Hypergraph **A**rtificial **B**enchmark for **C**ommunity **D**etection (**h–ABCD**)

Bogumił Kamiński* Paweł Prałat† François Théberge‡

July 18, 2023

Abstract

The **A**rtificial **B**enchmark for **C**ommunity **D**etection (**ABCD**) graph is a recently introduced random graph model with community structure and power-law distribution for both degrees and community sizes. The model generates graphs with similar properties as the well-known **LFR** one, and its main parameter ξ can be tuned to mimic its counterpart in the **LFR** model, the mixing parameter μ. In this paper, we introduce **h**ypergraph counterpart of the **ABCD** model, **h–ABCD**, which also produces random hypergraph with distributions of ground-truth community sizes and degrees following power-law. As in the original **ABCD**, the new model **h–ABCD** can produce hypergraphs with various levels of noise. More importantly, the model is flexible and can mimic any desired level of homogeneity of hyperedges that fall into one community. As a result, it can be used as a suitable, synthetic playground for analyzing and tuning hypergraph community detection algorithms.

1 Introduction

Many networks that are currently modelled as graphs would be more accurately modelled as hypergraphs. This includes the collaboration network in which nodes correspond to researchers and hyperedges correspond to papers that consist of nodes associated with researchers that co-authorship a given paper. After many years of intense research using graph theory in modelling and mining complex networks [51, 34, 27, 40], hypergraphs start gaining considerable traction [11, 8, 6, 9]. Standard but important questions in network science are revisited in the context of hypergraphs. This includes various aspects related to community detection in networks [37, 38, 45, 44, 19, 63, 62, 10, 16, 2] which we concentrate on in this paper. However, hypergraphs also create brand new questions which did not have their counterparts for graphs. For example, how hyperedges overlap in empirical hypergraphs [50]? Or how the existing patterns in a hypergraph affect the formation of new hyperedges [35]?

Despite the fact that currently there is a vivid discussion around hypergraphs, the theory and tools are still not sufficiently developed to allow most problems, including clustering, to be tackled

*Decision Analysis and Support Unit, SGH Warsaw School of Economics, Warsaw, Poland; e-mail: bogumil.kaminski@sgh.waw.pl

†Department of Mathematics, Toronto Metropolitan University, Toronto, ON, Canada; e-mail: pralat@torontomu.ca. Part of this work was done while the author was visiting the Simons Institute for the Theory of Computing.

‡Tutte Institute for Mathematics and Computing, Ottawa, ON, Canada; email: theberge@ieee.org

directly within this context. Indeed, researchers and practitioners often create the 2-section graph of a hypergraph of interest (that is, replace each hyperedge with a clique) and apply classical tools designed for graphs. After moving to the 2-section graph, one clearly loses some information about hyperedges of size greater than two and so there is a common belief that one can do better by using the knowledge of the original hypergraph.

As mentioned earlier, there are some recent attempts to directly deal with hypergraphs in the context of clustering. In [64], methods from spectral clustering are generalized to hypergraphs; [22] proposes a framework to infer missing hyperedges and detect overlapping communities, while in [17], extensions of non-backtracking spectral clustering are proposed in the context of hypergraphs. In Kumar et al. [45, 44], the authors reduce the problem to graphs but use original hypergraphs to iteratively adjust weights to encourage some hyperedges to be included in some cluster but discourage other ones (this process can be viewed as separating signal from noise). Moreover, in [37, 38] a number of extensions of the classic null model for graphs are proposed that can potentially be used by hypergraph algorithms.

Unfortunately, there are many ways such extensions can be done depending on how often nodes in one community share hyperedges with nodes from other communities. This is something that varies between networks at hand and usually depends on the hyperedge sizes. Let us come back to the collaboration network we discussed earlier. Hyperedges associated with papers written by mathematicians might be more homogeneous and smaller in comparison with those written by medical doctors who tend to work in large and multidisciplinary teams. Moreover, in general, papers with a large number of co-authors tend to be less homogeneous, and other patterns can be identified [35]. A good clustering algorithm should be able to automatically decide which extension should be used. However, in order to be able to design and properly tune such algorithms, one needs to have a synthetic random hypergraph model that is able to simulate various scenarios.

The hypergraph model, **h–ABCD**, is our response to this need from both practitioners and academia. This random graph model, similarly to the original **ABCD** model, produces hypergraphs with community structure and power-law distribution for both degrees and community sizes. Indeed, as with graphs, power-law distribution is a common feature present in many real-world hypergraphs; for example, [25] shows power-law distributions for several real-world hypergraph decompositions, and we have checked that in the presented cases the hypergraph degree distributions also follow power-law distributions.

The paper is structured as follows. In Section 2, we briefly discuss the history of **ABCD**, the "parent" of the **h–ABCD** model introduced in this paper, before summarizing other synthetic hypergraphs. We focus on models important in the context of community detection and explain the differences between the existing models and our model. The **h–ABCD** model is defined in Section 3. Experiments highlighting important properties of the model are discussed in Section 4. In Section 5, we conclude the paper with a few future research directions.

2 Existing Models

2.1 ABCD graph Models

There are very few datasets with ground-truth identified and labelled. As a result, there is need for synthetic random graph models with community structure that resemble real-world networks in order to benchmark and tune clustering algorithms that are unsupervised by nature. The **LFR**

(**L**ancichinetti, **F**ortunato, **R**adicchi) model [48, 46] generates networks with communities and at the same time it allows for the heterogeneity in the distributions of both node degrees and of community sizes. It became a standard and extensively used method for generating artificial networks.

The **A**rtificial **B**enchmark for **C**ommunity **D**etection (**ABCD**) [39] was recently introduced and implemented*, including a fast implementation† that uses multiple threads (**ABCDe**) [42]. Undirected variant of **LFR** and **ABCD** produce graphs with comparable properties but **ABCD**/**ABCDe** is faster than **LFR** and can be easily tuned to allow the user to make a smooth transition between the two extremes: pure (disjoint) communities and random graph with no community structure. Moreover, it is easier to analyze theoretically—for example, in [36] various theoretical asymptotic properties of the **ABCD** model are investigated including the modularity function that, despite some known issues such as the "resolution limit" reported in [28], is an important graph property of networks in the context of community detection. Finally, the building blocks in the model are flexible and may be adjusted to satisfy different needs. For example, the original **ABCD** model was recently adjusted to include potential outliers in [41] resulting in **ABCD+o** model. Adjusting the model to hypergraphs is much more complex but doable.

2.2 Other Hypergraph Models

The classical configuration model, which was first introduced by Bollobás [13], is a standard model producing graphs with a given degree sequence. It was generalized to higher order structures many times [18, 26]. One early example is the folksonomy, a tripartite structure of users, resources, and tags, that was modelled as hypergraphs generated via configuration-type model in [29]. The variant of the configuration model in [23, 24] generalizes it even further, namely, to simplicial complexes. Simplicial complexes are attractive tools when studying topological aspects of discrete data [15], but subset inclusion is a strong property, not suitable from our application point of view, namely, community detection. In general, configuration models do not pay attention to labels of nodes and so cannot produce graphs with community structure. Having said that, they can be used as an ingredient of models which produce networks with communities. The configuration model was used in the original **ABCD** model and we will use its generalization again for **h–ABCD**.

The Chung-Lu model for graphs (see [20] for details) gives an efficient and simple way to generate graphs with an *expected* degree sequence, but without community structure. It can also be easily generalized for bipartite graphs, as described in [3]; the bipartite representation is often used to model hypergraphs. In the same paper, a generalization of the **BTER** (block two-level Erdős-Rényi) model is proposed, also for bipartite graphs. In the standard BTER model, given degree distribution is preserved as well as the degree-wise clustering coefficients. With bipartite graphs, there are no 3-cycles, so a new *metamorphosis* coefficient is introduced, which is based on 4-cycles, and the model preserves this coefficient in a degree-wise matter.

The stochastic block model (**SBM**), first introduced in [33], is one of the most important graph models in community detection and clustering. One benefit of it is that, being a generative model we can formally study the probability of inferring the ground truth. This is what distinguishes it from the configuration and Chung-Lu models. There are many variants and various applications of the **SBM**. For more details, we direct the reader to [1], one of the many surveys on this model. In particular, one can generalize the **SBM** to hypergraphs as it was done in, for example, [30, 43, 14], while

*`https://github.com/bkamins/ABCDGraphGenerator.jl/`
†`https://github.com/tolcz/ABCDeGraphGenerator.jl/`

in [49], a generalization of **SBM** is proposed for bipartite networks. Most SBM-type models have ground-truth communities embedded into the model but do not produce graphs with realistic degree distributions; a variant of SBM is proposed in [56] to approximate power-law degree distribution but as far as we know, there is no such model for hypergraphs.

In [57], an interesting model is introduced in which hyperedges can be generated via a sampling method, given the number of communities, node memberships, and affinity parameters between the communities. The node degrees and hyperedge sizes can either be given explicitly or sampled from the model. This approach is more complex than the one taken by the family of **ABCD** models. The authors of [57] report that generating sparse hypergraphs with 10^5 nodes takes roughly half an hour, which is orders of magnitude slower than with **h–ABCD**.

In [22], the probabilistic **Hypergraph-MT** model for hypergraphs is presented and is shown to be useful for inferring missing hyperedges as well as detecting overlapping communities. While the model as presented does not explicitly provide a benchmark, in the conclusion of the paper, the authors mention that their model can also be used to sample synthetic data with hypergraph structure.

Other hypergraphs models include core-periphery structure model [55, 59], preferential attachment model with power-law degree distribution and high modularity [31], and entropy-based models [58].

Finally, the authors of [32] propose the **HyGen** model that seems to be similar to the one introduced by us in this paper. Their algorithm is shown to be fast and scalable using MPI standard for its distributed generation. (MPI, message passing interface is a standardized means of exchanging messages between multiple computers running a parallel program across distributed memory.) The **HyGen** model uses the idea of building independently community and background hypergraphs that was also considered in the original **ABCD** graph generator [42] and is used in our **h–ABCD** model. The main difference between the two models is that **HyGen** assumes that hyperedges belonging to communities are completely contained in them, that is, all members of a community hyperedge belong to one community. Our **h–ABCD** algorithm allows for community hyperedges to be only partially contained within a community as long as majority of their members belong to that community. This feature is a significant advantage of **h–ABCD** over **HyGen** as most complex networks have non-strict hyperedges. See Section 3 for more details of community and background hypergraph generation.

2.3 Distinctive Features of h–ABCD

The goal of this research project is to introduce (and efficiently implement) a scalable synthetic hypergraph benchmark model with community structure and power-law degree distribution as well as community sizes. Since none of the existing hypergraph models satisfy all of the desired properties, we moved back to graph models and tried to adjust one of them to our needs.

h–ABCD model that is proposed in this paper produces networks that have power-law degree distributions and distributions of community sizes. Alternatively, the user may easily inject the two distributions as parameters of the model. The user may control the level of noise which covers all spectrum of possibilities of community strength. The model returns the ground-truth communities that then may be used to benchmark and tune community detection algorithms. The model, by design, is fast but it is also efficiently implemented in Julia language. (Julia is a high-level, high-performance, dynamic programming language that recently gained a lot of interest in scientific computing applications [12].) As a result, it is by orders of magnitude faster than other models that we are aware of. Finally, because of its simplicity, its properties as well as various processes

(such as spreading of information, anomalies detection, etc.) can be analyzed theoretically. Such studies, complemented by simulations that are typically easier to perform, often uncover important mechanisms that are present in real-world complex networks. One such spectacular example is a study of the preferential attachment model that explains the following phenomena: "rich-get-richer" mechanisms are responsible for creating power-law distributions that are typically observed in complex networks [4]. Analysis of theoretical properties of **h-ABCD** are left as future work.

3 Definition of the Model

3.1 Parameters of the Model

Let us start with introducing parameters of the model. The **h–ABCD** model is governed by parameters summarized in Table 1. Note, in particular, that the number of hyperedges, m, is not a parameter of the model. It is a random variable that depends on the number of nodes n, the degree distribution of the graph, and the distribution of hyperedge sizes q_d. This random variable is well concentrated around its expectation but it is not a parameter provided as an input. Similarly, the number of communities, ℓ, is not a parameter of the model but a random variable that depends on the number of nodes n and the distribution of community sizes. The process of determining of these values, as well as a detailed explanation of the interpretation of the presented parameters, is discussed further in this section.

parameter	range	description
n	$\mathbb{N}$	number of nodes
γ	$(2,3)$	power-law exponent of degree distribution
δ	$\mathbb{N}$	minimum degree at least δ
D	$\mathbb{N}$	maximum degree at most D
β	$(1,2)$	power-law exponent of distribution of community sizes
s	$\mathbb{N} \setminus [\delta]$	community sizes at least s
S	$\mathbb{N} \setminus [s-1]$	community sizes at most S
ξ	$(0,1)$	level of noise (fraction of non-community hyperedges)
L	$\mathbb{N}$	size of largest hyperedges
q_d	$[0,1]$	fraction of hyperedges that are of size d; $\sum_{d=1}^{L} q_d = 1$
$w_{c,d}$	$[0,1]$	fraction of community hyperedges of size d that have exactly c within-community nodes; $\sum_{c=\lfloor d/2 \rfloor+1}^{d} w_{c,d} = 1$

Table 1: Parameters of the **h–ABCD** model

The model generates a hypergraph on n nodes. The degree distribution follows power-law with exponent γ, minimum and maximum value equal to δ and, respectively, D. Community sizes are between s and S, and also follow power-law distribution, but this time with exponent β. The suggested range of values for parameters γ and β are chosen according to experimental values commonly observed in complex networks not only represented as graphs [5, 54] but also as hypergraphs [25]. Parameter ξ is responsible for the level of noise. If $\xi = 0$, then each hyperedge is a community hyperedge meaning that majority of its nodes belong to one community. On the other extreme, if $\xi = 1$, then communities do not play any roles and hyperedges are simply "sprinkled"

majority $w_{c,d}$

c \ d	1	2	3	4	5
1	1				
2		1	1/2		
3			1/2	1/2	1/3
4				1/2	1/3
5					1/3

linear $w_{c,d}$

c \ d	1	2	3	4	5
1	1				
2		1	2/5		
3			3/5	3/7	3/12
4				4/7	4/12
5					5/12

strict $w_{c,d}$

c \ d	1	2	3	4	5
1	1				
2		1	0		
3			1	0	0
4				1	0
5					1

Table 2: Example values of $w_{c,d}$ for *majority*, *linear*, and *strict* weights for $d \in [5]$.

across the entire hypergraph that we will refer to as background hypergraph. Vector $(q_1, \ldots, q_L)$ determines the distribution of the number of hyperedges of a given size.

Finally, parameters $w_{c,d}$ specify how many nodes from its own community a given community hyperedge should have. We call a community hyperedge to be of a (c, d) type if it has size d and exactly c of its nodes belong to one of the communities. Note, in particular, that we require that a community hyperedge must have more than a half of its nodes from the community. Therefore, $w_{c,d}$ is defined for $d/2 < c \leq d$, where $d \in [L]$.

The model is flexible and may accept any family of parameters $w_{c,d}$ satisfying specific needs of the users, but here is a list of three standard options implemented in the code (see Table 2 for example calculations):

- *majority* model: $w_{c,d}$ is uniform for all admissible values of c, that is, for any $d/2 < c \leq d$,

$$w_{c,d} = \frac{1}{(d - \lfloor d/2 \rfloor)} = \frac{1}{\lceil d/2 \rceil},$$

- *linear* model: $w_{c,d}$ is proportional to c for all admissible values of c, that is, for any $d/2 < c \leq d$,

$$w_{c,d} = \frac{2c}{(d + \lfloor d/2 \rfloor + 1)(d - \lfloor d/2 \rfloor)} = \frac{2c}{(d + \lfloor d/2 \rfloor + 1)\lceil d/2 \rceil},$$

- *strict* model: only "pure" hyperedges are allowed, that is

$$w_{d,d} = 1 \quad \text{and} \quad w_{c,d} = 0 \quad \text{for} \quad d/2 < c < d.$$

In the above formulas and later in the paper, for a given $x \in \mathbb{R}$, $\lfloor x \rfloor$ is used to denote the floor of x (that is, the largest integer not larger than x) and $\lceil x \rceil$ to denote its ceiling (that is, the smallest integer not smaller than x).

3.2 Distribution of Hyperedges Sizes

Clearly, modelling complex networks as hypergraphs is still in an early stage but the initial experiments, such as the ones done on 13 real-world hypergraphs (publication coauthors, drug abuse warning network drugs, emails from an European research institution, national drug code directory drug, online question tags, and thread participants) suggest that in many networks the largest hyperedges are typically of a small size, not comparable to the number of nodes of the hypergraph [25].

As a result, parameter L in **h–ABCD** should be set to be some relatively small value that does not grow with the order of the network.

Having said that, the datasets used in [25] are part of the Benson's collection[‡], which contains real-world hypergraphs with a wide range of different maximum hyperedge sizes. There are datasets with large values of L such as the network of Amazon reviews with $n = 2{,}268{,}231$ and $L = 9{,}350$ or the network of stackoverflow answers with $n = 15{,}211{,}989$ and $L = 61{,}315$.

In this paper, since we aim for a synthetic model for community detection, we concentrate on modelling networks with relatively small hyperedges. Indeed, enormous hyperedges are often regarded as noise in this context and removed during the preprocessing phase. Because of that, our goal is rather ambitious and we aim to generate hypergraphs with a distribution of hyperedges as uniform as possible. Insisting on this important property is inherently computationally expensive for large values of L (see Section 4.6). Dealing with larger values of L needs a slightly different approach but is doable after scarifying uniformity condition (see Conclusions, Section 5).

3.3 The Big Picture

We summarize the main steps followed to generate the hypergraph **h–ABCD** below. More details are provided right after.

1. Sample degrees of nodes.
2. Sample community sizes ensuring that the desired properties hold, in particular, that the sum of community sizes equals the number of nodes n.
3. Compute the number of hyperedges of size one and then generate them.
4. For each node, compute what fraction of its degree should be assigned to community hyperedges and what fraction remains to be used for background hyperedges.
5. Assign nodes to communities ensuring that they fit into them (for example, nodes of very high degree cannot be assigned to small communities, as they would not be able to form simple hyperedges within such communities). One of such admissible assignments is selected randomly.
6. Generate community hypergraphs.
7. Generate the background hypergraph.
8. Resolve potential problems with infeasible hyperedges (either being multisets or duplicates) by executing the rewiring process.

3.4 Definition of the Model—Details

Simple Hypergraphs vs. Multi-hypergraphs

Two variants of the model are considered in this paper, and both of them are implemented and available at the associated GitHub repository[§]. The first variant (that is assumed to be used by

[‡] `https://www.cs.cornell.edu/~arb/data/`

[§] `https://github.com/bkamins/ABCDHypergraphGenerator.jl`

default) produces simple hypergraphs whereas the second one generates multi-hypergraphs. "Simple" in the context of hypergraphs means that hyperedges are sets of nodes but multi-sets are not allowed. Indeed, since all edges in a graph are of size two, loops in the context of graphs are multi-sets of size two in which one node is repeated twice. In particular, there are no multi-sets of size two which correspond to loops in the graph terminology. Similarly, no hyperedges can be repeated so, in particular, there are no parallel edges in the language of graphs (two identical hyperedges of size two in the language of hypergraphs).

Degree Distribution

In the context of hypergraphs, the degree of a node $v \in V$ of a hypergraph $G = (V, E)$ is defined to be the number of hyperedges this node belongs to, regardless of their sizes. (For multi-hypergraphs, the number of occurrences of a node in a hyperedge as well as the number of repetitions of a hyperedge are taken into account.) The volume $\mathrm{vol}(V)$ of G is defined to be the sum of degrees over all nodes in the hypergraph. Hence, if there are m_d hyperedges of size d, then $\mathrm{vol}(V) = \sum_{d=1}^{L} d\, m_d$ giving us the counterpart of the *handshaking lemma* for hypergraphs. Also denote $m = \sum_{d=1}^{L} m_d$ to be a total number of hyperedges in the hypergraph. As noted above, in **h–ABCD**, in contrast to n (the number of nodes), m is a random variable, not a parameter of the model.

The degree distribution of nodes of **h–ABCD** is generated randomly following the (truncated) *power-law distribution* $\mathcal{P}(\gamma, \delta, D)$ with exponent $\gamma \in \mathbb{R}_+$, minimum value $\delta \in \mathbb{N}$, and maximum value $D \in \mathbb{N}$ ($\delta \le D$). Formally, if $X \in \mathcal{P}(\gamma, \delta, D)$, then for any $k \in \{\delta, \delta+1, \dots, D\}$ we have that

$$\Pr(X = k) = \frac{k^{-\gamma}}{\sum_{x=\delta}^{D} x^{-\gamma}}. \tag{1}$$

For typical applications, it is recommended to use $\gamma \in (2, 3)$ [5, 54], some small value of δ such as 5 or 10, and $D \approx n^{\zeta}$ for some $\zeta \in (0, 1)$, where n is the number of nodes.

In order to generate **h–ABCD** hypergraph on n nodes with a given degree distribution $\mathbf{x} := (x_1, \dots, x_n)$, we will use a straightforward generalization of the classical random graph model with a given degree sequence known as the **configuration model** (sometimes called the **pairing model**), which was first introduced by Bollobás [13]. (See [7, 60, 61] for related models and results.) We start with a set P of $\mathrm{vol}(V) = \sum_{i=1}^{n} x_i$ *points* that is partitioned into n *buckets* labelled with labels $v_1, \dots, v_n$; bucket v_i consists of x_i points. We will additionally randomly partition the set P into disjoint sets P_h, $h \in [m]$, of various sizes such that $P = P_1 \cup \ldots \cup P_m$, and construct a multi-hypergraph $\mathcal{P}(\mathbf{x})$ as follows: nodes are the buckets $v_1, \dots, v_n$, and a set P_h of points corresponds to a hyperedge $e_h \subseteq V$ (possibly a multi-set) in $\mathcal{P}(\mathbf{x})$ that consists of nodes (buckets) that contain some point in P_h. In our model, the process of generating random partition $P = P_1 \cup \ldots \cup P_m$ is quite complex and will be done in various phases. At this stage let us only mention that eventually there will be m_d sets/hyperedges of size d for a total of m hyperedges. We will provide more details soon.

Distribution of Community Sizes

Community sizes of **h–ABCD** are generated randomly following the (truncated) *power-law distribution* $\mathcal{P}(\beta, s, S)$ with exponent $\beta \in \mathbb{R}_+$, minimum value $s \in \mathbb{N} \setminus [\delta]$, and maximum value $S \in \mathbb{N}$ ($s \le S$). It is recommended to use $\beta \in (1, 2)$ [5, 54], some relatively small value of s (in our experiments in this paper we set $s = 50$), and $S \approx n^{\tau}$ for some $\zeta < \tau < 1$. The assumption that $\tau > \zeta$ is recommended to make sure large degree nodes have large enough communities to be assigned to.

Similarly, the assumption that $s \geq \delta + 1$ is required to guarantee that small communities are not too small and so that they can accommodate small degree nodes. (These conditions are needed to make sure that generating a simple hypergraph with the desired properties is feasible.)

The procedure of sampling the community size distribution is as follows. First, it is checked if for a given parameters s, S, and n it is possible to generate appropriate community sizes. If it is not possible, then the generation process is stopped and error is returned. Then, the algorithm draws `maxiter` samples from the truncated power-law distribution; in the implementation, `maxiter`=1,000. Each sample is a vector $(c_1, \ldots, c_\ell)$ of community sizes such that $\sum_{j\in[\ell]} c_j \geq n$ and $\sum_{j\in[\ell-1]} c_j < n$. If for any of these samples the corresponding sum is exactly n, then the process is stopped and the obtained community size distribution is retained. If none of the `maxiter` samples yields the sum equal to n, then from the obtained distributions a vector $(c_1, \ldots, c_\ell)$ with minimum sum is selected; note that $\sum_{j\in[\ell]} c_j > n$. Clearly, $\ell > n/S$ but, more importantly, $\ell \leq \lceil n/s \rceil$. The algorithm checks if $\ell > n/s$, and in such (highly unlikely) case we truncate the vector to that length; note that then the sizes add up to less than n. The reason for checking this condition is that it is impossible to generate a distribution having the sum equal to n, length greater than n/s, and with all entries at least s. Finally, we start the process of fixing the community sizes until their sizes add up to n. The updates are made in rounds. In each round all community sizes are shuffled in random order. Then, sequentially, their sizes are increased or decreased by 1, depending if the total sum of sizes is less than or greater than n. To satisfy the desired properties, for a given community an update is made only if it results in a new community size in the range from s to S. The process stops if the sum of community sizes reaches n. If the sum does not reach n in one round, then we repeat the process until the desired property is reached.

Distribution of Hyperedge Sizes and Their Compositions

The distribution of hyperedge sizes is captured by a vector $(q_1, \ldots, q_L)$ with $\sum_{d=1}^{L} q_d = 1$. The value of $q_d \in [0, 1]$ indicates what fraction of the total volume is devoted to form hyperedges of size d. The model distinguishes two types of hyperedges: community hyperedges and background ones. Community hyperedges, by design, will have majority of their members to be part of one community. Because of the majority rule, each community hyperedge is uniquely assigned to one such community. The distribution of the desired composition of hyperedges is reflected by parameters $w_{c,d}$ with $d \in [L]$ and $d/2 < c \leq d$ such that for each $d \in [L]$, $\sum_{c=\lfloor d/2 \rfloor+1}^{d} w_{c,d} = 1$. The value of $w_{c,d} \in [0, 1]$ guides what fraction of community hyperedges of size d have exactly c members from its own community. We will call such hyperedges to be of type (c, d).

Hyperedges of Size 1

Hyperedges of size 1 play a special role and therefore are generated before hyperedges of larger sizes. For example, they are "neutral" for community detection algorithms, such as the classical Louvain algorithm that uses the modularity function, since they are inherently part of communities their unique node belongs to. Having said that, potential users of the **h–ABCD** model might be interested in generating such hyperedges for some other reasons, and so we provide such option.

The number of hyperedges of size 1 is $m_1 = \lfloor q_1 \mathrm{vol}(V) \rceil$, where for a given integer $a \in \mathbb{Z}$ and real

number $b \in [0,1)$ the random variable $\lfloor a+b \rceil$ is defined as

$$\lfloor a+b \rceil = \begin{cases} a & \text{with probability } 1-b \\ a+1 & \text{with probability } b. \end{cases} \tag{2}$$

(Note that $\mathbb{E}[\lfloor a+b \rceil] = a(1-b) + (a+1)b = a+b$.) In the variant of the model generating multi-hypergraphs, each hyperedge selects a node with probability proportional to the number of points in the configuration model associated with this node that are not assigned to any hyperedges yet. In order to generate simple hypergraphs, only points associated with nodes that are not yet assigned to any hyperedges are considered. For this to be feasible, a trivial condition has to be satisfied: $m_1 \leq n$. If this property does not hold, then we truncate it to $m_1 = n$ and, as a result, each node has exactly one hyperedge of size 1 that it is a part of. Finally, regardless whether we generate multi-hypergraphs or simple ones, we update the degree distribution $(x_1, \ldots, x_n)$ to reflect the fact that some nodes are associated with hyperedges of size 1. In other words, in the following subsections it will be convenient to assume that the degree distribution $(x_1, \ldots, x_n)$ ignores hyperedges of size one but only with them the degree distribution matches the one requested by the user.

Level of Noise

As mentioned earlier, there are two types of hyperedges of size at least 2: community hyperedges and background hyperedges. The ratio between the two types is guided by the main parameter of the model: $\xi \in [0,1]$. Indeed, the expected fraction of points (ignoring the ones that are already associated with hyperedges of size 1) that are going to be associated with background hyperedges is equal to ξ. In order to achieve this desired property, we split the degree x_i of each node (equivalently, the associated points in the configuration model) into community degree y_i and background degree z_i ($x_i = y_i + z_i$). We split the weights randomly as follows: for each node, we independently assign $z_i = \lfloor \xi x_i \rceil$ and fix $y_i = x_i - z_i$ (or, equivalently, assign $y_i = \lfloor (1-\xi)x_i \rceil$ and fix $z_i = x_i - y_i$).

Recall that the degree of v is equal to the number of hyperedges v belongs to (in case of non-simple hypergraphs, taking into account both potential repetitions of hyperedges as well as repetitions of nodes within one hyperedge). On the other hand, neighbours of node v are defined to be the nodes that are together with v in some hyperedge. (Alternatively, one may consider the so-called 2-section multi-graph in which each hyperedge of size d is replaced by a complete graph on d nodes from that hyperedge. Then the neighbours of v in the hypergraph are simply neighbours in the associated 2-section graph.) Since hyperedge sizes could be larger than 2, the number of neighbours of v is often larger than the degree of v. In the original **ABCD** model for graphs, parameter ξ guided the fraction of the degree of a node v that is assigned to non-community edges which coincided with the fraction of neighbours of v from community that is different than the community of v. For hypergraphs, in the **h–ABCD** model introduced in this paper, parameter ξ still guides the fraction of the degree of v that is assigned to background hyperedges but it does *not* have a direct interpretation for the number of neighbours of v that belong to different communities than v. For hypergraphs, this fraction will be generally larger than ξ. Indeed, even in the extreme case when $\xi = 0$ (that is, all hyperedges are community hyperedges), hyperedges of size greater than 2 that are assigned to some community will have majority of their members from such community but may not have all of them, depending on the hyperedge type distribution. Unless there are only hyperedges of type (d,d), we still should expect some nodes forming these hyperedges to be from outside of such communities. As a result, not all neighbours of node v will belong to the community of v.

Assigning Nodes into Communities

At this point, the degree distribution ($x_1 \geq x_2 \geq \ldots \geq x_n$) and the distribution of community sizes ($c_1 \geq c_2 \geq \ldots \geq c_\ell$) are already fixed. (In fact, the degree distribution is already split into the community degree distribution (y_i, $i \in [n]$) and the background degree distribution (z_i, $i \in [n]$).) In what follows, as already signalled above, we assume that both the degree sequence (x_i) as well as the sequence of community sizes (c_j) are sorted in a non-increasing order.

h–ABCD hypergraph will be formed as the union of $\ell+1$ independent hypergraphs: ℓ community hypergraphs $G_j = (V, E_j)$ in which $C_j \subseteq V$ with $|C_j| = c_j$ plays a special role ($j \in [\ell]$), and a single background hypergraph $G_0 = (V, E_0)$, where $V = \bigcup_{j\in[\ell]} C_j$. Recall that a hyperedge belongs to a community if majority of its nodes belong to it. By design, all hyperedges of G_j, $j \in [\ell]$, (community hyperedges) will belong to its own community (which does not mean that all members of these hyperedges belong to C_j; that is why $V(G_j) = V$ instead of $V(G_j) = C_j$) but a few additional edges from the background graph G_0 might end up in that community. In order to create enough room for these hyperedges, node of degree x_i needs to be assigned to community C_j of large enough size c_j. The property that needs to be satisfied is as follows: for any $i \in [n]$, node v_i is allowed to be assigned to a community of size c_j if for any $d \in [L] \setminus \{1\}$ and any $\lfloor d/2 \rfloor + 1 \leq c \leq d$,

$$y_i \left(\sum_{f=\lfloor d/2 \rfloor+1}^{c} q_d\, w_{f,d} \cdot \binom{d-f}{c-f} \left(\frac{c_j}{n}\right)^{c-f} \left(1-\frac{c_j}{n}\right)^{d-c} \right) + z_i\, q_d \cdot \binom{d-1}{c-1} \left(\frac{c_j}{n}\right)^{c-1} \left(1-\frac{c_j}{n}\right)^{d-c} \leq \binom{c_j-1}{c-1}\binom{n-c_j}{d-c}. \tag{3}$$

This condition is especially important for the variant which generates simple hypergraphs but we insist on it even when multi-hypergraphs are created to avoid creating hyperedges in which nodes are repeated multiple times. For example, if a hyperedge of size 100 that is of type $(100, 100)$ is assigned to a community of size 10, then some node will have to be repeated at least 10 times.

The rationale behind the above inequality is as follows. There are y_i points associated with community degree of v_i and we expect $q_d w_{f,d}$ fraction of them to be devoted to community hyperedges of type (f, d). Each of these hyperedges has a probability of having precisely c members in community C_j well approximated by

$$\binom{d-f}{c-f} \left(\frac{\mathrm{vol}(C_j)}{\mathrm{vol}(V)}\right)^{c-f} \left(1-\frac{\mathrm{vol}(C_j)}{\mathrm{vol}(V)}\right)^{d-c}.$$

(Unfortunately, to verify this, the reader needs to wait for the description of the processes that generate community and background graphs.) The volume of C_j is not known before the assignment of nodes into communities is finalized but, assuming that nodes are assigned randomly, it is expected that $\mathrm{vol}(C_j) = (c_j/n)\mathrm{vol}(V)$. Similarly, $z_i q_d$ points associated with background degree of v_i is expected to be devoted to background hyperedges of size d. Each of these points has a probability of having precisely c members in community C_j well approximated by

$$\binom{d-1}{c-1} \left(\frac{\mathrm{vol}(C_j)}{\mathrm{vol}(V)}\right)^{c-1} \left(1-\frac{\mathrm{vol}(C_j)}{\mathrm{vol}(V)}\right)^{d-c}.$$

Hence, the left hand side of (3) corresponds to the expected number of hyperedges of type (c, d) from both community C_j and the background graphs. This explains (3) since in simple hypergraphs the

community of size c_j may accommodate at most $\binom{c_j-1}{c-1}\binom{n-c_j}{d-c}$ hyperedges of type (c,d) containing node v_i.

Let us also briefly comment on the complexity aspect of this part of the algorithm, since it is one of the two computationally expensive parts of the algorithm. Note that (3) can be rewritten in the form $y_iA_j + z_iB_j \leq C_j$, where constants A_j, B_j, C_j depend on the distribution of community sizes but does not depend on the degree distribution. As a result, these constants can be pre-computed for all triples (j,d,c), where $j \in [\ell]$, $d \in [L] \setminus \{1\}$, and $\lfloor d/2 \rfloor + 1 \leq c \leq d$, before verifying the inequality. Moreover, for any group of nodes of degree x_i with the same split into y_i and z_i we need to check the condition (3) only once. These observations significantly reduce the running time of the algorithm.

An assignment of nodes into communities will be called admissible if the above family of inequalities (3) is satisfied for all nodes. Our goal is to select one admissible assignment at random, with distribution close to being uniform. Sampling uniformly one of such assignments for graphs turns out to be relatively easy from both theoretical and practical points of view [39, 36]. Indeed, in order to assign nodes to communities, the following easy and natural algorithm is used. We consider nodes, one by one, in non-increasing order of their degrees, that is, we start with v_1 (highest degree node) and finish with v_n (lowest degree node). We assign node v_i randomly to one of the communities that have size large enough so that (3) is satisfied, and still have some "available spots." We do it with probability proportional to the number of available spots left.

It is easy to see that for graphs the above simple algorithm generates one of the admissible assignments uniformly at random. The proof uses the fact that if node v_i can be assigned to a given community, then node $v_{i'}$ for some $i' > i$ can also be assigned to that community. For hypergraphs, each node has to satisfy a few inequalities (3) (for various combinations of c and d) and so this property might not be satisfied. However, this simple algorithm still produces near uniform sampling that is statistically indistinguishable from uniform, provided that n is large enough.

Note that a randomly selected community (with probability proportional to the number of available spots left) often satisfies inequalities (3), especially later in the process when nodes of small degrees are being assigned. Hence, in order to speed up the generation process, we first select 10 random communities and only if none of them satisfies the desired property, we move on to the exhaustive search that identifies potential destinations before selecting one of them randomly.

Creating Community Graphs

In order to generate the community hypergraphs $G_j = (V, E_j)$ we will use a generalization of the configuration model. Recall that there are $p_j = \sum_{v_i \in C_j} y_i$ points associated with nodes in C_j that will be part of community hyperedges. Some of such points will belong to community hyperedges of type (c,d) with $c > d/2$ members from C_j but some of them will form hyperedges with majority of their members from some other community.

First, we need to fix the number of hyperedges of a given size. For each community graph G_j, we consider all values of $d \in [L] \setminus \{1\}$ in decreasing order, and fix the number of hyperedges of size d to be

$$m_d = \left\lfloor \frac{q_d}{\sum_{f=2}^{d} q_f} \left(p_j - \sum_{f=d+1}^{L} f m_f \right) \frac{1}{d} \right\rfloor . \tag{4}$$

(We use the convention that $0/0 = 0$ or simply skip the values of d for which $q_d = 0$.) This guarantees that the distribution of hyperedge sizes follows (approximately) the desired distribution, that is, $dm_d \approx q_d p_j$. Note that, due to some possible divisibility issues, there might be some points left

at the end of this process (at most $d-1$, where d is the smallest value in $[L]\setminus\{1\}$ with $q_d \neq 0$, since we considered values of d in decreasing order). Indeed, in order to avoid potential deficit of points, we rounded real numbers down in the definition of m_d above—see equation (4). These points are simply moved to the background graph, that is, we decrease some random values of y_i (and increase the corresponding values of z_i); the selection of such nodes v_i from community C_j is made with probabilities proportional to y_i. As a consequence, since $p_j = \sum_{v_i \in C_j} y_i$, the value of p_j is also appropriately updated, if needed, and now $p_j = \sum_{d=2}^{L} d \cdot m_d$.

Now, we need to fix the initial number of hyperedges of type (c,d). (The final number of them might be slightly different.) As before, for each community graph G_j and each value of $d \in [L]\setminus\{1\}$, we consider all values of c, $\lfloor d/2 \rfloor + 1 \leq c \leq d$, in decreasing order, and fix the number of hyperedges of type (c,d) to be

$$m_{c,d} = \left\lfloor \frac{w_{c,d}}{\sum_{f=\lfloor d/2 \rfloor+1}^{c} w_{f,d}} \left(m_d - \sum_{f=c+1}^{d} m_{f,d} \right) \right\rfloor .$$

Once all $m_{c,d}$'s are computed, we know that exactly p'_j points out of p_j points associated with nodes in C_j will belong to community hyperedges of type (c,d) with majority of their neighbours from C_j. Clearly,

$$p'_j = \sum_{d=2}^{L} \sum_{c=\lfloor d/2 \rfloor+1}^{d} c \cdot m_{c,d} \quad \leq \quad \sum_{d=2}^{L} \sum_{c=\lfloor d/2 \rfloor+1}^{d} d \cdot m_{c,d} = \sum_{d=2}^{L} d \cdot m_d = p_j .$$

Most of the remaining $p_j - p'_j$ points will typically be assigned to community hyperedges with majority of their neighbours from some other community. Those exceptional points that will end up in community hyperedges with majority of their neighbours from C_j will change their type from (c,d) to (f,d) for some $c < f \leq d$. We will explain how such assignment is done soon.

To make sure each node v_i from C_j is part of many community hyperedges with majority of their members from C_j, we further split y_i points associated with the community degree of v_i and assign y'_i of them to be part of such community hyperedges. The value of $y'_i \leq y_i$ that we aim to be close to $y_i \cdot p'_j/p_j$ is chosen as follows. To guarantee that each y'_i is at least $\lfloor y_i \cdot p'_j/p_j \rfloor$ we initially set this value for every y'_i. Next, we compute by how much the sum of such floors is less than p'_j, that is, we set $t := p'_j - \sum_{v_i \in C_j} \lfloor y_i \cdot p'_j/p_j \rfloor$. Clearly, $0 \leq t < |C_j|$. Then, we randomly sample, without replacement, t points (with probability proportional to $y_i \cdot p'_j/p_j - \lfloor y_i \cdot p'_j/p_j \rfloor$). Such points have their corresponding values of y'_i increased by one that is, $y'_i = \lceil y_i \cdot p'_j/p_j \rceil$. As a result, $\sum_{v_i \in C_j} y'_i = p'_j$ and no value of y'_i is more than 1 away from its desired value of $y_i \cdot p'_j/p_j$.

Let us summarize properties of the auxiliary partition of points that we just created as it is crucial for the process of creating the **h–ABCD** hypergraph. There are $\sum_{i\in[n]} x_i = \mathrm{vol}(V)$ points in total; $p = \sum_{i\in[n]} z_i$ of them are put aside to form a set $\mathcal{B}$ that will be used to create the background graph. From $p_j = \sum_{v_i \in C_j} y_i$ remaining points associated with nodes in C_j we selected $p'_j = \sum_{v_i \in C_j} y'_i$ points to form a set $\mathcal{C}_j$ that will be used to create community hyperedges with majority of members from C_j. The remaining $\sum_{i\in[n]} (y_i - y'_i) = \sum_{j=1}^{\ell} (p_j - p'_j)$ points form a set $\mathcal{C}$.

Finally, we are ready to create random community graphs G_j, $j \in [\ell]$. For each community hyperedge of type (c,d) that belongs to community C_j, we randomly select c points from $\mathcal{C}_j$ (without replacement). At this point all points from $\mathcal{C}_j$ are exhausted. Once we process all community hyperedges of all community graphs, points from $\mathcal{C}$ are randomly assigned to community hyperedges

so that their sizes match the desired values, that is, a hyperedge of type (c, d) gets additional $d - c$ points. As mentioned earlier, note that these points are typically taken from outside of the community a given hyperedge of type (c, d) belongs to, but it might happen that some hyperedge becomes of type (f, d) for some value of f, $c < f \leq d$.

Creating Background Graph

Background graph is created similarly to community graphs. There are $p = \sum_{i \in [n]} z_i$ points in set $\mathcal{B}$ that will be associated with background hyperedges of G_0. As before, we consider all values of $d \in [L] \setminus \{1\}$ in decreasing order, and fix m_d, the number of hyperedges of size d in the background graph, using formula (4).

We first randomly partition points in $\mathcal{B}$ into sets of appropriate sizes that become background hyperedges. Note that after this process, there could be $r \geq 0$ points left, where $0 \leq r < R = \min\{d \in [L] \setminus \{1\} : q_d > 0\}$. (In particular, when edges are allowed, $R = 2$ and so there is at most one point left.). If $r = 0$, then we are done, and so in what follows we discuss how the case $r > 0$ is handled.

First we check whether $q_1 > 0$. If this is the case, then for the multi-hypergraph variant of the model, we simply create r additional hyperedges of size 1 from the remaining r points. On the other hand, if simple hypergraph is requested, then we check whether all remaining points are assigned to unique nodes and none of them corresponds to an already created hyperedge of size 1. If both of these constraints are satisfied, then one more time we create r hyperedges of size 1 from these points.

If $q_1 = 0$, or $q_1 > 0$ and simple hypergraphs were requested but at least one of the above constraints were not satisfied, then we increase the background degrees z_i of $R - r$ random nodes making room for one additional hyperedge of size R (that is, we increase m_R by 1); the selection of such nodes v_i is made with probabilities proportional to z_i.

Finally, let us mention that background hyperedges might produce hyperedges of type (c, d) for some $d/2 < c \leq d$. However, for large graphs with many communities there will typically not be very many of them. Type $(2, 3)$ has the best chance to get additional "boost", followed by type $(2, 2)$.

Creating Simple Hypergraphs—Rewiring

In the variant of the model that produces simple hypergraphs, it remains to deal with multi-sets or hyperedge repetitions that might potentially get created.

We take all hyperedges generated so far and split them into two objects. We put all hyperedges that are sets into set S. All the remaining hyperedges (that is, hyperedges that are multi-sets or hyperedges that were duplicated) are put into vector B. Note that in case of having $t \geq 2$ duplicates of a given hyperedge that is a set, one such hyperedge goes to S and $t - 1$ of them go to B. Next, we try to fix problematic hyperedges from vector B as follows. In one round, we pick a hyperedge b from vector B. We compute the *indisposition* of b as the sum of two terms. The first term is equal to 1 if b is in S, and 0 otherwise. The second term is equal to the number of duplicates in b. Note that exactly one element of this sum is positive.

Next, we randomly pick a good hyperedge g from S. Let s_b be the size of b and s_g be the size of g. We merge points from b and g and randomly split them into new hyperedges, h_1 and h_2, of sizes s_b and s_g. Now, we calculate the sum of indispositions of h_1 and h_2 the same way as it was done for b. If the total indispositions is less than the initial indisposition of b, then we remove b and g from S and, respectively, B, and put h_1 and h_2 back into these sets (to S if they have indispositions

of zero, and to B otherwise). This process is repeated no more than $\texttt{maxiter} \cdot |B|$ times (that is, `maxiter` times per one initial bad hyperedge); in implementation, $\texttt{maxiter} = 100$. If after that many iterations there are still some bad edges left in B, then we give up and the generation process is terminated and appropriate warning is returned.

4 Experiments

In this section we present a series of experiments investigating various properties of the **h–ABCD** model. For such experiments, unless otherwise indicated, we used the following parameters: $\gamma = 2.5$, $\delta = 5$, $D = n^{\zeta}$ with $\zeta = 0.5$ (parameters affecting the degree distribution), $\beta = 1.5$, $s = 50$, $S = n^{\tau}$ with $\tau = 3/4$ (parameters affecting the distribution of community sizes), $\xi = 0.2$ (the level of noise), and $q_2 = q_3 = q_4 = q_5 = 0.25$ (the distribution of hyperedge sizes). In the first few experiments, community edges could have any number of nodes from its own community as long as majority of them belong to that community. In other words, for a fixed value of d, we consider a uniform distribution of $w_{c,d}$. As mentioned at the very beginning of this section, we refer to this distribution as majority model. However, other models will be investigated later on. In all cases, we generate simple hypergraphs.

4.1 Degree Distribution

The degree distribution, by design, follows power-law with exponent γ and from that perspective there is no difference between **h–ABCD** and the original model for graphs, **ABCD**. As a result, we have a good understanding of its asymptotic behaviour [36]. We start with a simple experiment to see whether theoretical, asymptotic results can be used to predict the empirical behaviour for relatively small values of n.

In Figure 1, we plot the fractions of nodes with degree larger than or equal to K for all $K \in \{\delta, \delta + 1, \ldots, D\}$. Results are presented for small ($n = 1,000$) and larger ($n = 1,000,000$) graphs. For each graph size, 100 independent runs were performed and the shaded areas correspond to the standard deviation for each value. We compare those simulation results with the values predicted by theory. We observe a good correspondence even for small graphs, with almost perfect match for larger graphs. Additionally in Table 3, as a reference, we give mean and standard deviation of the number of communities for $n = 2^i$, $i \in \{10, \ldots, 20\}$.

4.2 Distribution of Community Sizes

The process of generating community sizes in **h–ABCD** is the same as in **ABCD**, and it is designed to follow (truncated) power-law distribution with exponent β. Asymptotic behaviour is then quite easy to analyze [36]. However, since there are substantially less communities than nodes, one should not expect to have equally good behaviour of the distribution of community sizes as for the degree distribution.

In Figure 2, we plot the fractions of communities with sizes larger than or equal to K for all $K \in \{s, s + 1, \ldots, S\}$. As in the previous experiments, results are presented for small ($n = 1,000$) and larger ($n = 1,000,000$) graphs. For each graph size, we performed 100 independent runs and compared simulation results with the values predicted by theory. The conclusions are similar but, as expected and mentioned earlier, due to the fact that the number of communities is much smaller

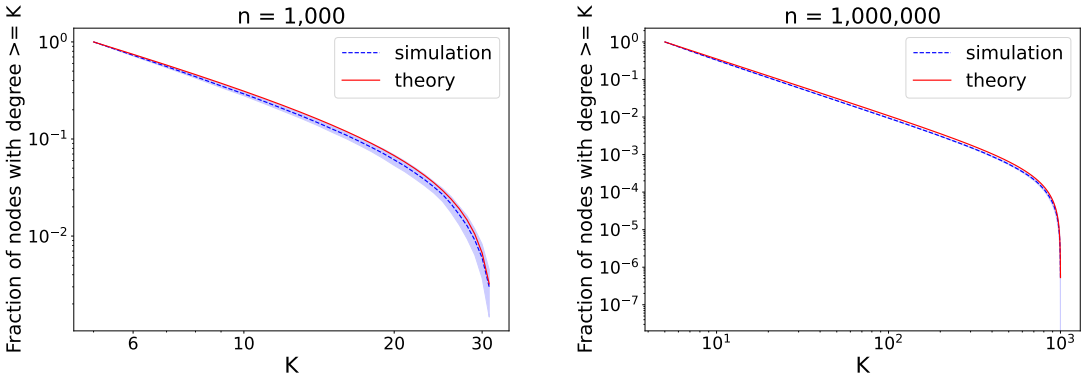

Figure 1: Comparison of the degree distribution predicted by theoretical, asymptotic results with results from simulations.

n	mean ℓ	std ℓ
1,024	11.03	1.29
2,048	16.81	2.23
4,096	26.61	3.52
8,192	39.41	6.01
16,384	62.12	8.39
32,768	96.60	14.92
65,536	148.49	21.74
131,072	229.72	28.61
262,144	352.59	40.51
524,288	539.00	54.11
1,048,576	844.01	82.39

Table 3: Mean and standard deviation of the number of communities ℓ as a function of n. Reported values are computed using 100 independent samples.

than the number of nodes, the standard deviations are substantially larger than those for the degree distribution presented in Figure 1.

4.3 Distribution of Edge Sizes

The distribution of edge sizes is controlled by vector q. In our experiment, we requested that the same fraction of the total volume to be associated with edges of sizes between 2 and 5 ($q_2 = q_3 = q_4 = q_5 = 0.25$). However, because of rounding (see (4)), there is a mild bias towards smaller hyperedges. For large hypergraphs, the difference should not be visible but for small ones could be. The goal of the next experiment is to investigate how strong the bias is.

We independently generated 100 graphs for each size $n = 2^i$, $i \in \{10, \ldots, 20\}$. We considered two different values for the parameter responsible for the level of noise, respectively $\xi = 0.2$ and $\xi = 0.7$. The choice of hyperedge composition (matrix w) does not affect the distribution of edge sizes so we select the majority model, one of the three standard models of **h–ABCD**.

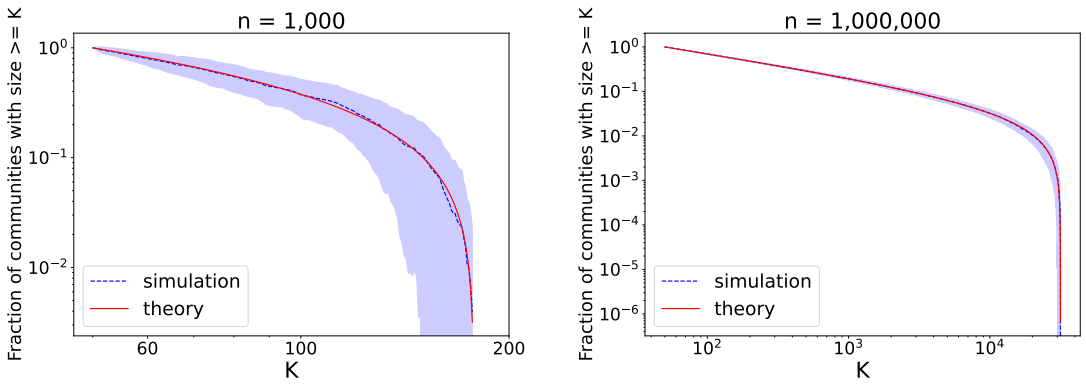

Figure 2: Comparison of the distribution of community sizes predicted by theoretical, asymptotic results with results from simulations.

The results of the experiment are shown in Figure 3. As expected, we see a small bias toward smaller edges for small graphs, and convergence toward a uniform distribution for larger graphs. However, the difference, even for the smallest graphs on 2^{10} nodes, is quite small. Parameter ξ might potentially affect the distribution. To see it, consider hyperedges of the largest size d. For a small value of parameter ξ, almost all communities (if not all) have their volumes much larger than d. Such communities are expected to assign, on average, $(d-1)/2$ less points to the largest hyperedges than requested. For small graphs, the number of communities is not negligible in comparison to the total number of edges, and so we expect to see the difference. On the other hand, if ξ is very close to 1, most of the volume is assigned to the background graph which may affect at most one edge. Hence, the bias should be small for such cases, even for small graphs (of course, large values of ξ are not of practical importance). In our experiments, the results for the two values of ξ are indistinguishable; as expected, ξ affects which nodes are put into hyperedges but not the distribution of hyperedge sizes.

Additionally in Table 4, as a reference, we give mean and standard deviation of the number of generated hyperedges m for $n = 2^i$, $i \in \{10, \ldots, 20\}$, majority model and $\xi = 0.2$ (type of the model and ξ would not affect the results). Note that the number of hyperedges is well concentrated around the mean.

4.4 Distribution of Hyperedge Composition

The setup for this experiment is exactly the same as in the previous subsection, but this time we compare the distribution of hyperedge composition, that is, the number of hyperedges of type (c, d) for $d \in \{2, 3, 4, 5\}$ and $d/2 < c \leq d$. We say that the remaining hyperedges are of type $(0, d)$, that is, hyperedges of type $(0, d)$ are hyperedges of size d that do not have a majority of nodes in any of the communities.

Recall that the initial number of hyperedges of type (c, d) is governed by parameter $w_{c,d}$ stored in matrix w. However, some hyperedges of type (c, d) might "get promoted" and become of type (f, d) for some $c < f \leq d$, if at least one of the additional $d - c$ points are taken from the same community. Moreover, some of the background hyperedges might become of type (c, d) for some $d/2 < c \leq d$. Finally, in the experiments we produce simple hypergraphs (recall that this is a default option but the framework also allows for generation of multi-hypergraphs). The requirement of generating

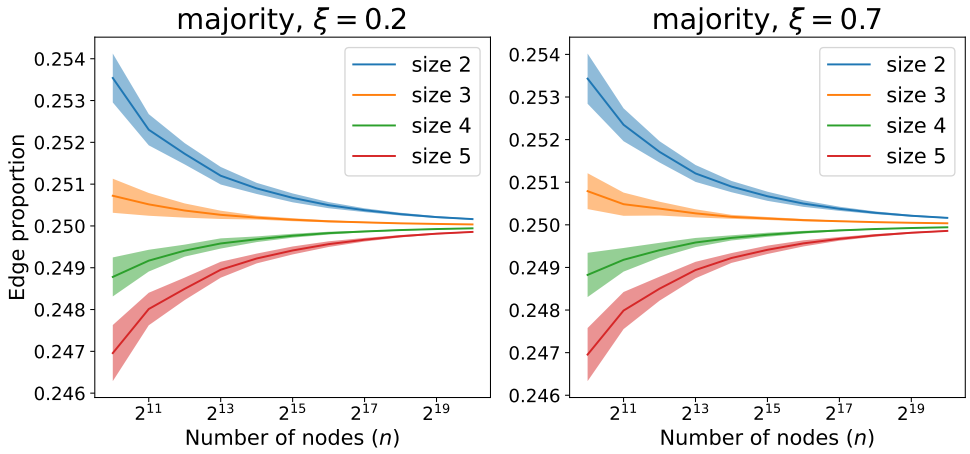

Figure 3: Distribution of the total volume associated with hyperedge sizes, given parameters $q_2 = q_3 = q_4 = q_5 = 0.25$. The majority model and two values of the parameter responsible for the level of noise ($\xi = 0.2, 0.7$) were used.

simple hypergraphs means that when a multi-hyperedge or parallel hyperedge is generated, then it has to be rewired as discussed in Section 3. The rewiring process, if needed, performed on an edge of type (c, d) typically reduces its value of c. This is especially visible for hyperedges of type (d, d) that are most likely to be multi-hyperedges for large d and small communities. For large graphs, the discussed discrepancies should not be noticeable but small graphs might have distribution of hyperedges types slightly different than the desired one, requested via matrix w. The results for two standard models (majority and strict) and $\xi \in \{0.2, 0.7\}$ are shown in Figure 4 (majority) and Figure 5 (strict).

There are several interesting observations that can be made from those plots. Let us first start from the majority model (Figure 4). The requested fraction of hyperedges of size d that are of type $(0, d)$ is ξ, and those that are of type (c, d) for some $d/2 < c \leq d$ should have frequency close to $(1 - \xi)w_{c,d} = (1 - \xi)/\lceil d/2 \rceil$, regardless of the value of c. Large hypergraphs, as expected, have distributions that are very close to the requested one. For smaller graphs we see that some of the background edges got "promoted" and became of type (c, d) for some $d/2 < c \leq d$. As a result, functions associated with types $(0, d)$ for $\xi = 0.7$ tend to increase with the size of the hypergraph. On the other hand, for $\xi = 0.2$ and, say, $d = 5$ the function associated with type $(0, 5)$ seems to be initially larger than 0.2 before converging to 0.2. This is because the total number of community hyperedges of size $d = 5$ is smaller than requested (as discussed in the previous subsection) and so background hyperedges consume a larger share of the total number of hyperedges of size $d = 5$. We also notice that some types of hyperedges are more likely than others, despite the fact that their corresponding values of $w_{c,d}$ are equal. For example, with $d = 3$, we see more hyperedges of type $(2, 3)$ than of type $(3, 3)$ (larger deviation for $\xi = 0.7$ than for $\xi = 0.2$). This is, again, due to hyperedges from the background graph that are more likely to generate such edges by pure randomness.

Similar observations can be made for the *strict* model (Figure 5). As in the previous situation, the requested fraction of hyperedges of size d that are of type $(0, d)$ is ξ. In fact, the initial distribution of hyperedges of type $(0, d)$ is exactly the same as last time (before rewiring), as the processes of

n	mean m	std m
1,024	2,969	48.11
2,048	6,325	92.79
4,096	13,369	185.96
8,192	28,056	311.11
16,384	58,408	492.11
32,768	120,715	956.52
65,536	248,123	1386.05
131,072	508,032	2,201.55
262,144	1,035,753	3,631.05
524,288	2,105,073	5,278.08
1,048,576	4,267,800	7,893.15

Table 4: Mean and standard deviation of the number of hyperedges m as a function of n. Reported values are computed using 100 independent samples.

generating background graphs are the same. However, this time the bias introduced by rewiring is slightly larger than for the majority rule, as hyperedges of type (d, d) are most likely to be non-simple. This discrepancy diminishes as size of the graph increases, as then the fraction of hyperedges that need to be rewired drops. The fraction of hyperedges that are of type (d, d) is requested to be $(1 - \xi)w_{c,d} = 1 - \xi$ but other types should not be present at all. The distribution of hyperedges types is very close to the desired ones for large graphs. For small graphs, while most hyperedges are either of type (d, d) or type $(0, d)$, we see some of them with $d/2 < c < d$, again, a side effect of the background graph. Not surprisingly, there are more of type $(3, 5)$ than of type $(4, 5)$.

4.5 Modularity Function and the Need for Matrix w

As mentioned in the introduction, there are many ways members of one community could form hyperedges. In some real-world networks, most hyperedges are homogeneous (recall an example with papers written by mathematicians) but in some other networks many hyperedges are heterogeneous (again, recall an example with papers written by medial doctors). Hence, **h–ABCD** model has to be able to simulate various scenarios. The main reason behind experiments in this subsection is to show that one may use **h–ABCD** model to generate hypergraphs that are indistinguishable from their 2-section point of view but, at the same time, have completely different structures when viewed as hypergraphs.

Before we move to our experiments, we need to introduce a few definitions. The modularity function for graphs was first introduced by Newman and Girvan in [53] and is currently often used to measure the presence of community structure in networks. Many popular algorithms for partitioning nodes of large graphs use it [21, 47, 52] and perform very well. The modularity function favours partitions of the set of nodes of a graph in which a large proportion of the edges fall entirely within the parts, but benchmarks it against the expected number of edges one would see in those parts in the corresponding Chung-Lu random graph model.

Formally, for a graph $G = (V, E)$ and a given partition $\mathbf{A} = \{A_1, A_2, \ldots, A_k\}$ of V, the *modularity*

function is defined as follows:

$$q_G(\mathbf{A}) \quad = \quad \sum_{A_i \in \mathbf{A}} \frac{e_G(A_i)}{|E|} - \sum_{A_i \in \mathbf{A}} \left(\frac{\mathrm{vol}_G(A_i)}{\mathrm{vol}_G(V)} \right)^2, \tag{5}$$

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* set A_i and $\mathrm{vol}_G(A_i) = \sum_{v_j \in A_i} \deg_G(v_j)$. The first term in (5), $\sum_{A_i \in \mathbf{A}} e_G(A_i)/|E|$, is called the *edge contribution* and it computes the fraction of edges that fall within one of the parts. The second one, $\sum_{A_i \in \mathbf{A}} (\mathrm{vol}_G(A_i)/\mathrm{vol}_G(V))^2$, is called the *degree tax* and it computes the expected fraction of edges that do the same in the corresponding random graph (the null model). The modularity measures the deviation between the two. 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, $q^*(G) = \max_{\mathbf{A}} q_G(\mathbf{A})$.

For edges of size greater than 2, several definitions can be used to quantify the edge contribution for a given partition $\mathbf{A}$ of the set of nodes. As a result, the choice of hypergraph modularity function is not unique; it depends on how strongly one believes that a hyperedge is an indicator that some of its nodes fall into one community. The fraction of nodes of a given hyperedge that belong to one community is called its *homogeneity* and it is assumed that it is more than 50%. As a result, we are guaranteed that hyperedges contribute to at most one part. Once a concrete variant is fixed, one needs to benchmark the corresponding edge contribution using the degree tax computed for the generalization of the Chung-Lu model to hypergraphs proposed in [37].

In [38], various definitions of modularity functions from [37] were put into a common framework, and are available in the `HyperNetX` library[¶]. This general framework is flexible and so can be tuned and applied to hypergraphs with hyperedges of different homogeneity. For each hyperedge size d, we will independently deal with contribution to the modularity function coming from hyperedges of size d with precisely c members from one of the parts, where $c > d/2$. For $d \in \mathbb{N}$ and $p \in [0, 1]$, let $\mathrm{Bin}(d, p)$ denotes the binomial random variable with parameters d and p. Let

$$q_H^{c,d}(\mathbf{A}) = \frac{1}{|E|} \sum_{A_i \in \mathbf{A}} \left(e_H^{d,c}(A_i) - |E_d| \cdot \mathrm{P}\left(\mathrm{Bin}\left(d, \frac{\mathrm{vol}(A_i)}{\mathrm{vol}(V)} \right) = c \right) \right),$$

where $e_H^{d,c}(A_i)$ is the number of hyperedges of size d that have exactly c members in A_i. The hypergraph modularity function is controlled by *hyper-parameters* $u_{c,d} \in [0, 1]$ ($d \geq 2$, $\lfloor d/2 \rfloor + 1 \leq c \leq d$). For a fixed set of hyper-parameters, we simply define

$$q_H(\mathbf{A}) = \sum_{d \geq 2} \sum_{c = \lfloor d/2 \rfloor + 1}^{d} u_{c,d} \; q_H^{c,d}(\mathbf{A}). \tag{6}$$

This definition gives us a lot of flexibility and allows us to value hyperedges of some types (c, d) more than others, depending on their size and level of homogeneity. The choice of these hyper-parameters depends on how strongly we believe that a hyperedge is an indicator that nodes belonging to it fall into one community. In our experiments, we restricted ourselves to three families of parameters $u_{c,d}$, corresponding to the three standard models of $w_{c,d}$ used in the **h–ABCD** model: majority, linear, and strict.

[¶] `https://github.com/pnnl/HyperNetX`

h–ABCD		Modularity			
method	ξ	strict	linear	majority	2-section
strict	0.43	0.533546	0.528192	0.525261	0.501700
linear	0.25	0.514351	0.636436	0.685292	0.504892
majority	0.20	0.508085	0.663819	0.727940	0.502773

Table 5: Three hypergraphs with very similar 2-section modularities but different models for the community hyperedges and different noise values. The differences are evident when looking at the corresponding hypergraph modularities.

Let us now come back to experiments. We generated hypergraphs on $n = 10,000$ nodes and with 50 different choices for the parameter ξ responsible for the level of noise, namely, $\xi \in \{0.01, 0.02, \ldots, 0.50\}$. For each value of ξ, three different models for matrix w were tested: majority and strict, that we already experimented with, but also linear—see the beginning of Section 3 for details. Results are reported in Figure 6.

Before we start discussing particular figures, let us mention that, in general, for the same level of noise, generating strict hyperedges yields higher modularity, while the majority model yields the smallest; the linear lies in-between the two models. The reason is clear. All models generate the same number of community hyperedges but their levels of homogeneity depends on matrix w. Now, the 2-section (graph) modularity replaces each hyperedge with a complete graph and then applies standard graph modularity to the resulted graph. As a result, it favours hyperedges that are more homogeneous since such hyperedges generate more edges within communities in the corresponding 2-section graph and so they contribute more to the modularity. Similarly, each of the three hypergraph modularities value hyperedges that are more homogeneous at least as much as less homogeneous ones, that is, the corresponding parameters $u_{c,d}$ are non-decreasing. In fact, all of them but the majority modularity are strictly increasing. As a consequence, for these modularity functions (see Figure 6 top-right and bottom-right), the values for the strict model are larger than for the ones for the linear model which in turn are larger than the values for the majority model. Finally, note that for the majority modularity (see Figure 6, bottom-left), the composition of the community edges does not matter; regardless of which model is used, all community hyperedges contribute equally to the modularity function and so the modularity functions are the same. Another observation is that generating community hyperedges with majority or linear methods yield more similar hypergraphs, while the strict model is visibly distinct.

For each plot in Figure 6, we added a dashed line at the modularity value 0.5. This is to show that with the exception of the majority modularity, different values of the noise parameter ξ are required for different models to obtain the same modularity. It is clear and expected but it is important when the **h–ABCD** model is used to benchmark the performance of clustering algorithms so that "apples and oranges" are not compared against each other. For example, one can easily generate hypergraphs with similar 2-section modularities using different models for the community hyperedges (strict, majority, linear) and different noise parameters. Such hypergraphs would seem very similar when looking at their two-section graphs, but are really quite different hypergraphs. We show some specific numbers illustrating this phenomena in Table 5.

4.6 Time Complexity of the Algorithm

The framework to generate "**ABCD**-type" models is very flexible; the original **ABCD** model was already parallelized (**ABCDe**) and generalized to include outliers (**ABCD+o**). In this paper, we adjust it to hypergraphs (**h–ABCD**). Another important feature of these models is that they are also, by design, very fast. This is in contrast to the main problem with high order structures—there are $\binom{n}{d}$ potential hyperedges of size d which is strikingly larger than the number of potential edges, $\binom{n}{2}$. As a result, hypergraph synthetic graph models are inherently slow but there are some exceptions. For example, the experiments presented in [57] show that their "model is highly efficient, as it allows to sample sparse hypergraphs of dimensions up to 10^5 nodes in less than one hour". Our model generates graphs of order ten times larger in less than 10 seconds. Similarly, [32] reports that **HyGen** takes approximately four minutes to generate a hypergraph with 4.8 million vertices, 1.6 million hyperedges, and 800 clusters using 1,024 processes on a leadership class computing platform. **h–ABCD** is able to generate hypergraphs of similar size under one minute on a single core of a desktop computer. This shows a drastic advantage of the used framework, even in comparison to other scalable approaches.

The two most computationally expensive parts of the algorithm are: (1) assignment of nodes to communities, and (2) hyperedge generation (the remaining are sampling node degrees and sampling community sizes and their time only depends on n).

The cost of assignment of nodes to communities is $O(n + \lambda\ell L^2)$, where $\lambda \leq n$ is the number of unique (y_i, z_i) combinations that appear in the inequality (3). Note that in power-law graphs there are many nodes of the same degree and so that although λ grows with n it is in practice much smaller than it. Similarly, ℓ grows with n, but is of order of magnitude smaller than it; see [36] for asymptotic analysis of distribution of degree and community sizes.

The cost of hyperedge generation is $O(\mathrm{vol}(V) + \ell L^2)$, where ℓL^2 is the cost of preprocessing which has to be done for each community for each (c, d) combination. Again using the results from [36] we know that $\mathrm{vol}(V) = O(n)$, and thus we can expect that actual hyperedge generation should be the most expensive part of the algorithm for large values of n, provided that L is a fixed constant.

The performance benchmarks results are presented in Table 6 to show the scalability of **h-ABCD** in practice for the case of small values of L. In particular, the timings reflect not only asymptotic complexity of the algorithm but also impact on timing of implementation details like: CPU cache locality and memory allocation management cots, as we want to be sure that they do not significantly impact it.

In these experiments, the level of noise was set to $\xi = 0.2$, uniform distribution of hyperedge sizes was used (that is, $q_d = 1/(L-1)$ for any $d \in [L] \setminus \{1\}$) together with the majority model. Total time we report, apart from assignment to communities and hyperedge generation includes time to generate node degrees and community sizes (these two times depend only on n).

The results of the experiments confirm that hyperedge generation is the most significant component of the hypergraph generation process for the case when L is small compared to n. As expected, the number of nodes n has a major impact on total runtime of the algorithm, but also increasing L influences it as is predicted by the asymptotic formulas. In general, the proposed algorithm allows to generate hypergraphs having one million nodes in several seconds in the considered cases.

Additionally, in Table 7 we check if the time complexity of the algorithm indeed is proportional to L^2. For this test we fix $n = 10^6$ and check values of L up to 320, keeping the distribution of hyperedge sizes uniform, that is, $q_d = 1/(L-1)$ for $d > 1$. For this test we set the larger minimum

n	L	Assignment to communities	Hyperedge generation	Total time*
500,000	5	0.12	1.67	2.09
500,000	10	0.18	1.71	2.19
500,000	20	0.48	1.77	2.55
1,000,000	5	0.28	3.50	4.18
1,000,000	10	0.40	3.64	4.44
1,000,000	20	1.09	3.73	5.22
2,000,000	5	0.70	9.77	11.48
2,000,000	10	0.87	9.88	11.74
2,000,000	20	1.97	13.81	15.76

* Total time also includes time to generate node degrees and community sizes (these two times depend only on n).

Table 6: Generation time (in seconds) of **h-ABCD** hypergraph for different values of n and L.

L	Assignment to communities	Hyperedge generation	Total time*
20	0.34	3.39	2.76
40	1.07	2.18	3.28
80	8.46	3.43	11.92
160	46.73	5.62	52.37
320	252.78	13.19	266.01

* Total time also includes time to generate node degrees and community sizes.

Table 7: Generation time (in seconds) of **h-ABCD** hypergraph for different values of L, with $n = 10^6$ and minimum community size fixed to 10,000.

community size to $s = 10,000$. The reason for this change is that communities must be large enough so that hyperedges can fit into the community graphs. The tests show that, indeed, the assignment to communities component becomes the most time consuming as L grows and the relationship is quadratic as predicted by theoretical considerations.

5 Conclusions

In this paper we introduced **h–ABCD**, one of the very first synthetic random hypergraph models with community structure. This model produces "**LFR**-type" hypergraphs but its building blocks are inherited from the **ABCD** model. Because of that, one can easily generate synthetic hypergraphs with the degree distribution as well as the distribution of community sizes following power-law. The generation process is fast and the ground truth partition of nodes can be easily used to benchmark hypergraph community detection algorithms. The benchmark is available on GitHub.

Modelling and mining complex networks as hypergraphs is an exciting and brand new research direction. There are many open problems left to be investigated. We plan to work on the following questions next.

- We plan to design and implement hypergraph clustering algorithm. One approach that is worth

considering is to optimize the hypergraph modularity function introduced in [37]; see [38] for encouraging initial results. In any case, **h–ABCD** model will be instrumental in validating and benchmarking various ideas during this process.

- We plan to analyze typical, asymptotic properties of **h–ABCD** model, especially the behaviour of the hypergraph modularity function. Similar study is already done for the original **ABCD** model in [36] but not all questions are answered.

- Generating hypergraphs is more challenging and time consuming than generating graphs. **h–ABCD** model can generate hypergraphs having 1 million nodes and the maximum hyperedge size 5 on an average laptop in several seconds. It means that the time complexity is similar to the original **ABCD** graph generator for comparable graph sizes, but significantly faster than the **LFR** graph generator and much faster than other hypergraph competitors. Since there are still no scalable algorithms for hypergraphs that can handle huge networks, there is no need for parallel and scalable implementations of **h–ABCD** yet, but the situation might change in the near future. We plan to implement a multi-threaded version of the model as it was done for the original **ABCDe** model in [42].

- The framework used to generate **ABCD** model is flexible. It was modified to include outliers (**ABCD+o**) and in this paper we modified it to generate hypergraphs (**h–ABCD**). Since detecting overlapping communities is an important problem and there are few synthetic benchmarks for this task, it would be good to modify the framework one more time and add option for overlapping communities to our "tool-box".

- **h–ABCD** model allows for any value of L, the maximum size of hyperedges, but it is designed with relatively small values of L in mind such as $L = 10$ or $L = 20$. Such hypergraphs are typically of interest in the context of community detection algorithms. However, if one wants to create hypergraphs with much larger values of L, then the algorithm will slow down. Indeed, "collisions" when generating large hyperedges will occur with larger probability. In order to handle such large hyperedges, the algorithm needs to be slightly adjusted. The drawback, and the reason why we do not do it currently, is that the distribution of hyperedges would deviate from uniform distribution.

Acknowledgements

PP and BK have been supported by the Polish National Agency for Academic Exchange under the Strategic Partnerships programme, grant number BPI/PST/2021/1/00069/U/00001.

References

[1] Emmanuel Abbe. Community detection and stochastic block models: recent developments. *The Journal of Machine Learning Research*, 18(1):6446–6531, 2017.

[2] Kwangjun Ahn, Kangwook Lee, and Changho Suh. Hypergraph spectral clustering in the weighted stochastic block model. *IEEE Journal of Selected Topics in Signal Processing*, 12(5):959–974, 2018.

[3] Sinan G. Aksoy, Tamara G. Kolda, and Ali Pinar. Measuring and modeling bipartite graphs with community structure. *Journal of Complex Networks*, 5(4):581–603, 03 2017. doi:10.1093/comnet/cnx001.

[4] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[5] Albert-László Barabási. *Network science*. Cambridge University Press, 2016. URL: http://barabasi.com/networksciencebook/.

[6] Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania, Jean-Gabriel Young, and Giovanni Petri. Networks beyond pairwise interactions: structure and dynamics. *Physics Reports*, 874:1–92, 2020.

[7] Edward A Bender and E Rodney Canfield. The asymptotic number of labeled graphs with given degree sequences. *Journal of Combinatorial Theory, Series A*, 24(3):296–307, 1978.

[8] Austin R Benson, Rediet Abebe, Michael T Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 115(48):E11221–E11230, 2018.

[9] Austin R Benson, David F Gleich, and Desmond J Higham. Higher-order network analysis takes off, fueled by classical ideas and new data. *arXiv preprint arXiv:2103.05031*, 2021.

[10] Austin R Benson, David F Gleich, and Jure Leskovec. Tensor spectral clustering for partitioning higher-order network structures. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 118–126. SIAM, 2015.

[11] Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

[12] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[13] Béla Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1(4):311–316, 1980.

[14] Luca Brusa and Catherine Matias. Model-based clustering in simple hypergraphs through a stochastic blockmodel. *arXiv preprint arXiv:2210.05983*, 2022.

[15] Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

[16] I Chien, Chung-Yi Lin, and I-Hsiang Wang. Community detection in hypergraphs: Optimal statistical limit and efficient algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 871–879. PMLR, 2018.

[17] Philip Chodrow, Nicole Eikmeier, and Jamie Haddock. Nonbacktracking spectral clustering of nonuniform hypergraphs. *arXiv preprint arXiv:2204.13586*, 2022.

[18] Philip S Chodrow. Configuration models of random hypergraphs. *Journal of Complex Networks*, 8(3):cnaa018, 2020.

[19] Philip S Chodrow, Nate Veldt, and Austin R Benson. Generative hypergraph clustering: From blockmodels to modularity. *Science Advances*, 7(28):eabh1303, 2021.

[20] Fan Chung Graham and Linyuan Lu. *Complex graphs and networks*. Number 107. American Mathematical Soc., 2006.

[21] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[22] Martina Contisciani, Federico Battiston, and Caterina De Bacco. Inference of hyperedges and overlapping communities in hypergraphs. *Nature Communications*, 13(1):7229, 2022.

[23] Owen T Courtney and Ginestra Bianconi. Generalized network structures: The configuration model and the canonical ensemble of simplicial complexes. *Physical Review E*, 93(6):062311, 2016.

[24] Owen T Courtney and Ginestra Bianconi. Weighted growing simplicial complexes. *Physical Review E*, 95(6):062301, 2017.

[25] Manh Tuan Do, Se-eun Yoon, Bryan Hooi, and Kijung Shin. Structural patterns and generative models of real-world hypergraphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 176–186, 2020.

[26] Martin Dyer, Catherine Greenhill, Pieter Kleer, James Ross, and Leen Stougie. Sampling hypergraphs with given degrees. *Discrete Mathematics*, 344(11):112566, 2021.

[27] David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge university press, 2010.

[28] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the national academy of sciences*, 104(1):36–41, 2007.

[29] Gourab Ghoshal, Vinko Zlatić, Guido Caldarelli, and Mark EJ Newman. Random hypergraphs and their applications. *Physical Review E*, 79(6):066118, 2009.

[30] Debarghya Ghoshdastidar and Ambedkar Dukkipati. Consistency of spectral hypergraph partitioning under planted partition model. *The Annals of Statistics*, 45(1):289–315, 2017.

[31] Frédéric Giroire, Nicolas Nisse, Thibaud Trolliet, and Malgorzata Sulkowska. Preferential attachment hypergraph with high modularity. *[Research Report] Université Cote d'Azur*, hal-03154836, 2021.

[32] S M Shamimul Hasan, Neena Imam, and Ramakrishnan Kannan. A scalable parallel hypergraph generator (hygen). 8 2020. URL: `https://www.osti.gov/biblio/1651312`.

[33] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[34] Matthew O Jackson. *Social and economic networks*. Princeton university press, 2010.

[35] Jonas L Juul, Austin R Benson, and Jon Kleinberg. Hypergraph patterns and collaboration structure. *arXiv preprint arXiv:2210.02163*, 2022.

[36] Bogumił Kamiński, Bartosz Pankratz, Paweł Prałat, and François Théberge. Modularity of the abcd random graph model with community structure. *Journal of Complex Networks*, 10(6):cnac050, 2022.

[37] Bogumił Kamiński, Valérie Poulin, Paweł Prałat, Przemysław Szufel, and François Théberge. Clustering via hypergraph modularity. *PloS one*, 14(11):e0224307, 2019.

[38] Bogumił Kamiński, Paweł Prałat, and François Théberge. Community detection algorithm using hypergraph modularity. In *International Conference on Complex Networks and Their Applications*, pages 152–163. Springer, 2020.

[39] Bogumił Kamiński, Paweł Prałat, and François Théberge. Artificial benchmark for community detection (abcd)—fast random graph model with community structure. *Network Science*, pages 1–26, 2021.

[40] Bogumił Kamiński, Paweł Prałat, and François Théberge. Mining complex networks. 2021.

[41] Bogumił Kamiński, Paweł Prałat, and François Théberge. Outliers in the abcd random graph model with community structure (abcd+o). In *Proceedings of the 11th International Conference on Complex Networks and their Applications*, 2022 (in press).

[42] Bogumił Kamiński, Tomasz Olczak, Bartosz Pankratz, Paweł Prałat, and François Théberge. Properties and performance of the abcde random graph model with community structure. *Big Data Research*, 30:100348, 2022. URL: `https://www.sciencedirect.com/science/article/pii/S2214579622000429`, `doi:https://doi.org/10.1016/j.bdr.2022.100348`.

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

[44] Tarun Kumar, Sankaran Vaidyanathan, Harini Ananthapadmanabhan, Srinivasan Parthasarathy, and Balaraman Ravindran. Hypergraph clustering by iteratively reweighted modularity maximization. *Applied Network Science*, 5(52), 2020.

[45] Tarun Kumar, Sankaran Vaidyanathan, Harini Ananthapadmanabhan, Srinivasan Parthasarathy, and Balaraman Ravindran. A new measure of modularity in hypergraphs: Theoretical insights and implications for effective clustering. In Hocine Cherifi, Sabrina Gaito, José Fernendo Mendes, Esteban Moro, and Luis Mateus Rocha, editors, *Complex Networks and Their Applications VIII*, pages 286–297, Cham, 2020. Springer International Publishing.

[46] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):016118, 2009.

[47] Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Physical review E*, 84(6):066122, 2011.

[48] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.

[49] Daniel B. Larremore, Aaron Clauset, and Abigail Z. Jacobs. Efficiently inferring community structure in bipartite networks. *Phys. Rev. E*, 90:012805, Jul 2014. doi:10.1103/PhysRevE.90.012805.

[50] Geon Lee, Minyoung Choe, and Kijung Shin. How do hyperedges overlap in real-world hypergraphs?-patterns, measures, and generators. In *Proceedings of the Web Conference 2021*, pages 3396–3407, 2021.

[51] Mark Newman. *Networks*. Oxford university press, 2018.

[52] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.

[53] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[54] Günce Keziban Orman and Vincent Labatut. A comparison of community detection algorithms on artificial networks. In *Discovery Science: 12th International Conference, DS 2009, Porto, Portugal, October 3-5, 2009 12*, pages 242–256. Springer, 2009.

[55] Marios Papachristou and Jon Kleinberg. Core-periphery models for hypergraphs. KDD '22, page 1337–1347. Association for Computing Machinery, 2022. doi:10.1145/3534678.3539272.

[56] Maoying Qiao, Jun Yu, Wei Bian, Qiang Li, and Dacheng Tao. Adapting stochastic block models to power-law degree distributions. *IEEE transactions on cybernetics*, 49(2):626–637, 2018.

[57] Nicolò Ruggeri, Federico Battiston, and Caterina De Bacco. A principled, flexible and efficient framework for hypergraph benchmarking. *arXiv preprint arXiv:2212.08593*, 2022.

[58] Fabio Saracco, Giovanni Petri, Renaud Lambiotte, and Tiziano Squartini. Entropy-based random models for hypergraphs, 2022. doi:10.48550/ARXIV.2207.12123.

[59] Francesco Tudisco and Desmond J Higham. Core-periphery detection in hypergraphs. *SIAM Journal on Mathematics of Data Science*, 5(1):1–21, 2023.

[60] Nicholas C Wormald. Generating random regular graphs. *Journal of algorithms*, 5(2):247–280, 1984.

[61] Nicholas C Wormald et al. Models of random regular graphs. *London Mathematical Society Lecture Note Series*, pages 239–298, 1999.

[62] Hao Yin, Austin R Benson, and Jure Leskovec. Higher-order clustering in networks. *Physical Review E*, 97(5):052306, 2018.

[63] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 555–564, 2017.

[64] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. *Advances in neural information processing systems*, 19, 2006.

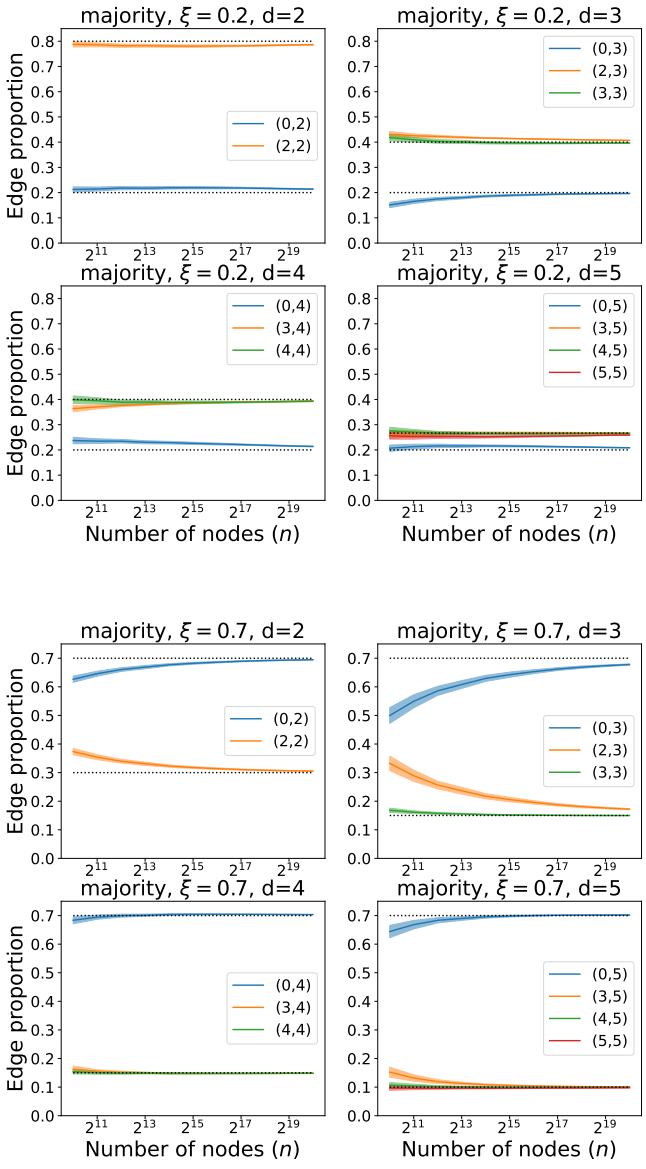

Figure 4: Distribution of type (c, d) hyperedges for $d \in \{2, 3, 4, 5\}$: *majority* model with $\xi = 0.2$ (top 4 figures) and $\xi = 0.7$ (bottom 4 figures).

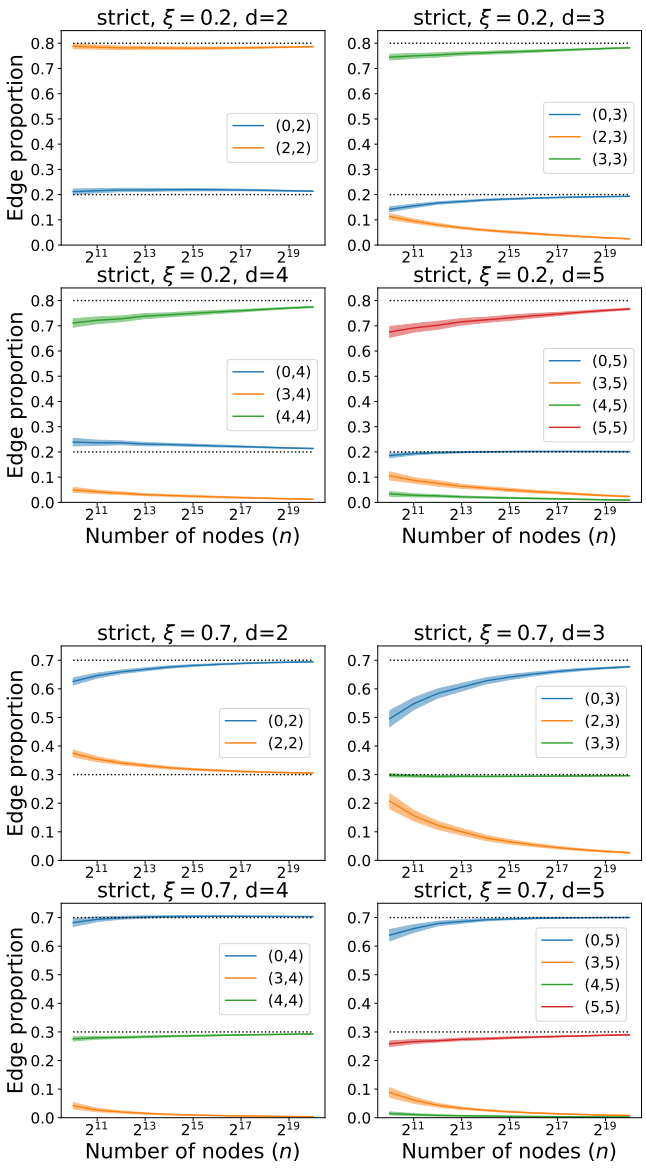

Figure 5: Distribution of type (c, d) hyperedges for $d \in \{2, 3, 4, 5\}$: *strict* model with $\xi = 0.2$ (top 4 figures) and $\xi = 0.7$ (bottom 4 figures).

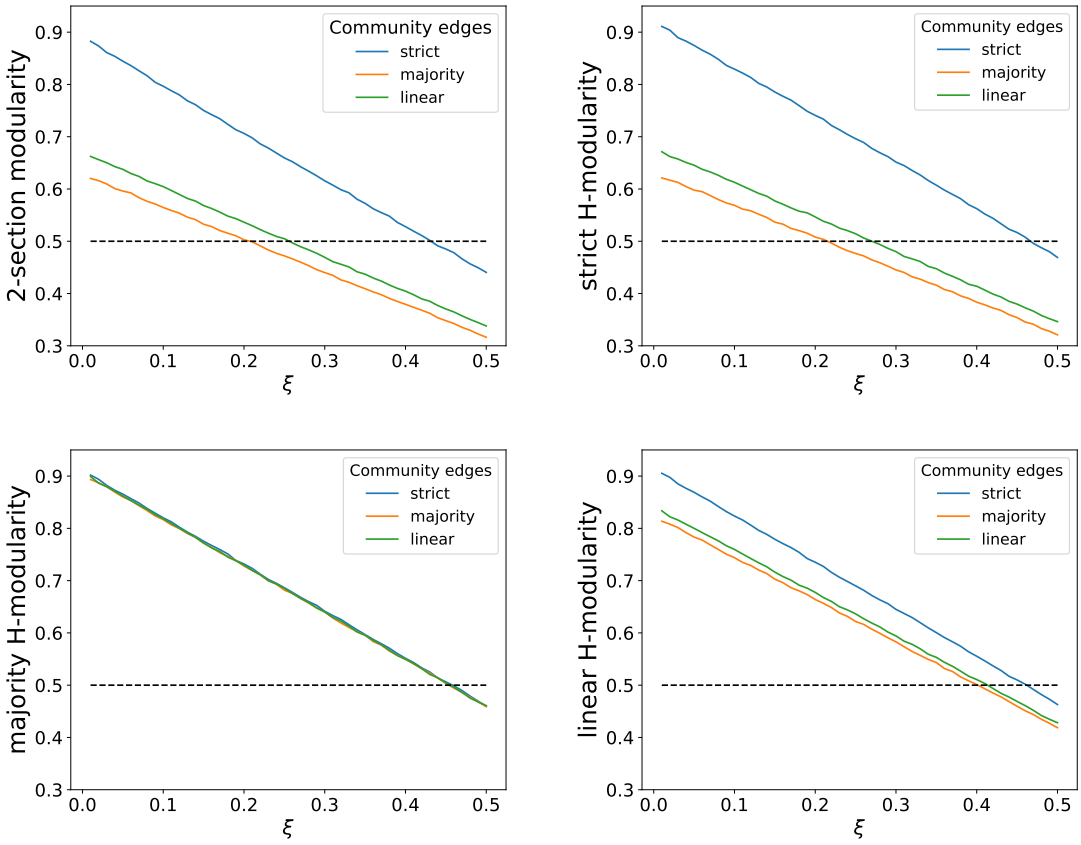

Figure 6: Four modularity functions of **h–ABCD** hypergraphs with varying noise parameter ξ. The community hyperedges were generated according to three different models: strict, majority, and linear.