Framework to generate hypergraphs with community structure

Nicolò Ruggeri ,[1,2,*] Federico Battiston ,[3,†] and Caterina De Bacco [1,‡]

[1]*Max Planck Institute for Intelligent Systems, Cyber Valley, 72076 Tübingen, Germany*
[2]*Department of Computer Science, ETH, 8004 Zürich, Switzerland*
[3]*Department of Network and Data Science, Central European University, 1100 Vienna, Austria*

(Received 23 December 2022; accepted 11 January 2024; published 19 March 2024)

In recent years hypergraphs have emerged as a powerful tool to study systems with multibody interactions which cannot be trivially reduced to pairs. While highly structured methods to generate synthetic data have proved fundamental for the standardized evaluation of algorithms and the statistical study of real-world networked data, these are scarcely available in the context of hypergraphs. Here we propose a flexible and efficient framework for the generation of hypergraphs with many nodes and large hyperedges, which allows specifying general community structures and tune different local statistics. We illustrate how to use our model to sample synthetic data with desired features (assortative or disassortative communities, mixed or hard community assignments, etc.), analyze community detection algorithms, and generate hypergraphs structurally similar to real-world data. Overcoming previous limitations on the generation of synthetic hypergraphs, our work constitutes a substantial advancement in the statistical modeling of higher-order systems.

DOI: 10.1103/PhysRevE.109.034309

I. INTRODUCTION

Over the past decades, networks have emerged as a fundamental tool to describe complex relational data in nature, society, and technology [1]. Indeed, most real-world systems are nowadays known to be characterized by a highly nontrivial organization, which includes triadic closure and high clustering [2], low diameter and an efficient communication structure [3], and unequal degree distributions [4]. Noticeably, many systems reveal the existence of modules or communities, where nodes are naturally clustered in different groups based on their patterns of connections [5]. Identifying communities is an important task that allows performing various downstream analysis on networks, describing the roles of nodes and, generally, providing a low dimensional representation of possibly large systems. Since the seminal papers by Newman *et al.* [6] and Lanchichenetti *et al.* [7], the problem of generating synthetic data for highly structured graphs with prescribed features has attracted enormous interest in the community. On the one hand, these models have led to tremendous improvements in evaluating which community detection algorithms perform best at a given task [8]. On the other hand, they have allowed the reliable generation of large synthetic data samples, useful to analyze nontrivial statistics from single instances of real networks and systematically investigate the impact of mesoscale structure on dynamical processes on graphs [9,10]. This methodology has been applied to different domains, including studies on polarization on social media [11], percolation thresholds in brain networks [12], and structural and covariate information [13,14].

Despite their success, recent evidence suggests that graphs can only provide a limited description of reality, as links are inherently limited to describe pairwise interactions [15–18]. By contrast, nondyadic higher-order interactions have been observed across different domains, including the human brain [19–21], collaboration networks [22], species interactions [23], cellular networks [24], drug recombination [25], and face-to-face human [26] and animal [27] interactions. Interestingly, such higher-order interactions naturally lead to the emergence of new collective phenomena in synchronization [28–32] and contagion [33–35] dynamics, diffusive process [36,37], and evolutionary games [38,39]. Hypergraphs [40], where hyperedges encode interactions among an arbitrary number of system units, are a natural framework to describe relational data beyond the pair [15]. In the past few years many tools have been developed to characterize the higher-order organization of real-world hypergraphs, including new centrality measures [41,42], higher-order clustering [43] and motif analysis [44], hypergraph backboning [45], hyperedge prediction [46], and methods to infer higher-order interactions from low-order data [47]. In particular, several tools to extract higher-order communities have been proposed, either based on flow distribution [48,49] or statistical inference frameworks [46,50].

Nevertheless, how to generate structured hypergraphs is still an open problem. The few currently available models mainly focus on “unstructured” higher-order generalizations of the configuration [51–53] and the Erdos-Renyi model [54], or on growth models for hypergraphs [55–57]. A different perspective is that of relational hyperevent models [58], which

[*] nicolo.ruggeri@tuebingen.mpg.de
[†] battistonf@ceu.edu
[‡] caterina.debacco@tuebingen.mpg.de

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article’s title, journal citation, and DOI. Open access publication funded by the Max Planck Society.

specify event rates based on hyperedge statistics for hyperedges to exist, similarly to what exponential random graphs do for networks [59,60]. All these approaches, however, do not account for community structure, hence are of limited usage when it comes to reproducing the complex mesoscale organization of real-world higher-order systems. Recent works introduced latent variables models to infer community structure in hypergraphs [46,50,61], however they do not explain how to sample from the generative model. Indeed, while sampling and inference are often studied jointly in standard networks, these two tasks present distinct computational and theoretical challenges in the case of hypergraphs.

In this work, we provide a principled and general framework to sample hypergraphs. In particular, our method allows flexible sampling of higher-order networks with prescribed microscale and mesoscale features, controlling the distribution of node degrees and hyperedge sizes, as well as specifying arbitrary community structure (e.g., hard versus overlapping membership, assortative versus disassortative, etc.). The method is highly efficient, and scales well with the number of nodes, hyperedges, as well as hyperedge size, making it suitable for the analysis of real-world systems. In the following, we first introduce our generative model and sampling strategy. Then, we extensively characterize the hypergraphs obtained by investigating the phase space associated with the different structural parameters. Finally, we show how to utilize our method to analyze the structural and statistical properties of real-world data.

II. GENERATIVE MODEL

We consider hypergraphs $\mathcal{H}(V, E)$ consisting of N nodes $V = \{1, \dots, N\}$ and a hyperedge set E, where each hyperedge $e \in E$ describes an interaction among an arbitrary set of unique nodes, i.e., $e \subseteq V$, and $|e|$ is the hyperedge size. The degree of a node i, i.e., the number of hyperedges it belongs to, is denoted as d_i. Similarly, we define the degree sequence $d = \{d_1, \dots, d_N\}$ as the vector of node degrees and the size sequence $k = \{k_1, \dots, k_D\}$ as the count of hyperedges per hyperedge size [53]. We consider hyperedges of arbitrary sizes, up to a maximum of $D \leqslant N$, and denote the space of all possible such hyperedges with Ω. We assume positive and discrete hyperedge weights, encoded using a vector $A \in \mathbb{N}^{|\Omega|}$, so that $E = \{e \in \Omega : A_e > 0\}$.

Our sampling approach introduces a flexible way to generate highly structured weighted hypergraphs with mesoscale structure, where hyperedges are generated probabilistically and nodes belong to K communities. Specifically, each node $i \in V$ is assigned a K-dimensional membership vector u_i, where we allow $u_{ik} \geqslant 0$ for the general case of soft membership, where nodes can belong to multiple communities. The particular case of hard membership assignment, where a node can only belong to one community, is recovered by setting only one nonzero entry for u_i. In Fig. 1 we illustrate these two cases by showing two small hypergraphs with hard or overlapping community structure. The nonnegative symmetric $K \times K$-dimensional *affinity* matrix w regulates the interactions between communities. Classic patterns are assortative affinity matrices, with dominant diagonal signaling stronger inter-community interactions, and disassortative ones, where the out-diagonal terms have higher magnitude. For any given hypergraph, we define the following likelihood function:

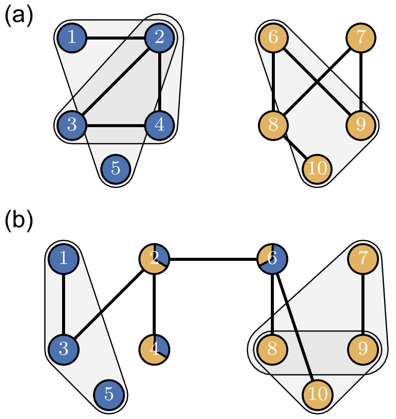

FIG. 1. Sampling hypergraphs with community structure. A pictorial representation of two small hypergraphs with $N = 10$ nodes, $K = 2$ communities, and (a) hard or (b) overlapping membership assignment. Every node's membership assignment $u_i = (u_{i1}, u_{i2})$ is represented as a pie chart. Nodes with a single color have hard assignments, mixed pie charts represent overlapping assignments. Due to the likelihood in Eq. (2), nodes with overlapping assignments are more likely to belong to between-community interactions.

$$p(\mathcal{H}; w, u) = \prod_{e \in \Omega} p(A_e; u, w) = \prod_{e \in \Omega} \text{Pois}\left(A_e; \frac{\lambda_e}{\kappa_{|e|}}\right), \tag{1}$$

where

$$\lambda_e := \sum_{i<j \in e} u_i^T \, w \, u_j = \sum_{i<j \in e} \sum_{k,q=1}^{K} u_{ik} w_{kq} u_{jq}. \tag{2}$$

This parametrization allows generating hypergraphs under different scenarios, e.g., with assortative or disassortative community structures, and is reminiscent of those used in probabilistic models for pairwise networks [62,63] and in variants of nonnegative tensor factorization as used in the machine learning community [64,65] when $D = 2$. In addition, restricting our model to $D = 2$ and $\kappa_2 = 1$ recovers the canonical Poisson stochastic block model [66]. The parameter $\kappa_{|e|}$ is a normalization factor and is a function of the size $|e|$ of the hyperedge e only (i.e., it only depends on the size of the interaction, and not on the nodes involved in it). These constants regulate the expected statistics of the model, such as expected degree and hyperedge size distribution. In general, any choice of $\kappa_d > 0$ yields a well-defined probabilistic model. We illustrate sensible values for κ_d in Appendix A 2.

Alternative generative models for hypergraphs have been recently proposed. In particular, the works of Chodrow *et al.* [50] and Contisciani *et al.* [46] can be more closely compared to the model in Eq. (1), since they are both based on factorized Poisson likelihoods based on communities. The former work assumes sufficient statistics only evaluated on hard community assignments and we are not aware of any computationally efficient sampling procedure from the relative generative process. The model of Contisciani *et al.* [46], instead, bears closer resemblance to the one proposed in this paper. The main difference lies in the specific form of the Poisson means, which, for every hyperedge e, are based on a product of $|e|$ terms, as opposed to the bilinear form in Eq. (1). Despite the similar generative process, the tools utilized in this work cannot be straightforwardly applied to that model, as closed-form statistics and approximate Central Limit Theorem results cannot be derived in the same manner.

More generally, the primary goal of the aforementioned models is to infer hypergraph structure, leaving the problem of sampling unsolved. While our model is also well suited to efficiently infer hypergraph structure, as we illustrate in Ruggeri *et al.* [67], the primary objective of this work is to demonstrate how we can effectively sample from its probability distribution. This key model's capability makes it possible to generate highly structured synthetic data with higher-order interactions. This is a key advancement for practitioners handling hypergraph data and follows influential work on such a topic for pairwise networks [6,7].

III. SAMPLING HYPERGRAPHS

We now propose an efficient way to sample hypergraphs from the generative model defined in Eq. (1). Such a task is far from being straightforward. To see why, let us consider a pairwise network model, where the configuration space is of size $|\Omega| = N^2$, and compare it with our higher-order problem. In the former case, generation is feasible by simply exploring every single edge separately and sampling from the relative Poisson distribution. In the latter case, however, the rapid growth of the Ω space renders both naive sampling techniques and Markov chain Monte Carlo (MCMC) algorithms inapplicable. Here, we propose a solution to this challenge using approximate sampling. In the following, we focus on the intuition behind our method and illustrate relevant usage examples. For a more technical description, we defer to Appendix B.

A. Sampling algorithm

Our sampling procedure follows three consecutive steps:

1. Sampling node degrees and hyperedge sizes

The first sampling step consists of approximately sampling the d and k vectors for a given choice of community memberships u and affinity matrix w. Then, we use these two quantities to draw a first proposal of a binary hypergraph defined by the array $A^b \in \{0, 1\}^{|\Omega|}$. More in detail, we first approximate $p(d, k; u, w) \approx p(d; u, w)\, p(k; u, w)$ and then use the central limit theorem (CLT) to sample from $p(d; w, u)$ and $p(k; w, u)$ separately. We note that these are the only approximations needed in the whole sampling routine. We elaborate more on their validity in Appendix E. After sampling the d, k sequences, we combine them into a first binary hypergraph configuration (i.e., a list of hyperedges) to be passed in input to the next sampling step. Intuitively, we incrementally build a hyperedge list until exhaustion of both sequences, starting by first taking the nodes with the highest degrees. If the two sequences are not compatible, i.e., there is no hypergraph that satisfies both, then one can choose which of the two sequences to preserve during the hyperedge list construction. Such sequence will be exactly replicated, while the other will be modified to construct the first list proposal. Notice that the recombination problem has connections with the Havel-Hakimi algorithm [68] and the Erdös-Gallai theorem [69]. Hence, the algorithm we propose for this task is a technical novelty of independent interest. We explain the algorithm in detail and present a pseudocode for it in Appendix E.

2. Sampling hyperedges

In this second step, we sample the binary hyperedges A^b_e, conditioned on d and k, using a MCMC routine. This works by continuously mixing the hyperedges starting from the initial proposal A^b obtained at step a. The main tool utilized here is the reshuffling operator introduced in Chodrow *et al.* [53]: given two hyperedges e_1, e_2, reshuffle the nodes not belonging to the intersection $e_1 \cap e_2$ to obtain two new hyperedges e'_1, e'_2. Then, accept or reject the new proposal according to the Metropolis-Hastings algorithm [70], whose acceptance rates depend on the Poisson means $\lambda_{e_1}/\kappa_{e_1}, \lambda_{e_2}/\kappa_{e_2}$ and consequently on the u, w parameters. Due to the properties of the reshuffling operator the new hyperedges e'_1, e'_2 have same sizes as e_1, e_2, hence the sequences d and k are preserved. Intuitively, the Markov chain achieves good mixing owing to conditioning on (d, k), which restricts the space of the possible configurations.

3. Sampling hyperedge weights

In the third and final step, we sample the weights A_e from $p(A_e|A^b_e = 1)$. This conditional distribution is a zero-truncated Poisson with mean $\lambda_e/\kappa_{|e|}$. A related efficient sampling procedure based on inverse transform sampling is proposed in Appendix B 3.

Altogether, the three sampling steps described above correspond to the following probabilistic decomposition:

$$p(A; u, w) = p(A|A^b; u, w)\, p(A^b|d, k; u, w)\, p(d, k; u, w). \tag{3}$$

We provide the pseudocode of the sampling procedure in Algorithm 1 and provide an open-source implementation at Ref. [71].

B. Additional user input

The sampling procedure described above only requires the community assignments u and affinity matrix w as generative parameters. However, a practitioner may desire to generate hypergraphs with specific features, such as a given degree or hyperedge size sequence. Our model allows doing so naturally, either by providing such statistics as additional input or by tuning the generative parameters prior to sampling. More precisely, one can skip the initial step and simply fix d or k (or both) as input instead of sampling them. As explained in Sec. III A, these quantities are guaranteed to be preserved in the sampled hypergraphs. Algorithmically, this corresponds to starting directly from line Line 3 in Algorithm 1.

In some cases, one might be interested in replicating the $d^{\text{data}}, k^{\text{data}}$ sequences observed in a real hypergraph dataset. In such a simplified scenario, one can condition on the (binarized) hyperedges of the data, and proceed by directly mixing

Algorithm 1. Sampling algorithm. *a*: Lines 1–3; *b*: Lines 4–10; *c*: Line 11.

```
   Input: Number of communities $K$, memberships $u$, affinity $w$,
      MCMC burn-in steps $n_b$ and intermediate steps $n_i$, number of
      samples $S$.
   Result: $\{A^{(s)}\}_{s=1,\dots,S}$
1  Sample binary degree sequence $d \sim p(d; u, w)$
2  Sample size sequence $k \sim p(k; u, w)$
3  Create first proposal $A^b$ from $d, k$
4  for $i = 1, \dots, n_b$ do
5      $A^b \leftarrow$ reshuffle($A^b$), accept according to Metropolis-Hastings,
         depending on $(u, w)$
6  end
7  for $s = 1, \dots, S$ do
8      for $i = 1, \dots, n_i$ do
9          $A^b \leftarrow$ reshuffle($A^b$), accept according to Metropolis-Hastings,
             depending on $(u, w)$
10     end
11     sample $A^{(s)} \sim p(A|A^b; u, w)$
12     yield $A^{(s)}$
13 end
```

them via the MCMC procedure in the second sampling step. Since the hyperedges define the degree and size sequences, these will be preserved and identical to those of the real data, while the samples will come from the model's probability distribution. As per Eq. (3), the MCMC procedure will yield samples from $p(A^b|d^{\text{data}}, k^{\text{data}}; u, w)$. Notice that, in general, conditioning on any given sequence d or k might yield samples A outside the high-density areas of the distribution. This is a desirable feature, as it allows the user to further specify constraints and sample hypergraphs that would otherwise be far from the typical samples obtained without conditioning [72].

Finally, with our model we can obtain closed-form expressions for relevant hypergraph properties in terms of u and w, e.g., the expected degree of nodes, as shown in Appendix A 1. This means that, by tuning the u, w parameters, such properties can be specified prior to sampling. We illustrate some examples of this procedure in Sec. IV.

IV. SYNTHETIC DATA

In this section, we illustrate how the generative parameters u and w can be tuned to sample hypergraphs with desired structures at a micro (node and hyperedge) and mesoscale (community structure and hypergraph-level statistics) level. We release ready-to-use examples of these synthetic datasets along with the open-source implementation.

A. Community assignment

We begin by showing how varying the overlap in the membership assignments u leads to different intra and inter-commmunity structure. In Fig. 2 we tune the assignments from hard (u_i has only one nonzero entry) to soft ($u_i > 0$ for multiple entries), and we highlight the strength of the interactions between and within communities by varying the thickness of edges and circles. We include additional details on data and plot generation in Appendix F 1. As memberships vary from hard to soft (left to right), edges become thicker and circles smaller, as intercommunity (intracommunity) interactions increase (decrease). Quantitatively, we compute the

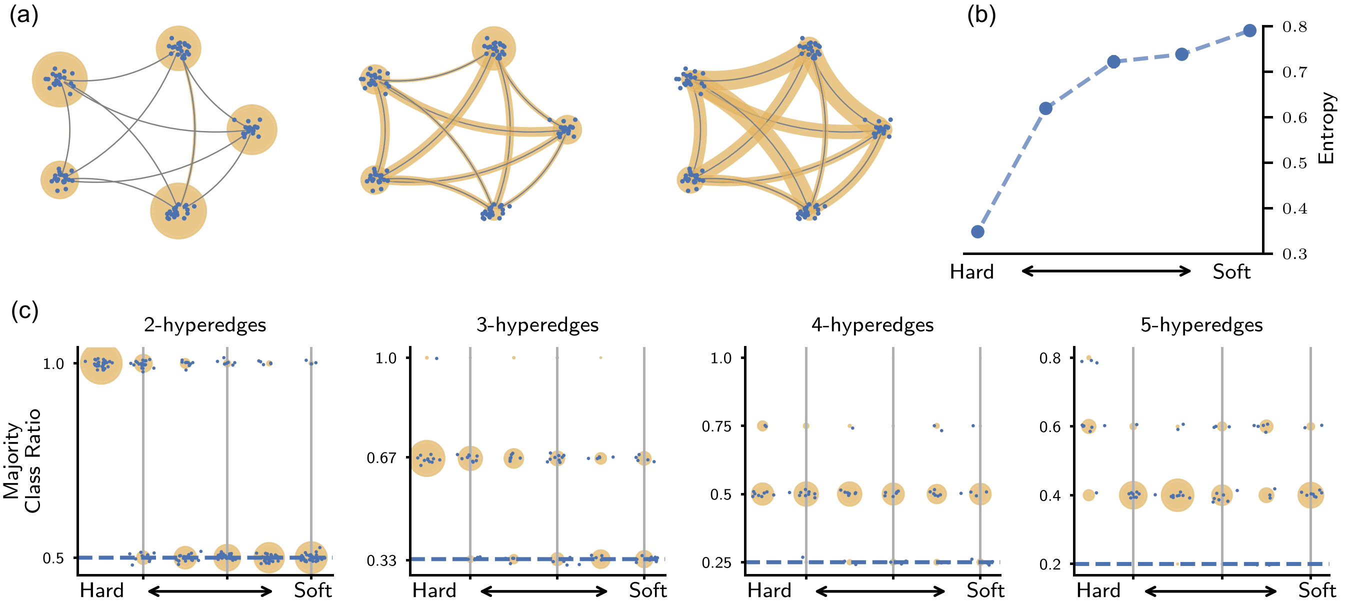

FIG. 2. Sampling hypergraphs with hard and soft community assignment. (a) We sample hypergraphs from a model with $K = 5$ equally sized communities, an assortative affinity matrix w, and different node community memberships u (from hard to soft). The five vertices represent different communities, the thicknesses of the edges and circles are proportional to the interaction strength between and within communities, see Appendix F 1 for details. (b) The entropy of community memberships grows as increasingly overlapping configurations are considered. (c) We show the maximum assignment ratio (the relative number of nodes belonging to the majority class for each hyperedge) across hyperedge sizes. The set sizes are proportional to the amount of hyperedges with a given maximum assignment ratio.

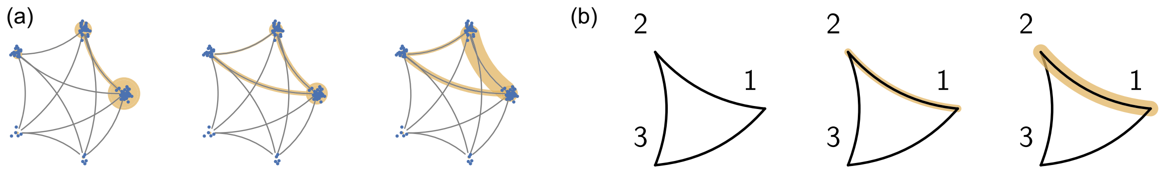

FIG. 3. Sampling hypergraphs with assortative and disassortative affinity and heterogeneous community size. (a) We sample hypergraphs with five communities of different sizes and hard membership assignments. We vary the affinity matrix w from assortative (left, diagonal) to disassortative (right, uniform matrix filled with ones). The five vertices represent the communities, the thicknesses of the edges and circles are proportional to the interaction strength between and within communities, respectively. (b) We vary the affinity w from diagonal (left) and increase its entries w_{12}, w_{21} (right) for $K = 3$ equally sized communities. The three vertices represent the communities and the thickness of the edges and circles is proportional to the strength of the interactions between and within communities respectively, see Appendix F 1 for details.

entropy $-\sum_{k=1}^{K} r_k \log r_k$, where r_k is the ratio of nodes belonging to community k. In mixed-membership settings, one can extract a proxy for a hard assignment for node i by selecting the $k = \arg\max_k u_{ik}$; we use this to compute r_k. Lower entropy denotes hyperedges whose nodes mostly belong to the same communities, higher values denote hyperedges with nodes distributed across different communities. In Fig. 2(b) we show how the entropy of the community distribution grows as we sample from increasingly overlapping models. We also study the partition in communities of nodes belonging to hyperedges of different sizes. For each hyperedge we compute the ratio of nodes that belong to the majority class. For example, in a hyperedge of size 5 with two nodes in class 1 and three in class 2, the majority class is 2, yielding a majority class ratio of 3/5. Figure 2(c) shows how this ratio decreases going from hard to soft memberships, illustrating the heterogeneity of the nodes' communities across hyperedges of different sizes.

B. Affinity matrix and heterogenous community size

While varying u acts on the propensity of individual nodes to participate in groups, the affinity matrix w controls the density of interactions within and between communities. The generative model in Eq. (1) is well-defined for any nonnegative symmetric affinity matrix w, allowing simulating various structures by properly tuning its entries. To illustrate the generation of hypergraphs with different affinity matrices, here we consider a range of matrices that start from diagonal (assortative) to gradually move to the uniform matrix of ones (disassortative), and rescale them to obtain an expected degree of five. For simplicity we set the assignments u to hard membership. The method is well suited to sample not only homogenous hypergraphs, but also higher-order networks with heterogenous distribution of the community size. Here we consider five communities with different sizes. As shown in Fig. 3(a), moving from an assortative to a disassortative configuration, the inter-community interactions strengthen substantially. Further, notice that the strengths of the interactions are influenced by the heterogeneity of the community size, as larger communities are expected to participate in more interactions.

It is also possible to tune individual entries of the affinity matrix w. In particular, in Fig. 3(b) we perform an experiment where we start from a diagonal matrix, and gradually increase only the w_{12} (and w_{21}) entries, using three equally sized communities. In this way, only the expected interactions between communities 1 and 2 are affected, while interactions among other communities are left unchanged.

C. Analzying community detection

One of the most useful applications of generating synthetic data with a desired underlying structure is the possibility to evaluate how competing algorithms perform on a given task that depends on the structure under control. In fact, when synthetic data with a known structure is available, it is possible to quantitatively compare the outcome of various algorithms and measure their ability to recover ground truth information. In network science, a classical and much investigated problem is assessing the ability of community detection algorithms to extract meaningful partitions of the network [7]. For higher-order networks, the current lack of sampling methods for synthetic data with flexible community structure has led to a variety of custom-built examples, which renders comparison difficult and subject to individual choices [46,50,73].

In this section, we show how our synthetic data can be utilized to analyze the behavior of some of the current algorithms for higher-order community detection. To this end, we generate hypergraphs with assortative structure and hard community assignments, and perform inference with a variety of methods, namely Hy-MMSBM [67], Hypergraph-MT [46], spectral clustering [74], and hypergraph modularity [50]. In Fig. 4, we show the cosine similarity of the inferred communities with the ground truth as a function of the maximum hyperedge size. As can be observed, Hy-MMSBM attains the best performance when group interactions beyond a critical size are introduced, successfully recovering the ground truth assignments. Hy-MMSBM is a flexible inference tool whose inference procedure is based on the same generative model described in Eq. (1), and generally able to extract mixed-membership assignments for arbitrary (e.g., assortative or disassortative) community structure. Other algorithms attain varying scores, which might be explained by the different assumptions of the underlying models. For example, Hypergraph-MT is designed to extract overlapping communities, while spectral clustering can only be utilized for the detection of hard assignments. As such, the latter can be expected to perform well only in scenarios where interactions are dictated by hard communities, while the former can be employed when nodes may belong to more than one module. Considerations of this type can be useful to compare the

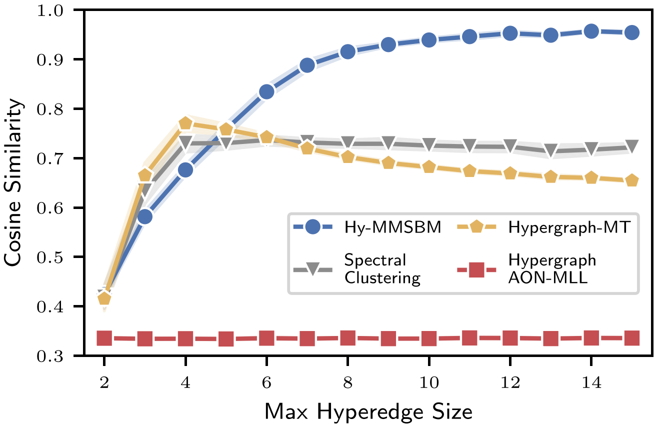

FIG. 4. Evaluating higher-order community detection algorithms. We sample hypergraphs to test the ability of different higher-order community detection algorithms to recover well-defined planted partitions. We consider hypergraphs with $N = 500$ nodes, $K = 3$ equally sized assortative communities and hard assignments. We plot the cosine similarity between the inferred partitions and the ground truth as a function of the maximum hyperedge size. Additional details on the data generation are given in Appendix F 2.

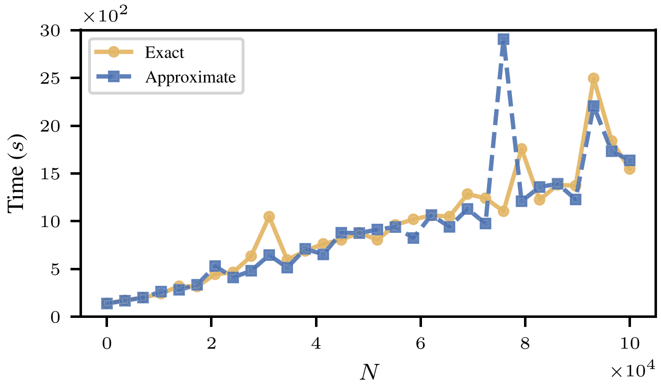

FIG. 5. Computational complexity and scalability. We plot the computational cost of our sampling model for sparse hypergraphs as a function of the system size N. Our model is highly efficient, as it allows sampling of sparse hypergraphs of dimensions up to $N = 10^5$ nodes in less than one hour. We show results for hypergraphs with fixed expected degree equal to 5, both for an exact (solid line with circles) and an approximate approach (dashed line with squares) based on central limit theorem sampling of dyadic interactions. Here, we utilize $K = 5$ communities and unconstrained maximum hyperedge size $D = N$.

alignment of different algorithms with the data generation procedure, which is expect to correlate with their performance on such data. However, due to the additional optimization and implicit bias of most algorithms, it is sometimes unclear to know beforehand on what types of data each algorithm will perform well empirically. We believe that synthetic data with known ground-truth structure can provide useful diagnostic and comparison tools in this direction.

Procedures like the one presented in this section can be used to understand the limitations and strengths of different algorithms, allowing researchers to effectively test new proposals in different scenarios by varying the properties of the samples generated with our method, e.g., the degree of assortativity.

D. Computational cost

Our sampling method is highly efficient and computationally scalable. We analyze the cost of our sampling strategy by discussing the cost of the individual sampling steps. The first step, consisting of sampling the degree and size sequences, can be cheaply performed in $O(N)$ time. In fact, to sample the d, k sequences we need to compute the mean and standard deviations defined in the Central Limit Theorem, and thus draw the sequences from the relative Gaussian distributions. These operations have linear cost, see Appendix B 1. In the second step we first combine the sampled d, k sequences into a first hyperedge configuration, and successively mix the hyperedges via MCMC. Generally, while the number of Markov chain steps needed for mixing is a function of N and $|E|$ [75], it is difficult to specify a pre-defined number. In Fig. 5, we fix $n_b = 100\,000$ burn-in steps and $n_i = 20\,000$ intermediate steps between samples, which is a default value we utilized in most experiments. Nonetheless, the main cost we observe in this case is that prior to MCMC, i.e., the producing the first hyperedge configuration from the sequences. Empirically, such step dominates the computational cost. Finally, the third step consists of sampling the nonzero weights according to $p(A|A^b; u, w)$. The cost of this operation is proportional to the number of hyperedges $|E|$; for sparse hypergraphs—and as often observed in real data—this is comparable to N.

Empirically, we find the CLT approximations to be working well. Nevertheless, one could further improve on the quality of sampling by drawing the pairwise edges from their *exact* Poisson distribution [Eqs. (1) and (2)], with cost $O(N^2)$, and resorting to approximations only for interactions of order three or greater. This is of particular help when sampling denser hypergraphs: since the MCMC does not necessarily guarantee nonrepeated hyperedges, sampling directly the order-two interactions reduces the probability of repeated edges. For higher-order interactions, the probability of repetitions is negligible, in particular in sparse regimes [53]. Indeed, in all the experiments presented in this paper we sample the order-two interactions directly, and resort to the CLT approximations for hyperedges of order at least three.

In Fig. 5 we investigate the efficiency of both exact and solely CLT-based sampling and observe the difference to be negligible. As discussed above, this is a consequence of the higher computational effort required in other sampling steps. Altogether, our model is highly efficient, as it allows sampling sparse hypergraphs of dimensions up to 10^5 nodes in less than one hour.

V. REAL DATA

A. Modeling real-world systems

In this section we aim at sampling hypergraphs that mimic the community structure of a given dataset. To this end, we proceed as follows. First, we infer the affinity matrix w and community assignments u using the Hypergraph-MT algorithm [46] on the real data. Since this algorithm returns a (diagonal) matrix w_d for every possible hyperedge size d,

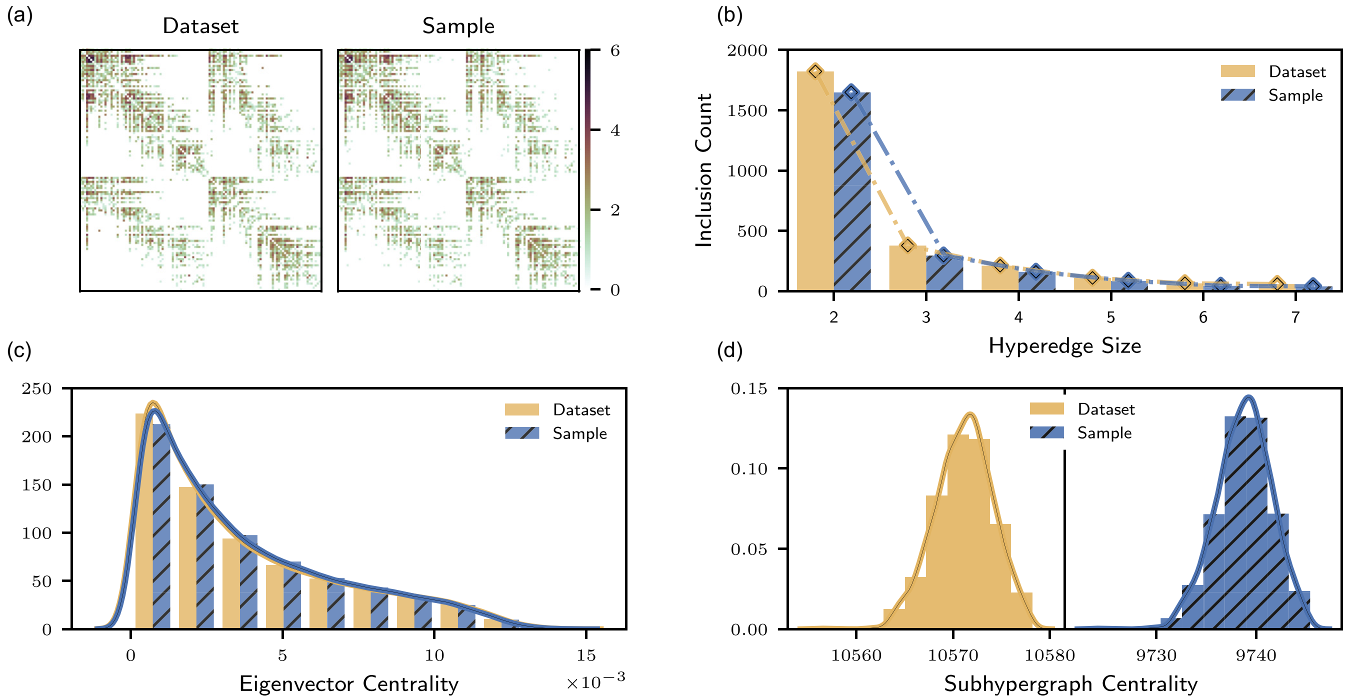

FIG. 6. Matching statistics of real-world data and samples: the case of the House Bills dataset We plot (a) the adjacency matrices, (b) the hyperedge inclusions occurrences, (c) the hyperedge eigenvector centrality distribution and (d) the subhypergraph centrality distribution for the House Bills dataset, where nodes represent congresspersons, and hyperedges describe subsets of them that cosponsor a bill. For all such cases, we observe a good correspondence between the statistics measured on the real data and those obtained from a single sample of our generative model.

we take their element-wise geometrical mean to construct the matrix w utilized in Eq. (2). Notice that a similar approach could have been taken utilizing the Hy-MMSBM algorithm, which employs the same probabilistic model of our sampling method, as explained in Sec. IV C. To highlight the flexibility of our methodology, which can be applied along with any community detection methodology, here we utilize Hypergraph-MT. In fact, our method accepts input parameter w and u regardless how these are obtained; in particular, these can be obtained by using different inference methods applied to the input data. Our method is capable of generating synthetic data conditioning on the desired input communities and affinity matrix. As such, it can be used in a complementary way together with community-based method focusing solely on inference. Second, we condition the degree and size sequences by providing in input the observed hyperedge configuration, i.e., the hyperedges present in the real data. As explained in Sec. III B, this means skipping the first step of our sampling procedure and moving directly to perform MCMC starting from such configuration. The returned hypergraphs will have a structure similar to that of the data, but will be sampled according to the generative model in Sec. II.

B. Comparing data and sample statistics

We now apply the proposed methodology on a variety of real datasets. As a representative example, we consider a dataset of cosponsoring of bills for the U.S. House of Representatives [76,77]. Nodes correspond to congresspersons, and hyperedges connect subsets of them that cosponsor a bill. The dataset contains $N = 1494$ nodes, $|E| = 54\,933$ hyperedges with maximum size $D = 399$, and has been previously analysed via higher-order stochastic block models [46,50].

As a first sanity check, in Fig. 8 of the Appendix we verify that the degree and size sequences measured on the samples are identical to those of the data. This is guaranteed by the properties of the reshuffling operator described in Sec. III. We then proceed by comparing additional relevant statistics as measured on the real data and on the samples. Such statistics serve as a test for the goodness of fit, as they should match if the dataset is well-represented by the model.

We start by performing a visual comparison of the adjacency matrices [15,78], where the adjacency value X_{ij} of any two nodes i, j is defined as $X_{ij} := \sum_{e\in E:i,j\in e} A_e$. As shown in Fig. 6(a), our samples are well aligned with the real data.

Another relevant structural property of a hypergraph is the inclusion relationships between hyperedges, i.e., which hyperedges are subsets of others [44]. This is of particular interest when comparing a hypergraph with its clique expansion, i.e., the graph obtained by projecting hyperedges onto pairwise interactions, or when comparing with other higher-order representations such as simplicial complexes [79,80]. In Fig. 6(b), we count the number of hyperedges of size n that are included in hyperedges of size $n + 1$. Also in this case, results on our sample match well those measured on the input dataset.

Finally, we explore two centrality measures on hypergraphs. As a first example, in Fig. 6(c) we consider a generalization of eigenvector centrality [81] for hyperedges. In particular, we consider the dual representation of the hypergraph,

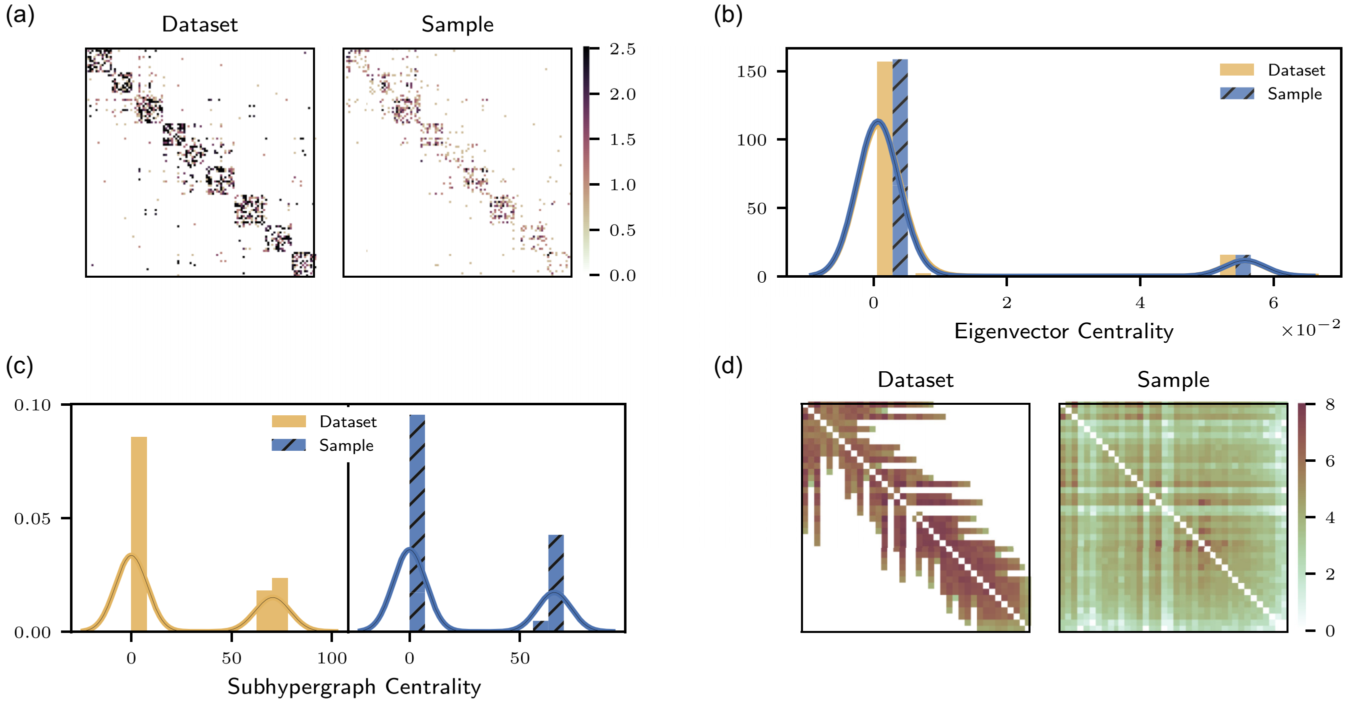

FIG. 7. Hypergraph sample statistics, null hypothesis and generative assumptions. To illustrate the wide applicability of our model, we compare several statistics on real and sampled data. We plot (a) the adjacency matrix of face-to-face higher-order interactions among students in a High School dataset, (b) the eigenvector centrality distribution of copurchasing behavior at Walmart, (c) the subhypergraph centrality distribution from committees data in the U.S. House. Similarly to the results presented in Fig. 6, our model correctly reproduces the desired statistics. In panel (d) we show the adjacency matrix associated with covoting Justices of the U.S Supreme Court. Such data have a strong temporal structure which is not included in the generative assumption of the model, hence explaining the limited correspondence between real and synthetic statistics.

where nodes represent interactions in the original hypergraph and are connected if they have a nonempty intersection [82]. Moreover, in Fig. 6(d) we also compute subhypergraph centrality [78], which returns a measure of node importance in hypergraphs. Also in such cases, the quantities measured on our samples behave similarly to those based on the input data.

We highlight that the resemblance between samples and real data is not simply due to the Markov Chain being stuck in a local optimum given by the initial configuration, i.e., the real dataset. To prove this, we further investigate the Markov Chain mixing while producing the samples based on the House-Bills dataset. We observe that 73% of the shuffling steps are accepted by the Metropolis-Hastings algorithm, signaling good mixing. As an additional structural confirmation, we measure the Jaccard similarity between the real data and 10 samples, defined as the number of hyperedges in the intersection divided by the number of hyperedges in the union. Also in this case, the resulting score of 0.69 ± 0.11 signals that the microscopic structure of the samples detaches from that of the real data, while the macroscopic statistics in Fig. 6 are preserved. Finally, we also observe that less structured methods fail to replicate such statistics. In Appendix H we obtain samples utilizing the configuration model from Chodrow [53], which only takes into account the degree and size sequences. In this case, we observe a significant difference between the samples and the data, which could be explained by the lack of additional probabilistic structure in the sampling procedure. In Appendix I we provide additional studies based on synthetic samples, showing how these can be employed for formalizing and carrying out hypothesis testing on complex structural patterns.

To illustrate the wide applicability of our method, we extend this analysis to additional systems. In Fig. 7(a) we report the observed adjacency matrix of face-to-face interactions among High School students [83], and the one obtained from a sample of our generative model. In Fig. 7(b) we show the distribution of the hyperedge eigenvector centrality computed on copurchasing customer Walmart data [84]. Finally, in Fig. 7(c) we compare the subhypergraph centrality on the House Committees dataset [50,85], where hyperedges connect the members of the U.S. House participating in the same committees. In all such cases, we observe that our sampling method successfully models the desired statistics of the real data.

Synthetic data generated to incorporate a particular structure are often utilized as tests for null hypotheses. Indeed, discrepancies between sampled and real data may arise if some data features are not explicitly taken into account by the generative assumptions of the model [86]. Observing such differences can help unveil some relevant additional structure present in the data and originally neglected. As an example, we consider a dataset of covoting patterns of the U.S. Supreme Court Justices, where the nodes are Justices and hyperedges describe covoting behaviors observed from 1946 to 2019 [87]. Since the number of Justices is fixed to 9 at any point in time, only interactions between Justices working in overlapping years can exist. Such an intrinsic time dependency, however, is not enforced by our model. Hence, we do not expect samples of our model to match the input adjacency matrix well. We illustrate this in Fig. 7(d), where the comparison of the sampled and observed adjacency matrices are distinctively different, with the real data showing a clear time-dependence. Our example illustrates the importance of correctly identifying the existence of particular structures in real-world dataset, showcasing how our sampling method could be used for testing null hypotheses and reproducing real-world statistics.

VI. DISCUSSION

In this paper, we presented a framework for the generation of synthetic hypergraphs with flexible structure. Our model allows specifying different assortative and disassortative mesoscale configurations, tuning the size of the different communities and controlling the strengths of the interactions among them. Moreover, it allows regulating different node-level statistics, including hard or mixed community assignments and expected degrees. Through a variety of experiments, we showed how desired characteristics specified via input parameters are reflected in the generated data. Furthermore, we illustrated how practitioners can use our framework on real systems, both as a computationally efficient sampling tool for the replication of statistical measures, and as a structured null model for hypothesis testing. As an example, our model generates synthetic samples that successfully replicate centrality measures and inclusions relationship between hyperedges in higher-order data from different domains. Similarly, our model can help reveal important missing features in the generative assumptions made by different algorithms, showing clear discrepancies between samples and real data when, for instance, time dependence is ignored. Finally, our framework allows testing the performance of different higher-order community detection methods.

There are various interesting and relevant avenues for future work. A first one is moving from the likelihood in Eq. (1), which is based on a bilinear form, to one based on a multilinear form. Examples from the literature include symmetric tensors [46] and affinity functions [50]. While in principle this would allow for more flexible specifications, such as preventing the formation of certain hyperedge configurations, it is currently unclear how to obtain efficient expressions for the expected statistics and compute the moments required in the Central Limit Theorem. On a similar note, it is important to highlight that ours is only one of different possible definitions of community in hypergraphs. Studying the implications of different probabilistic and optimization procedures on the communities observed in higher-order systems, both theoretically and empirically, is a promising avenue for future work. Moreover, additional information, such as time dependency and attributes on the nodes and hyperedges, could be explicitly incorporated in the probabilistic model. Such an extension could be based on insights from models for dyadic interactions, and result in substantial improvements when this information correlates with the hypergraph structure [88–92].

Taken together, our methodology provides a principled, scalable and flexible framework to sample structured hypergraphs. To facilitate its usage we provide an open-source implementation at Ref. [71]. The method is also implemented as part of the HGX library [93].

ACKNOWLEDGMENTS

We thank Martina Contisciani for the extensive discussions and useful feedback. N.R. acknowledges support from the Max Planck ETH Center for Learning Systems. F.B. acknowledges support from the Air Force Office of Scientific Research under Award No. FA8655-22-1-7025.

APPENDIX A: THE PROBABILISTIC MODEL

We introduce some additional notation to that utilized in Sec. II. Recall that the hyperedges are independent realizations with distribution

$$p(A_e; w, u) = \mathrm{Pois}\left(A_e; \frac{\lambda_e}{\kappa_{|e|}}\right) \quad \forall e \in \Omega,$$

where we define

$$\lambda_e := \sum_{i<j\in e} u_i^T w u_j. \tag{A1}$$

To avoid clutter, we overload the notation and define, for any hyperedge e, $\kappa_e := \kappa_{|e|}$.

Furthermore, recall that Ω is the space of all possible hyperedges of sizes from 2 to D. We also define, for any hyperedge size d, the space of hyperedges of fixed size Ω^d. The function δ is the indicator function, taking value 1 if its argument is true, 0 otherwise.

1. Expected statistics

Our model allows obtaining closed-form expressions for the expectations of various relevant statistics. In the following, we assume that u, w are fixed, show how to derive some of these statistics and compute them in cheap linear time $O(N)$. As explained in Sec. III B, having these statistics available is useful to aid the tuning of u, w prior to sampling. We discuss the choice of the functional form of κ in Appendix A 2.

a. Expected weighted degree of a node. We define the weighted degree d_i^w of a node i as the weighted number of hyperedges it belongs to [15], that is

$$d_i^w := \sum_{e\in E: i\in e} A_e = \sum_{e\in\Omega: i\in e} A_e,$$

due to the fact that $A_e = 0$ for nonexisting hyperedges $A_e \in \Omega \setminus E$. Since A_e is a random variable, the degree of node i is also random and has expectation

$$\begin{aligned}
\mathbb{E}\left[d_i^w\right] &= \sum_{e\in\Omega: i\in e} \mathbb{E}[A_e] = \sum_{e\in\Omega: i\in e} \frac{\lambda_e}{\kappa_e} \\
&= \sum_{e\in\Omega: i\in e} \frac{1}{\kappa_e}\left(\sum_{j\in e: j\neq i} u_i^T w u_j + \sum_{j<m\in e: j,m,\neq i} u_j^T w u_m\right) \\
&= \sum_{e\in\Omega: i\in e} \frac{1}{\kappa_e} \sum_{j\in e: j\neq i} u_i^T w u_j \\
&\quad + \sum_{e\in\Omega: i\in e} \frac{1}{\kappa_e} \sum_{j<m\in e: j,m,\neq i} u_j^T w u_m
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{j\in V: j\neq i} \left[\sum_{n=2}^{D} \frac{\binom{N-2}{n-2}}{\kappa_n}\right] u_i^T w u_j \\
&\quad + \sum_{j<m\in V: j,m\neq i} \left[\sum_{n=3}^{D} \frac{\binom{N-3}{n-3}}{\kappa_n}\right] u_j^T w u_m \\
&= \left[\sum_{n=2}^{D} \frac{\binom{N-2}{n-2}}{\kappa_n}\right]\left[u_i^T w \left(\sum_{j\in V: j\neq i} u_j\right)\right] \\
&\quad + \left[\sum_{n=3}^{D} \frac{\binom{N-3}{n-3}}{\kappa_n}\right]\left[\sum_{j<m\in V: j,m\neq i} u_j^T w u_m\right]. \qquad \text{(A2)}
\end{aligned}
$$

The step from the fourth to fifth row is justified by counting the number of hyperedges of every size n (normalized by the relative κ_n) where both nodes i and j are contained. Notice that the second summand $\sum_{j<m\in V: j,m\neq i} u_j^T w u_m$ has computational cost of $O(N^2)$. We can reduce this to $O(N)$ by making the following general observation, which will also be used in other derivations.

For any fixed set of nodes S and defining $s := \sum_{j\in S} u_j$:

$$
\begin{aligned}
\sum_{j<m\in S} u_j^T w u_m &= \frac{1}{2} \sum_{j,m\in S, j\neq m} u_j^T w u_m \\
&= \frac{1}{2}\left(\sum_{j,m\in S} u_j^T w u_m - \sum_{j\in S} u_j^T w u_j\right) \\
&= \frac{1}{2}\left(\sum_{j\in S}\sum_{m\in S} u_j^T w u_m - \sum_{j\in S} u_j^T w u_j\right) \\
&= \frac{1}{2}\left(s^T w s - \sum_{j\in S} u_j^T w u_j\right). \qquad \text{(A3)}
\end{aligned}
$$

Both these terms can be calculated in $O(|S|)$.

In the case of the expected degree of node i, the second summand of Eq. (A2) can be computed with $S = V \setminus \{i\}$, while the first summand can be directly computed in linear time.

b. Expected weighted degree. This is the average weighted degree of all the nodes in the network, and is given by

$$
\begin{aligned}
\langle d^w \rangle &= \frac{1}{N}\sum_{i\in V} \mathbb{E}\left[d_i^w\right] = \frac{1}{N}\sum_{i\in V}\sum_{e\in\Omega: i\in e} \frac{\lambda_e}{\kappa_e} \\
&= \frac{1}{N}\sum_{e\in\Omega}\sum_{i\in e} \frac{\lambda_e}{\kappa_e} = \frac{1}{N}\sum_{e\in\Omega} |e| \frac{\lambda_e}{\kappa_e} \\
&= \frac{1}{N}\sum_{e\in\Omega} \frac{|e|}{\kappa_e} \sum_{i<j\in e} u_i^T w u_j \\
&= \frac{1}{N}\left(\sum_{n=2}^{D} \binom{N-2}{n-2} \frac{n}{\kappa_n}\right) \sum_{i<j\in V} u_i^T w u_j. \qquad \text{(A4)}
\end{aligned}
$$

This quantity can be reduced to $O(N)$ cost by utilizing the trick in Eq. (A3).

c. Accounting only for specified interactions. The statistics described above can also be computed by taking into account only interactions of a fixed size. For example, a user might be interested in computing the expected degree of a node by considering only hyperedges of sizes up to a certain value, or only for a fixed hyperedge size. These can be readily computed by repeating the derivations above. For example, computing the expected degree in Eq. (A4) only for hyperedges of sizes 2, 3, or 4, we obtain

$$
\frac{1}{N}\left(\sum_{n=2}^{4} \binom{N-2}{n-2} \frac{n}{\kappa_n}\right) \sum_{i<j\in V} u_i^T w u_j.
$$

Notice that only the multiplicative constant $\sum_{n=2}^{4} \binom{N-2}{n-2} \frac{n}{\kappa_n}$ has changed.

2. Choosing the normalization κ_n

The normalization constant κ_n rescales the probabilities of the hyperedges of size n. This rescaling is needed to contrast the effects of the high-dimensional configuration space Ω. Removing the constant (i.e., setting $\kappa_n \equiv 1$ for all n) yields exploding statistics due to the combinatorial factors appearing for larger hyperedges; see, e.g., Eq. (A4).

While it is theoretically possible to sample from a model with such κ_n values, the expected degree and size sequences would not match those observed in real data. In all our experiments we choose instead the following form for κ_n, in a way that yields reasonable expected statistics:

$$
\kappa_n := \frac{n(n-1)}{2}\binom{N-2}{n-2}. \qquad \text{(A5)}
$$

This expression satisfies two important properties. First, $\kappa_2 = 1$, so that the probabilistic model restricted to binary interactions is equivalent to the standard Poisson stochastic block model, second, the expected degree in Eq. (A4) reduces to

$$
\langle d^w \rangle = \frac{1}{N}\left(\sum_{n=1}^{D-1} \frac{1}{n}\right) \sum_{i<j\in V} u_i^T w u_j.
$$

This avoids combinatorial explosions in the expected degree, and allows tuning the model based only on u, w.

The form (A5) has also a valid interpretation. The binomial $\binom{N-2}{n-2}$ normalizes for the number of possible hyperedges of size n that any two fixed nodes belong to (since one needs to choose the remaining $n-2$ nodes in the hyperedge among the possible $N-2$). The value $\frac{n(n-1)}{2}$ is the number of possible binary interactions among n nodes, and is used to take the average of the summands appearing in the expression (A1) for the sufficient statistics: $\lambda_e = \sum_{i<j\in e} u_i^T w u_j$. We also note that similar combinatorial expressions arise naturally in the literature, due to the exploding configuration space [53,94].

While practitioners can make other possible choices with similar properties, e.g., $\kappa_n = \binom{N-2}{n-2}$ or $\kappa_n = \frac{2}{n}\binom{N-2}{n-2}$, we remark that the methodology and the theory proposed in this paper hold for any choice of $\kappa_n > 0$.

APPENDIX B: TECHNICAL DETAILS ABOUT SAMPLING

We include here all the technical details to approximately sample from the probabilistic model. We start by giving some definitions, then we proceed to outline the three sampling steps introduced in Sec. III. In the following, we consider any fixed choice of parameters u, w. Notice also that all the probabilities $p(\cdot)$ utilized in this section depend on u, w. To avoid clutter, we implicitly assume this dependency in the following derivations.

Definition. The *unweighted* (or *binary*) *hypergraph* is the hypergraph with $A^b \in \{0, 1\}^{|\Omega|}$ derived from the original weights $A \in \mathbb{N}^{|\Omega|}$. Formally:

$$A^b_e := \delta(A_e > 0), \quad \forall e \in \Omega.$$

The *binary degree sequence* d is the degree sequence in the binary hypergraph. In other words, it is the degree sequence in the original hypergraph if we consider the hyperedges as unweighted.

The sampling procedure is divided in three consecutive steps:

(i) First, approximately sample the binary degree and size sequences (d, k).

(ii) Second, sample the binary hypergraph A^b from $p(A^b|d, k)$. Notice that, since we sample a binary hypergraph, hyperedges can only exist or not, i.e., $A^b_e \in \{0, 1\}\, \forall e \in \Omega$.

(iii) Third, sample the weights of the final graph given the binary one: $p(A|A^b)$.

Why is this correct? We aim at sampling from $p(A)$. The procedure above corresponds instead to sampling from $p(A, A^b, d, k)$, so why is this correct? Notice that

$$A \text{ uniquely defines } A^b,$$
$$A^b \text{ uniquely defines } (d, k).$$

Thus, if all the sampled quantities A, A^b, d, k are compatible, then we can observe that both the following equalities are true:

$$\begin{aligned} p(A, A^b, d, k) &= p(A^b, d, k|A)\, p(A) \\ &= p(A), \\ p(A, A^b, d, k) &= p(A|A^b, d, k)\, p(A^b|d, k)\, p(d, k) \\ &= p(A|A^b)\, p(A^b|d, k)\, p(d, k), \end{aligned}$$

hence

$$p(A) = p(A|A^b)\, p(A^b|d, k)\, p(d, k).$$

Having verified the consistency of our sampling routine, we now proceed to describe the three steps in more detail.

1. Sampling the binary degree and size sequences

The sequences d, k cannot be sampled exactly and efficiently. We propose instead an approximation based on a version of the central limit theorem.

Consider the binary degree sequence d. For a node i, we need to sample the number d_i of existing hyperedges that i belongs to

$$d_i := \sum_{e \in \Omega : i \in e} \delta(A_e > 0). \tag{B1}$$

Notice that the summands $\delta(A_e > 0)$ are independent Bernoulli random variables with probability $p_e = 1 - \text{Pois}(A_e = 0; \lambda_e/\kappa_e) = 1 - \exp(-\frac{\lambda_e}{\kappa_e})$, therefore easy to sample one by one. Due to the exponential size of Ω, however, sampling all of them is practically impossible.

Similarly, consider the size sequence k. For every hyperedge size $\ell \in \{2, \dots, D\}$ (potentially up to $D = N$), we need to sample the number k_ℓ of hyperedges of such size, defined as

$$k_\ell = \sum_{e \in \Omega^\ell} \delta(A_e > 0),$$

where $\Omega^\ell = \{e \in \Omega \mid |e| = \ell\}$.

The following theorem helps in approximately sampling these quantities.

Theorem 1. Consider d_i, k_ℓ as defined above. Furthermore, assume that u is bounded, i.e., $\exists L > 0 : u < L$, where the inequality is intended element-wise. Then:

(a) For any hyperedge size $\ell \geqslant 3$, both d_i and k_ℓ satisfy the assumptions of the Lyapunov Central Limit Theorem (see Appendix D for the statement). Thus, they can be approximately sampled from a Gaussian.

(b) Furthermore, if we assume that the assignments u are lower bounded away from zero, i.e., $\exists \epsilon > 0$ s.t. $u > \epsilon$, then the statement above also holds for $\ell = 2$.

(c) To a first-degree approximation, we can compute the mean and variance needed for the asymptotic Gaussian distributions as

$$\mathbb{E}[k_\ell] \approx \text{Var}(k_\ell) \approx \sum_{e \in \Omega^\ell} \frac{\lambda_e}{\kappa_e}, \tag{B2}$$

$$\mathbb{E}[d_i] \approx \text{Var}(d_i) \approx \sum_{e \in \Omega : i \in e} \frac{\lambda_e}{\kappa_e}, \tag{B3}$$

i.e., can be approximated by the weighted number of hyperedges and weighted degree.

The proof can be found in Appendix C. Practically, the sampling proceeds as follows. First, we compute the theoretical values in Eqs. (B2) and (B3), this can be done in linear time similarly to the statistics in Appendix A 1. Then, we separately sample d, k from Gaussian distributions with the given means and variances. Since both sequences d, k need to take integer values, we then round the samples element-wise to the closest integer. Finally, we combine the sampled sequences into a first list of hyperedges, representing a binary (i.e., unweighted) hypergraph. We describe the recombining algorithm in Appendix E.

2. Sampling the binary hypergraph

Once we condition on the binary degree and size sequences, we sample which hyperedges will be present in the final hypergraph, i.e., all $e \in \Omega$ such that $A^b_e = 1$.

Formally, the conditional sampling of hyperedges can be performed via the MCMC procedure introduced in Chodrow *et al.* [53]. For this, we use a hyperedge reshuffling operator to define the MCMC steps yielding a valid Markov chain that preserves the initial d, k. The Markov chain starts from a proposal list of hyperedges that is then modified at every step. Notice that the main difference with the MCMC procedure proposed in the original paper is that the acceptance-rejection probabilities are based on our generative models, and hence they depend on the u, w parameters. Hence, we present next how to compute such probabilities in detail.

During the MCMC, we start from two hyperedges e_1, e_2 and shuffle them to form two new hyperedges e'_1, e'_2. Importantly, one of the properties of the shuffle operator is that $|e_1| = |e'_1|$ and $|e_2| = |e'_2|$. The Metropolis-Hastings transition probability to be computed is then

$$\begin{aligned}\frac{p(\text{new configuration})}{p(\text{current configuration})} &= \frac{p(A_{e_1}=0)\,p(A_{e_2}=0)\,p(A_{e'_1}=1)\,p(A_{e'_2}=1)}{p(A_{e_1}=1)\,p(A_{e_2}=1)\,p(A_{e'_1}=0)\,p(A_{e'_2}=0)}\\ &= \frac{\big(\exp\big(\frac{\lambda_{e'_1}}{\kappa_{e'_1}}\big)-1\big)\big(\exp\big(\frac{\lambda_{e'_2}}{\kappa_{e'_2}}\big)-1\big)}{\big(\exp\big(\frac{\lambda_{e_1}}{\kappa_{e_1}}\big)-1\big)\big(\exp\big(\frac{\lambda_{e_2}}{\kappa_{e_2}}\big)-1\big)}.\end{aligned}$$

For some hyperedges it could happen that $\kappa_e \gg \lambda_e$, which may lead to numerical instabilities, as over or underflows in the exponentials (the same holds for the other hyperedges). To mitigate this risk we compute all the λ and the κ in log-space. Using the fact that $e^x - 1 \approx x$ when $x \ll 1$, if we measure that for a hyperedge e we have $\log \kappa_e - \log \lambda_e > \tau$ for a certain threshold τ, then we apply the approximation

$$\exp\left(\frac{\lambda_e}{\kappa_e}\right) - 1 \approx \frac{\lambda_e}{\kappa_e}.$$

In practice, when one among e_1 and e'_1 (and similarly for e_2, e'_2) satisfies the above condition, then we approximate the following ratio as

$$\frac{\exp\big(\frac{\lambda_{e'_1}}{\kappa_{e'_1}}\big)-1}{\exp\big(\frac{\lambda_{e_1}}{\kappa_{e_1}}\big)-1} \approx \frac{\big(\frac{\lambda_{e'_1}}{\kappa_{e'_1}}\big)}{\big(\frac{\lambda_{e_1}}{\kappa_{e_1}}\big)} = \frac{\lambda_{e'_1}}{\lambda_{e_1}},$$

where the last step is due to the fact that the hyperedges have the same size, i.e., $|e_1| = |e'_1|$. This avoids computing the κ values.

3. Sampling the weights of the hypergraph

Finally, we need to sample the Poisson weights from $p(A|A^b)$. Provided that the A^b weights yield a sparse realization, we only need to sample the weights of the few hyperedges present, signalled by $A^b = 1$. Hence, we need to sample from the distribution

$$\mathbb{P}\big(A_e = n | A^b_e = 1\big) = \mathbb{P}(A_e = n | A_e > 0),$$

which is a zero-truncated Poisson distribution. Sampling from such a distribution efficiently is not immediate, we propose a solution based on *inverse transform sampling* [95] next.

Consider a Poisson random variable $X \sim \text{Pois}(\lambda)$. We aim at sampling from the distribution of the random variable Y defined as

$$Y := X | X > 0.$$

To this end, we compute the cumulative distribution function (cdf) of Y. For any $v \in \mathbb{N} \setminus \{0\}$

$$\begin{aligned}F_Y(v) &= \mathbb{P}(Y \leqslant v)\\ &= \sum_{m=1}^{v} \mathbb{P}(Y = m)\\ &= \frac{1}{1-e^{-\lambda}} \sum_{m=1}^{v} \mathbb{P}(X = m)\\ &= \frac{F_X(v) - \mathbb{P}(X=0)}{1-e^{-\lambda}}\\ &= \frac{F_X(v) - e^{-\lambda}}{1-e^{-\lambda}}.\end{aligned}$$

Its inverse, called inverse-cdf or percent-point-function, is given for any $p \in [0, 1]$ by

$$\begin{aligned}Q_Y(p) &= \min\{v \in \mathbb{N} \mid p \leqslant F_Y(v)\}\\ &= \min\left\{v \in \mathbb{N} \mid p \leqslant \frac{F_X(v) - e^{-\lambda}}{1-e^{-\lambda}}\right\}\\ &= \min\{v \in \mathbb{N} \mid e^{-\lambda} + p(1-e^{-\lambda}) \leqslant F_X(v)\}\\ &= Q_X(e^{-\lambda} + p(1-e^{-\lambda})).\end{aligned} \tag{B4}$$

Thus, one can sample from Y by drawing a uniform random variable $p \sim \text{Unif}(0, 1)$ and computing $Q_Y(p)$ via Eq. (B4). Crucially, this is cheap to compute (and simple to implement) because it corresponds to the inverse-cdf of a Poisson distribution.

APPENDIX C: PROOF OF THEOREM 1

Proof. The derivations for the binary degree and size sequences are very similar, for simplicity we only present a proof for the size sequence k_ℓ, for any fixed possible value of ℓ. In the following, call $I_e := \delta(A_e > 0)$ and p_e its Bernoulli probability $p_e = 1 - \exp(-\frac{\lambda_e}{\kappa_\ell})$.

a. Lyapunov CLT for $\ell \geqslant 3$. A possible way to show that the Lyapunov CLT applies, is to show that

$$\lim_{N\to+\infty} \frac{\sum_{e\in\Omega^\ell} \mathbb{E}[|I_e - \mathbb{E}[I_e]|^3]}{\big(\sqrt{\sum_{e\in\Omega^\ell} \text{Var}(I_e)}\big)^3} = 0. \tag{C1}$$

First, observe that

$$\begin{aligned}&\lim_{N\to+\infty} \frac{\sum_{e\in\Omega^\ell} \mathbb{E}[|I_e - \mathbb{E}[I_e]|^3]}{\big(\sqrt{\sum_{e\in\Omega^\ell} \text{Var}(I_e)}\big)^3}\\ &\quad= \lim_{N\to+\infty} \frac{\sum_{e\in\Omega^\ell} p_e(1-p_e)\big((1-p_e)^2 + p_e^2\big)}{\big(\sum_{e\in\Omega^\ell} p_e(1-p_e)\big)^{3/2}}\\ &\quad\leqslant 2 \lim_{N\to+\infty} \frac{\sum_{e\in\Omega^\ell} p_e(1-p_e)}{\big(\sum_{e\in\Omega^\ell} p_e(1-p_e)\big)^{3/2}}\\ &\quad= 2 \lim_{N\to+\infty} \frac{1}{\big(\sum_{e\in\Omega^\ell} p_e(1-p_e)\big)^{1/2}}.\end{aligned}$$

Hence, proving that $\sum_{e\in\Omega^\ell} p_e(1-p_e) \to +\infty$, means that the Lyapunov condition in Eq. (C1) is satisfied. Due to the assumption that the u are bounded, there exists some $0 < L' < 1$ such that $p_e < L'$ for all e (recall that p_e depends on u through λ_e in Eq. (A1)). In the following derivation, let $i, j \in V$ be two arbitrary nodes. Then,

$$
\begin{aligned}
&\sum_{e\in\Omega^\ell} p_e(1-p_e)\\
&\quad\geqslant (1-L')\sum_{e\in\Omega^\ell} p_e\\
&\quad\geqslant (1-L')\sum_{e\in\Omega^\ell: i,j\in e} p_e\\
&\quad= (1-L')\sum_{e\in\Omega^\ell: i,j\in e}\left[1-\exp\left(-\frac{\sum_{l<m\in e} u_l^T w u_m}{\kappa_\ell}\right)\right]\\
&\quad\geqslant (1-L')\sum_{e\in\Omega^\ell: i,j\in e}\left[1-\exp\left(-\frac{u_i^T w u_j}{\kappa_\ell}\right)\right]\\
&\quad= (1-L')\left[1-\exp\left(-\frac{u_i^T w u_j}{\kappa_\ell}\right)\right]\sum_{e\in\Omega^\ell: i,j\in e} 1.
\end{aligned}
$$

Finally, notice that $\sum_{e\in\Omega^\ell: i,j\in e} 1 \to +\infty$ with $N \to +\infty$, due to the fact that the number of hyperedges containing i, j tends to infinity (if the hyperedge size ℓ is at least 3).

b. Lyapunov CLT for $\ell = 2$. Here we also assume that $u < \epsilon$. Then there exists some $L'' > 0$ s.t., for any two nodes m and l, their interaction is bounded away from zero, i.e., $u_l^T w u_m > L''$. Similarly to above, let i be an arbitrary node:

$$
\begin{aligned}
&\sum_{e\in\Omega^\ell} p_e(1-p_e)\\
&\quad\geqslant (1-L')\sum_{e\in\Omega^\ell: i\in e} p_e\\
&\quad= (1-L')\sum_{e\in\Omega^\ell: i\in e}\left[1-\exp\left(-\frac{\sum_{l<m\in e} u_l^T w u_m}{\kappa_\ell}\right)\right]\\
&\quad\geqslant (1-L')\sum_{e\in\Omega^\ell: i\in e}\left[1-\exp\left(-\frac{L''}{\kappa_\ell}\right)\right]\\
&\quad= (1-L')\left[1-\exp\left(-\frac{L''}{\kappa_\ell}\right)\right]\sum_{e\in\Omega^\ell: i\in e} 1.
\end{aligned}
$$

Similar to the case for $\ell \geqslant 3$, the quantity $\sum_{e\in\Omega^\ell: i\in e} 1$ tends to infinity for diverging values of N.

c. Approximation of the statistics. The statistics $\mathbb{E}[k_\ell]$ and $\mathrm{Var}[k_\ell]$ are hard to compute efficiently in closed form. However, we can approximate them using the fact that $1 - e^{-x} \approx x$ and $(1 - e^{-x})\,e^{-x} \approx x$, when $x \ll 1$. Then,

$$
\begin{aligned}
\mathbb{E}[k_\ell] &= \sum_{e\in\Omega^\ell} 1 - e^{-\frac{\lambda_e}{\kappa_\ell}} \approx \sum_{e\in\Omega^\ell} \frac{\lambda_e}{\kappa_\ell},\\
\mathrm{Var}(k_\ell) &= \sum_{e\in\Omega^\ell} \left(1 - e^{-\frac{\lambda_e}{\kappa_\ell}}\right) e^{-\frac{\lambda_e}{\kappa_\ell}} \approx \sum_{e\in\Omega^\ell} \frac{\lambda_e}{\kappa_\ell}.
\end{aligned}
$$

We expect these approximations to be valid when $\frac{\lambda_e}{\kappa_e} \ll 1$. This is a sparse regime attained when only few among all the possible hyperedges are present and is a realistic assumption in many practical applications.

APPENDIX D: LYAPUNOV CENTRAL LIMIT THEOREM

We state here the Lyapunov central limit theorem [96] that we utilize in our main proof in Appendix C.

Theorem 2. Consider a sequence of independent random variables $\{X_i\}_{i\in\mathbb{N}}$, and define $\mu_i := \mathbb{E}[X_i]$, and $\sigma_i^2 := \mathrm{Var}(X_i)$. Also, define

$$
s_n^2 := \sum_{i=0}^{n} \sigma_i^2.
$$

If there exists some $\epsilon > 0$ such that

$$
\lim_{n\to+\infty} \frac{1}{s_n^{2+\epsilon}} \sum_{i=0}^{n} \mathbb{E}[|X_i - \mu_i|^{2+\epsilon}] = 0,
$$

then the quantity

$$
\frac{1}{s_n}\sum_{i=0}^{n}(X_i - \mu_i)
$$

converges in distribution to a standard Gaussian random variable.

APPENDIX E: A MATCHING ALGORITHM FOR THE *d*, *k* SEQUENCES

After separately sampling the binary degree sequence d and the size sequence k, as described in Appendix B 1, these need to be combined to obtain a valid set of hyperedges. However, given two arbitrary sequences d, k, there does not necessarily exist a hypergraph that satisfies both. For example, the average (unweighted) degree can be calculated both from d and k; the two values obtained need to match. For this reason, we need a "matching algorithm" to extract a set of hyperedges that are consistent with both d, k. Notice that this task in not straightforward. Indeed, also Chodrow [53]—who first introduced the Markov chain shuffling procedure—had to start the MCMC from a valid hyperedge list. The currently available implementation [97] can only replicate the sequences of existing hypergraphs, but it is not clear how to proceed from scratch with no initial data, thus reducing the applicability of the method. Our work generalizes the sampling to arbitrary starting sequences. This is a contribution in-and-of-itself, as in principle this could be applied to other sampling procedures with different underlying generative processes than the one we described in Sec. II, as long as they provide some initial d, k sequences.

Before presenting our method, we make some remarks. First, sampling separately d, k assumes the approximation $p(d, k) \approx p(d)\,p(k)$. The matching procedure mitigates the impact of this approximation, as it modifies one of the two sequences should they not match. Second, it is possible to start the MCMC directly from the hyperedge configuration of a real hypergraph, as we illustrate in our experiments with real data in Sec. V. This effectively corresponds to fixing the sequences d, k to those of the data, which are necessarily consistent. These are then preserved during MCMC while the hyperedges are mixed. Hence, in this case there is no need for a matching algorithm. We thus assume that a user would like to start from a desired set of values for both quantities, without worrying about their consistency.

We now describe our proposed matching algorithm. If there exists at least one hypergraph with sequences d, k, then we

Algorithm 2. Hyperedge construction from sequences.

```
Input: Degree sequence $d$, size sequence $k$, priority sequence
        choice $pr$ $\in$ {degseq, dimseq}
Result: List of hyperedges $L$
 > Iterate over hyperdge sizes and specified number
 of such hyperedges
L = [ ]
for $s = 2, \dots, D$ do
    for $j = 1, \dots, k_i$ do
        hye = ExtractHye($s$, $d$, $k$, $pr$)
        if $len(hye) > 1$ then
            L ← L + [hye]
        end
    end
end
 > If the priority is the degree sequence, have all
   nodes reach the required degree
 > Call GeTwo the function counting the number of
   nodes with nonzero degrees in the sequence
if $pr = degseq$ then
 > While there are at least two nodes with nonzero
   remaining degree, produce hyperedges
    while $GeTwo(d) \geqslant 2$ do
        maxdim ← min(GeTwo($d$), D)
        $s$ ← random{2, ..., maxdim}
        hye ← ExtractHye($s$, $d$, $k$, $pr$)
        L ← L + [hye]
    end
end
```

call d and k *compatible*. Our algorithm guarantees that, if d, k are compatible, one hypergraph with such sequences will be produced. If they are not compatible, however —as is the case for most samples from Appendix B 1—then only one of them can be preserved. For this reason, the user is required to specify which sequence needs to be preserved; we refer to this as the priority sequence. The algorithm dynamically modifies d, k until exhaustion of the priority sequence, if the sequences are compatible then no modification to the other sequence will be made, as they will terminate together. Intuitively, our algorithm works as follows. It extracts a hyperedge of a given size from the nodes with the highest available degrees, until exhaustion of the size sequence k. If k is the priority sequence and no nodes are available, then random nodes are extracted to satisfy the required hyperedge sizes, otherwise smaller hyperedges might be produced. After exhaustion of the size sequence, if d is still not exhausted, but it is the priority sequence, then keep drawing hyperedges to exhaust d (even if k is not preserved). The function described in Algorithm 3 draws the hyperedges and prioritizes d or k depending on the priority sequence. We present a pseudocode description of the matching algorithm in Algorithm 2.

APPENDIX F: GENERATION OF THE SYNTHETIC DATA

1. Synthetic data with varying properties

We include additional details about the data and plots generated in Secs. IV A and IV B.

Algorithm 3. ExtractHye.

```
Input: Hyperedeg size $s$, degree sequence $d$, size sequence $k$,
       priority sequence choice $pr$ $\in$ {degseq, dimseq}
Result: A hyperedge hye.
          The input $d$ is modified in place.
 > Extract one hyperedge while preserving the
   degree or size sequence
if $pr = degseq$ then
    hye ← {$s$ nodes $v$ with the highest $d_v$ values.
              If not enough nodes satisfy $d_v > 0$, possibly return
              less than $s$ nodes}
end
else if $pr = dimseq$ then
    hye ← {$s$ nodes $v$ with the highest $d_v$ values.
              If not enough nodes satisfy $d_v > 0$,
              select some random nodes with $d_v^b = 0$
              untill $s$ nodes are chosen.}
end
 > Update the degree sequence by decreasing the
   degree of the nodes selected
for $v$ in hye do
    if $d_v > 0$ then
        $d_v \leftarrow d_v - 1$
    end
end
```

For the plots in Sec. IV A, we consider five equally sized communities with a total of $N = 100$ nodes and a maximum hyperedge size $D = 10$. For a node belonging to community 1 (and similarly for nodes belonging to other communities 2, 3, 4, and 5) we consider an initial hard assignment $u_i^{\text{hard}} = (1, 0, 0, 0, 0)$ and parametrize with α a convex combination given by

$$u_i^{\alpha} := \alpha\, u_i^{\text{hard}} + (1 - \alpha)\,(1, 1, 1, 1, 1).$$

In all plots in Fig. 2, the x axis represents different decreasing values of α: the value $\alpha = 1$ represents a hard assignment, and decreasing values move the membership towards a soft membership. Notice also that, for every node i, the maximum community assignment doesn't change with α: if $u_i^{\text{hard}} = (1, 0, 0, 0, 0)$, then $\arg\max_k u_{ik}^{\alpha} = 1$ for all $\alpha > 0$ (and similarly for nodes belonging to other communities). The affinity matrix w is the diagonal matrix of all ones, rescaled by a scalar to keep the average degree fixed to 5 across configurations of α.

For the plot in Fig. 2(a), we consider the strength of the interactions within and between communities. Every one of the five vertices represents one community. The thickness of the edges between communities a and b is proportional to the following quantity:

$$T(a, b) := \sum_{e \in E} \frac{A_e}{\binom{e}{2}} \sum_{i<j \in e} \delta_{i \in a} \delta_{j \in b},$$

where $\delta_{i \in a} = 1$ when node i belongs to community a, defined as above $a = \arg\max_k u_{ik}$.

The thickness of the circles surrounding the five vertices corresponds to the within-community interactions, i.e., $T(a, a)$ for every community a.

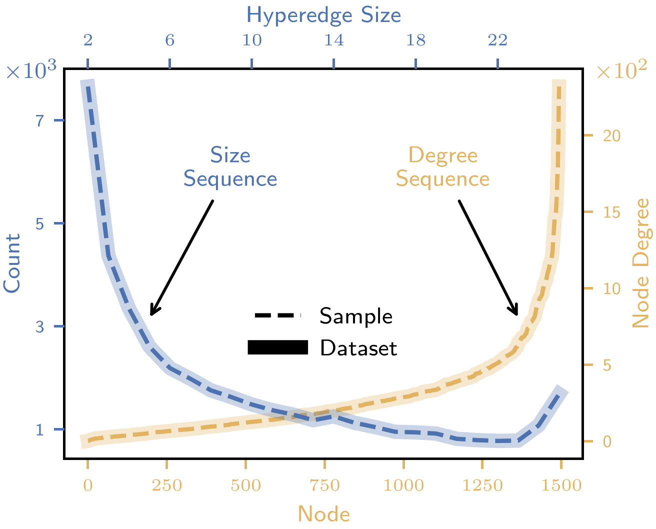

FIG. 8. Correspondence of the degree and size sequences between data and samples. We check for the correspondence between the sequences of the samples and the original House Bills dataset, which is utilized to initialize the MCMC procedure. Due to the properties of the reshuffling operator, the sequences need to coincide.

2. Synthetic benchmark data

We include additional details for the generation of the data utilized in Sec. IV C. We generate data with $N = 500$ nodes, $K = 3$ equally sized communities, and hard community assignments. The adjacency matrix w is the $K \times K$ identity matrix. Additionally, we condition on a size sequence, i.e., the count of hyperedges per hyperedge dimension, given by: `{2: 500, 3: 400, 4: 400, 5: 400, 6: 600, 7: 700, 8: 800, 9: 900, 10: 1000, 11: 1100, 12: 1200, 13: 1300, 14: 1400, 15: 1500}`. The expected degree resulting from such a sequence is 248.6. Like for other experiments, we utilize the default MCMC configuration of $n_b = 100\,000$ burn-in steps and $n_i = 20\,000$ steps between samples.

APPENDIX G: MATCHING SEQUENCES ON THE HOUSE BILLS DATASET

In Fig. 8 we show the perfect correspondence between the degree and size sequences as observed in the real data and in the samples. As we initialize our sampling procedure directly from the hyperedge configuration of the real data, such correspondence is guaranteed by the properties of the reshuffling operator.

APPENDIX H: STRUCTURAL MEASURES FROM CONFIGURATION MODEL

For comparison, here we run the experiments in Sec. V B, but create samples via the hypergraph configuration model from Chodrow [53], as opposed to our method. The configuration model takes a dataset and mixes its hyperedges preserving the initial degree and size sequences. In this sense, it can be seen as a less structured version of the method we propose here. In Fig. 9, we show the adjacency matrix, hyperedge inclusions, hyperedge eigenvector centrality, and subhypergraph centrality computed on samples based on the House

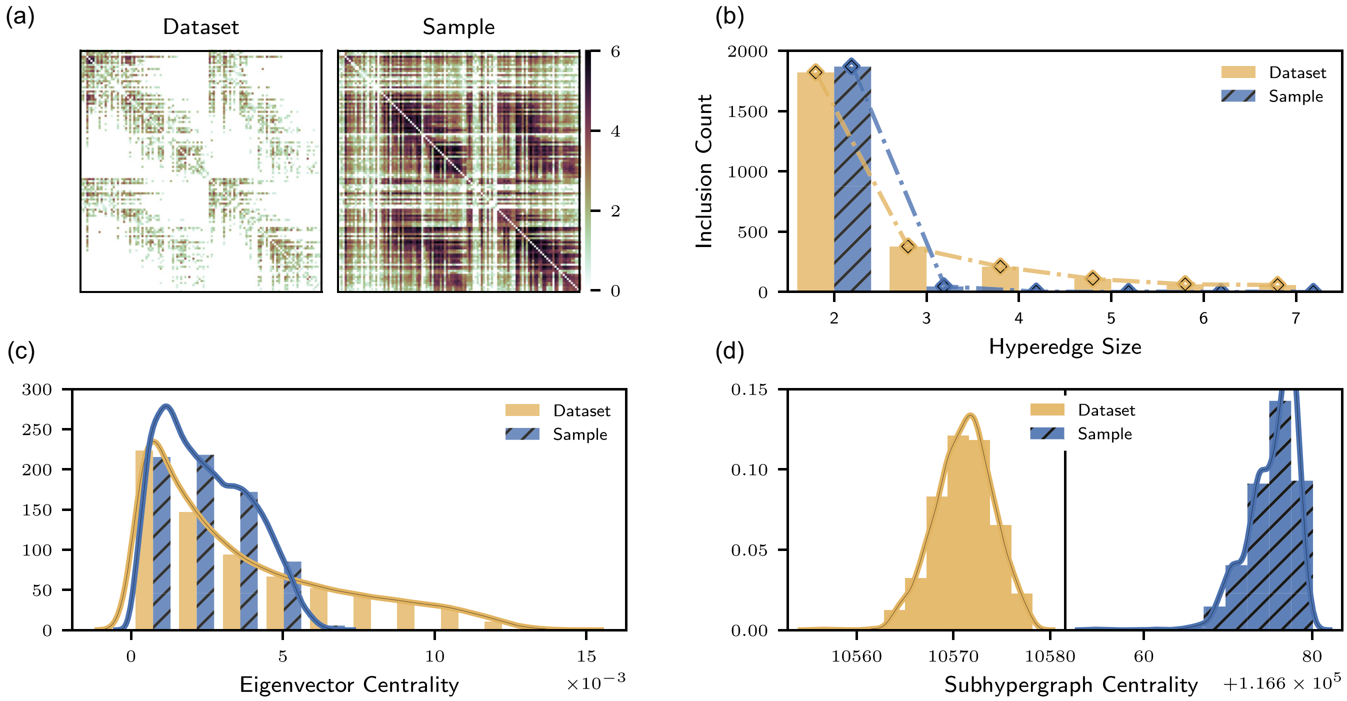

FIG. 9. Comparing the statistics on real data and samples obtained from the configuration model. We plot (a) the adjacency matrices, (b) the hyperedge inclusions occurrences, (c) the hyperedge eigenvector centrality distribution, and (d) the subhypergraph centrality distribution for the House Bills dataset. Here, samples are obtained via the hypergraph configuration model [53]. Due to less structure being incorporated into the sampling procedure, samples and real data present substantial differences.

Bills dataset. As can be observed, the samples obtained via the configuration model have less resemblance to the original dataset.

APPENDIX I: HYPOTHESIS TESTING ON THE HOUSE BILLS DATASET

Synthetic samples allow carrying out statistical hypothesis testing more formally. A common example is that of testing whether modeling explicitly one or more structural characteristics of the data allows explaining also other macroscopic features that are not explicitly considered in the modeling assumptions. We provide an example of such an application here.

We consider the case of the House Bills dataset from Sec. V B, and generate synthetic samples by explicitly modeling the community structure (represented by the inferred affinity matrix and membership vectors) and the degree and size sequences. We test whether such features help to explain also one that is not explicitly controlled: The average shortest path length between any two nodes. To perform testing, we collect the statistics from the generated samples, and perform a Student's t test to compare the sample population with the real data. The p value of 1.5×10^{-4} rejects the null hypothesis, implying that the average shortest path in the real data falls outside the population observed in the samples.

Similarly, we proceed testing for similarity in the hyperedge eigenvector centrality between the data and the samples, represented in Fig. 6(c). Here, we compute the overlap between the histograms of hyperedge eigenvector centralities from the samples and the real data, and proceed similarly by computing the overlap between pairs of samples. Comparing the two populations obtained yields a p value of 0.037: when considering a confidence interval of 99%, we cannot reject the null hypothesis that these populations have the same mean, confirming that the hyperedge eigenvector centrality is approximately recovered in the synthetic samples.

[1] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, Complex networks: Structure and dynamics, Phys. Rep. **424**, 175 (2006).

[2] D. J. Watts and S. H. Strogatz, Collective dynamics of "small-world" networks, Nature (London) **393**, 440 (1998).

[3] V. Latora and M. Marchiori, Efficient behavior of small-world networks, Phys. Rev. Lett. **87**, 198701 (2001).

[4] A.-L. Barabási and R. Albert, Emergence of scaling in random networks, Science **286**, 509 (1999).

[5] S. Fortunato, Community detection in graphs, Phys. Rep. **486**, 75 (2010).

[6] M. E. Newman and M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E **69**, 026113 (2004).

[7] A. Lancichinetti, S. Fortunato, and F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E **78**, 046110 (2008).

[8] S. Fortunato and D. Hric, Community detection in networks: A user guide, Phys. Rep. **659**, 1 (2016).

[9] A. Arenas, A. Diaz-Guilera, and C. J. Pérez-Vicente, Synchronization reveals topological scales in complex networks, Phys. Rev. Lett. **96**, 114102 (2006).

[10] A. Nematzadeh, E. Ferrara, A. Flammini, and Y.-Y. Ahn, Optimal network modularity for information diffusion, Phys. Rev. Lett. **113**, 088701 (2014).

[11] M. Coscia and L. Rossi, How minimizing conflicts could lead to polarization on social media: An agent-based model investigation, PLoS ONE **17**, e0263184 (2022).

[12] C. Bordier, C. Nicolini, and A. Bifone, Graph analysis and modularity of brain functional connectivity networks: Searching for the optimal threshold, Front. Neurosci. **11**, 441 (2017).

[13] D. Lusher, J. Koskinen, and G. Robins, *Exponential Random Graph Models for Social Networks: Theory, Methods, and Applications* (Cambridge University Press, Cambridge, UK, 2013).

[14] E. A. Hobson, M. J. Silk, N. H. Fefferman, D. B. Larremore, P. Rombach, S. Shai, and N. Pinter-Wollman, A guide to choosing and implementing reference models for social network analysis, Biol. Rev. **96**, 2716 (2021).

[15] F. Battiston, G. Cencetti, I. Iacopini, V. Latora, M. Lucas, A. Patania, J.-G. Young, and G. Petri, Networks beyond pairwise interactions: Structure and dynamics, Phys. Rep. **874**, 1 (2020).

[16] L. Torres, A. S. Blevins, D. Bassett, and T. Eliassi-Rad, The why, how, and when of representations for complex systems, SIAM Rev. **63**, 435 (2021).

[17] F. Battiston, E. Amico, A. Barrat, G. Bianconi, G. Ferraz de Arruda, B. Franceschiello, I. Iacopini, S. Kéfi, V. Latora, Y. Moreno *et al.*, The physics of higher-order interactions in complex systems, Nat. Phys. **17**, 1093 (2021).

[18] F. Battiston and G. Petri, *Higher-Order Systems* (Springer, Berlin, 2022).

[19] G. Petri, P. Expert, F. Turkheimer, R. Carhart-Harris, D. Nutt, P. J. Hellyer, and F. Vaccarino, Homological scaffolds of brain functional networks, J. R. Soc., Interface **11**, 20140873 (2014).

[20] C. Giusti, R. Ghrist, and D. S. Bassett, Two's company, three (or more) is a simplex, J. Comput. Neurosci. **41**, 1 (2016).

[21] A. Santoro, F. Battiston, G. Petri, and E. Amico, Higher-order organization of multivariate time series, Nat. Phys. **19**, 221 (2023).

[22] A. Patania, G. Petri, and F. Vaccarino, The shape of collaborations, EPJ Data Sci. **6**, 18 (2017).

[23] J. Grilli, G. Barabás, M. J. Michalska-Smith, and S. Allesina, Higher-order interactions stabilize dynamics in competitive network models, Nature (London) **548**, 210 (2017).

[24] S. Klamt, U.-U. Haus, and F. Theis, Hypergraphs and cellular networks, PLoS Comput. Biol. **5**, e1000385 (2009).

[25] A. Zimmer, I. Katzir, E. Dekel, A. E. Mayo, and U. Alon, Prediction of multidimensional drug dose responses based on measurements of drug pairs, Proc. Natl. Acad. Sci. USA **113**, 10442 (2016).

[26] G. Cencetti, F. Battiston, B. Lepri, and M. Karsai, Temporal properties of higher-order interactions in social networks, Sci. Rep. **11**, 7028 (2021).

[27] F. Musciotto, D. Papageorgiou, F. Battiston, and D. R. Farine, Beyond the dyad: Uncovering higher-order

structure within cohesive animal groups, bioRxiv (2022), doi:10.1101/2022.05.30.494018.

[28] C. Bick, P. Ashwin, and A. Rodrigues, Chaos in generically coupled phase oscillator networks with nonpairwise interactions, Chaos **26**, 094814 (2016).

[29] P. S. Skardal and A. Arenas, Higher order interactions in complex networks of phase oscillators promote abrupt synchronization switching, Commun. Phys. **3**, 218 (2020).

[30] A. P. Millán, J. J. Torres, and G. Bianconi, Explosive higher-order Kuramoto dynamics on simplicial complexes, Phys. Rev. Lett. **124**, 218301 (2020).

[31] M. Lucas, G. Cencetti, and F. Battiston, Multiorder Laplacian for synchronization in higher-order networks, Phys. Rev. Res. **2**, 033410 (2020).

[32] L. V. Gambuzza, F. Di Patti, L. Gallo, S. Lepri, M. Romance, R. Criado, M. Frasca, V. Latora, and S. Boccaletti, Stability of synchronization in simplicial complexes, Nat. Commun. **12**, 1255 (2021).

[33] I. Iacopini, G. Petri, A. Barrat, and V. Latora, Simplicial models of social contagion, Nat. Commun. **10**, 2485 (2019).

[34] S. Chowdhary, A. Kumar, G. Cencetti, I. Iacopini, and F. Battiston, Simplicial contagion in temporal higher-order networks, J. Phys. Complex. **2**, 035019 (2021).

[35] L. Neuhäuser, A. Mellor, and R. Lambiotte, Multibody interactions and nonlinear consensus dynamics on networked systems, Phys. Rev. E **101**, 032310 (2020).

[36] M. T. Schaub, A. R. Benson, P. Horn, G. Lippner, and A. Jadbabaie, Random walks on simplicial complexes and the normalized Hodge 1-Laplacian, SIAM Rev. **62**, 353 (2020).

[37] T. Carletti, F. Battiston, G. Cencetti, and D. Fanelli, Random walks on hypergraphs, Phys. Rev. E **101**, 022308 (2020).

[38] U. Alvarez-Rodriguez, F. Battiston, G. F. de Arruda, Y. Moreno, M. Perc, and V. Latora, Evolutionary dynamics of higher-order interactions in social networks, Nat. Hum. Behav. **5**, 586 (2021).

[39] A. Civilini, N. Anbarci, and V. Latora, Evolutionary game model of group choice dilemmas on hypergraphs, Phys. Rev. Lett. **127**, 268301 (2021).

[40] C. Berge, *Graphs and Hypergraphs* (Elsevier Science Ltd., Oxford, 1985).

[41] A. R. Benson, Three hypergraph eigenvector centralities, SIAM J. Math. Data Sci. **1**, 293 (2019).

[42] F. Tudisco and D. J. Higham, Node and edge nonlinear eigenvector centrality for hypergraphs, Commun. Phys. **4**, 201 (2021).

[43] A. R. Benson, R. Abebe, M. T. Schaub, A. Jadbabaie, and J. Kleinberg, Simplicial closure and higher-order link prediction, Proc. Natl. Acad. Sci. USA **115**, E11221 (2018).

[44] Q. F. Lotito, F. Musciotto, A. Montresor, and F. Battiston, Higher-order motif analysis in hypergraphs, Commun. Phys. **5**, 79 (2022).

[45] F. Musciotto, F. Battiston, and R. N. Mantegna, Detecting informative higher-order interactions in statistically validated hypergraphs, Commun. Phys. **4**, 218 (2021).

[46] M. Contisciani, F. Battiston, and C. De Bacco, Inference of hyperedges and overlapping communities in hypergraphs, Nat. Commun. **13**, 7229 (2022).

[47] J.-G. Young, G. Petri, and T. P. Peixoto, Hypergraph reconstruction from network data, Commun. Phys. **4**, 135 (2021).

[48] T. Carletti, D. Fanelli, and R. Lambiotte, Random walks and community detection in hypergraphs, J. Phys.: Complex. **2**, 015011 (2021).

[49] A. Eriksson, D. Edler, A. Rojas, M. de Domenico, and M. Rosvall, How choosing random-walk model and network representation matters for flow-based community detection in hypergraphs, Commun. Phys. **4**, 133 (2021).

[50] P. S. Chodrow, N. Veldt, and A. R. Benson, Generative hypergraph clustering: From blockmodels to modularity, Sci. Adv. **7**, eabh1303 (2021).

[51] O. T. Courtney and G. Bianconi, Generalized network structures: The configuration model and the canonical ensemble of simplicial complexes, Phys. Rev. E **93**, 062311 (2016).

[52] J.-G. Young, G. Petri, F. Vaccarino, and A. Patania, Construction of and efficient sampling from the simplicial configuration model, Phys. Rev. E **96**, 032312 (2017).

[53] P. S. Chodrow, Configuration models of random hypergraphs, J. Complex Netw. **8**, cnaa018 (2020).

[54] M. Barthelemy, Class of models for random hypergraphs, Phys. Rev. E **106**, 064310 (2022).

[55] K. Kovalenko, I. Sendiña-Nadal, N. Khalil, A. Dainiak, D. Musatov, A. M. Raigorodskii, K. Alfaro-Bittner, B. Barzel, and S. Boccaletti, Growing scale-free simplices, Commun. Phys. **4**, 43 (2021).

[56] A. P. Millán, R. Ghorbanchian, N. Defenu, F. Battiston, and G. Bianconi, Local topological moves determine global diffusion properties of hyperbolic higher-order networks, Phys. Rev. E **104**, 054302 (2021).

[57] P. Krapivsky, Random recursive hypergraphs, J. Phys. A: Math. Theor. **56**, 195001 (2023).

[58] J. Lerner, M. Tranmer, J. Mowbray, and M.-G. Hancean, Rem beyond dyads: Relational hyperevent models for multi-actor interaction networks, arXiv:1912.07403.

[59] G. Robins, P. Pattison, Y. Kalish, and D. Lusher, An introduction to exponential random graph (p*) models for social networks, Soc. Netw. **29**, 173 (2007).

[60] J. Park and M. E. J. Newman, Statistical mechanics of networks, Phys. Rev. E **70**, 066117 (2004).

[61] J. Mulder and P. D. Hoff, A latent variable model for relational events with multiple receivers, arXiv:2101.05135.

[62] C. De Bacco, E. A. Power, D. B. Larremore, and C. Moore, Community detection, link prediction, and layer interdependence in multilayer networks, Phys. Rev. E **95**, 042317 (2017).

[63] A. Schein, J. Paisley, D. M. Blei, and H. Wallach, in *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery, New York, 2015), pp. 1045–1054.

[64] T. G. Kolda and B. W. Bader, Tensor decompositions and applications, SIAM Rev. **51**, 455 (2009).

[65] D. D. Lee and H. S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature (London) **401**, 788 (1999).

[66] B. Karrer and M. E. Newman, Stochastic blockmodels and community structure in networks, Phys. Rev. E **83**, 016107 (2011).

[67] N. Ruggeri, M. Contisciani, F. Battiston, and C. De Bacco, Generalized inference of mesoscale structures in higher-order networks, Sci. Adv. **9**, eadg9159 (2023).

[68] S. L. Hakimi, On realizability of a set of integers as degrees of the vertices of a linear graph. i, J. Soc. Ind. Appl. Math. **10**, 496 (1962).

[69] S. A. Choudum, A simple proof of the Erdos-Gallai theorem on graph sequences, Bull. Aust. Math. Soc. **33**, 67 (1986).

[70] W. K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, Biometrika **57**, 97 (1970).

[71] github.com/nickruggeri/Hy-MMSBM.

[72] P. L. Erdös and I. Miklós, Not all simple looking degree sequence problems are easy, J. Combin. **9**, 553 (2018).

[73] P. Chodrow, N. Eikmeier, and J. Haddock, Nonbacktracking spectral clustering of nonuniform hypergraphs, SIAM J. Math. Data Sci. **5**, 251 (2023).

[74] D. Zhou, J. Huang, and B. Schölkopf, Learning with hypergraphs: Clustering, classification, and embedding, Adv. Neural Inf. Process. Syst. **19**, 1601 (2006).

[75] U. Dutta, Sampling random graphs with specified degree sequences, Ph.D. thesis, University of Colorado at Boulder, 2022.

[76] J. H. Fowler, Connecting the congress: A study of cosponsorship networks, Polit. Anal. **14**, 456 (2006).

[77] J. H. Fowler, Legislative cosponsorship networks in the U.S. house and senate, Soc. Netw. **28**, 454 (2006).

[78] E. Estrada and J. A. Rodríguez-Velázquez, Subgraph centrality and clustering in complex hyper-networks, Physica A **364**, 581 (2006).

[79] Y. Zhang, M. Lucas, and F. Battiston, Do higher-order interactions promote synchronization? Nat. Commun. **14**, 1605 (2023).

[80] F. Baccini, F. Geraci, and G. Bianconi, Weighted simplicial complexes and their representation power of higher-order network data and topology, Phys. Rev. E **106**, 034319 (2022).

[81] P. Bonacich, Factoring and weighting approaches to status scores and clique identification, J. Math. Soc. **2**, 113 (1972).

[82] A. Bretto, *Hypergraph theory*, *An Introduction. Mathematical Engineering* (Springer, Cham, 2013).

[83] R. Mastrandrea, J. Fournet, and A. Barrat, Contact patterns in a high school: A comparison between data collected using wearable sensors, contact diaries and friendship surveys, PLoS ONE **10**, e0136497 (2015).

[84] I. Amburg, N. Veldt, and A. Benson, *Clustering in Graphs and Hypergraphs with Categorical Edge Labels* (ACM, New York, NY, 2020), pp. 706–717.

[85] C. Stewart III and J. Woon, Congressional committee assignments, 103rd to 114th congresses, 1993–2017: House technical report, MIT (2008).

[86] D. R. Hunter, S. M. Goodreau, and M. S. Handcock, Goodness of fit of social network models, J. Am. Stat. Assoc. **103**, 248 (2008).

[87] H. J. Spaeth, L. Epstein, A. D. Martin, J. A. Segal, T. J. Ruger, and S. C. Benesh, 2022 supreme court database, version 2020 release 1, http://Supremecourtdatabase.org (2020).

[88] M. Contisciani, E. A. Power, and C. De Bacco, Community detection with node attributes in multilayer networks, Sci. Rep. **10**, 15736 (2020).

[89] M. E. Newman and A. Clauset, Structure and inference in annotated networks, Nat. Commun. **7**, 11863 (2016).

[90] X. Zhang, C. Moore, and M. E. Newman, Random graph models for dynamic networks, Eur. Phys. J. B **90**, 200 (2017).

[91] H. Safdari, M. Contisciani, and C. De Bacco, Reciprocity, community detection, and link prediction in dynamic networks, J. Phys.: Complex. **3**, 015010 (2022).

[92] A. Badalyan, N. Ruggeri, and C. De Bacco, Hypergraphs with node attributes: structure and inference, arXiv:2311.03857.

[93] Q. F. Lotito, M. Contisciani, C. De Bacco, L. Di Gaetano, L. Gallo, A. Montresor, F. Musciotto, N. Ruggeri, and F. Battiston, Hypergraphx: A library for higher-order network analysis, J. Complex Netw. **11**, cnad019 (2023).

[94] S. Pal and Y. Zhu, Community detection in the sparse hypergraph stochastic block model, Random Struct. Algor. **59**, 407 (2021).

[95] S. M. Ross, *Simulation* (Academic Press, San Diego, CA, 2022).

[96] P. Billingsley, *Probability and Measure* (John Wiley & Sons, New York, NY, 2008).

[97] https://github.com/PhilChodrow/hypergraph.