D, Belongie S. Beyond pairwise clustering. IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2005; 2: 838–845.
15. Agarwal S, Branson K, Belongie S. Higher Order Learning with Graphs, Proc. Int'l Conf. Machine Learning. 2006; 148: 17–24.
16. Voloshim VI, Introduction to Graph and Hypergraph Theory. Nova Kroshka Books; 2013.
17. Karypis G, Kumar V, Multilevel K-Way Hypergraph Partitioning. VLSI Design. 2000; 11: 285–300. https://doi.org/10.1155/2000/19436
18. Chien I, Chung-Yi L, I-Hsiang W. Community detection in hypergraphs: Optimal statistical limit and efficient algorithms. Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics. 2018: 84: 871–879.
19. GitHub gist repository, https://gist.github.com/pszufe/02666497d2c138d1b2de5b7f67784d2b.
20. Chung FRK, Lu L. Complex Graphs and Networks. American Mathematical Society; 2006.
21. Seshadhri C, Kolda TG, Pinar A. Community structure and scale-free collections of Erdös-Rényi graphs. Physical Review E. 2012; 85: 056109. https://doi.org/10.1103/PhysRevE.85.056109
22. Kolda TG, Pinar A, Plantenga T, Seshadhri C. A scalable generative graph model with community structure. SIAM Journal on Scientific Computing. 2014; 36: C424–C452. https://doi.org/10.1137/130914218
23. Winlaw M, DeSterck H, Sanders G. An In-Depth Analysis of the Chung-Lu Model. Lawrence Livermore Technical Report LLNL-TR-678729. 2015.
24. Newman M. Networks: An Introduction. Oxford University Press; 2010.
25. Fortunato S, Barthelemy M. Resolution limit in community detection. Proc. Natl. Acad. Sci. USA. 2007: 104: 36–41. https://doi.org/10.1073/pnas.0605965104 PMID: 17190818
26. Clauset A, Newman MEJ, Moore C. Finding community structure in very large networks. Phys. Rev. E. 2004; 70: 066111. https://doi.org/10.1103/PhysRevE.70.066111
27. Lancichinetti A, Fortunato S. Limits of modularity maximization in community detection. Phys. Rev. E. 2011; 84: 066122. https://doi.org/10.1103/PhysRevE.84.066122
28. Newman MEJ. Fast algorithm for detecting community structure in networks. Phys. Rev. E. 2004; 69: 066133. https://doi.org/10.1103/PhysRevE.69.066133

29. McDiarmid C, Skerman F. Modularity in random regular graphs and lattices. Electronic Notes in Discrete Mathematics. 2013; 43: 431–437. https://doi.org/10.1016/j.endm.2013.07.063

30. McDiarmid C, Skerman F. Modularity of tree-like and random regular graphs. Oxford Journal of Complex Networks. 2018; 6: 596–619. https://doi.org/10.1093/comnet/cnx046

31. Ostroumova Prokhorenkova L, Prałat P, and Raigorodskii A. Modularity of complex networks models. Internet Mathematics. 2017. https://doi.org/10.24166/im.12.2017

32. McDiarmid C, Skerman F. Modularity of Erdos-Renyi random graphs; 2018. Preprint. Available from: https://arxiv.org/abs/1808.02243. Cited 30 April 2019.

33. Leordeanu M, Sminchisescu C. Efficient Hypergraph Clustering. Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics. 2012; 22: 676–684.

34. Blondel V.D., Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment. 2008; 10: P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008

35. Poulin V, Théberge F. Comparing Graph Clusterings: Set partition measures vs. Graph-aware measures. 2018. Preprint. Available from: https://arxiv.org/abs/1806.11494. Cited 30 April 2019.

36. Newman MEJ. Community detection in networks: Modularity optimization and maximum likelihood are equivalent. Phys. Rev. E. 2016; 94: 052315.

37. Lu X, Szymanski BK. Asymptotic resolution bounds of generalized modularity and statistically significant community detection. 2019. Preprint. Available from: https://arxiv.org/abs/1902.04243. Cited 30 April 2019.