NETWORK SCIENCE

Generative hypergraph clustering: From blockmodels to modularity

Philip S. Chodrow[1]*, Nate Veldt[2], Austin R. Benson[3]

Hypergraphs are a natural modeling paradigm for networked systems with multiway interactions. A standard task in network analysis is the identification of closely related or densely interconnected nodes. We propose a probabilistic generative model of clustered hypergraphs with heterogeneous node degrees and edge sizes. Approximate maximum likelihood inference in this model leads to a clustering objective that generalizes the popular modularity objective for graphs. From this, we derive an inference algorithm that generalizes the Louvain graph community detection method, and a faster, specialized variant in which edges are expected to lie fully within clusters. Using synthetic and empirical data, we demonstrate that the specialized method is highly scalable and can detect clusters where graph-based methods fail. We also use our model to find interpretable higher-order structure in school contact networks, U.S. congressional bill cosponsorship and committees, product categories in copurchasing behavior, and hotel locations from web browsing sessions.

INTRODUCTION

Graphs are a fundamental abstraction for complex relational systems throughout the sciences (*1*–*3*). A graph represents components of a system by a set of nodes and represents interactions or relationships among these components using edges that connect pairs of nodes. Much of the structure in complex data, however, involves higher-order interactions and relationships between more than two entities at once (*4*–*9*). Hypergraphs are now a burgeoning paradigm for modeling these and many other systems (*10*–*13*). A hypergraph still represents the components by a set of nodes, but the edges (often called hyperedges) may connect arbitrary numbers of nodes. A graph is a special case of a hypergraph, in which each edge connects exactly two nodes.

Graph clustering is a fundamental task in network science that seeks to describe large graphs by dividing their nodes into closely related or interconnected groups (also called clusters or communities) (*5*, *10*, *14*, *15*). Clustering methods for hypergraphs have applications in parallel computation (*16*, *17*), circuit design (*18*), image segmentation (*19*), semisupervised learning (*20*, *21*), and higher-order network analysis of gene expression (*22*), food webs (*23*), and online social communities (*24*, *25*).

A well-established graph clustering approach is to model the graph as a sample from a probabilistic generative model, in which case the clustering task can be recast as a statistical inference problem (*26*–*32*). While generative modeling is a mainstay in graph clustering, generative techniques for hypergraphs are largely lacking. While a small number of generative models of clustered hypergraphs have been proposed (*33*–*36*), these models typically generate hypergraphs with edges of only one size. With a recent exception (*36*), these models also do not model degree heterogeneity between nodes. Heterogeneity in edge size and node degree are both key features of empirical data (*6*), and their omission limits the applicability of many of these models for practical data analysis. An alternative to generative hypergraph modeling is to transform the hypergraph into a dyadic graph via clique expansion, where a dyadic edge connects any pair of nodes that appear together in some hyperedge (*5*, *20*). While this enables the use of a wide array of existing models and algorithms for graphs, the higher-order structure is lost (*37*), and generative models of the resulting dyadic graph may rely on explicitly violated independence assumptions. Recently, nongenerative approaches based on the popular modularity clustering objective for graphs (*38*) have been proposed for hypergraphs (*39*–*41*), although their lack of connection to a generative model limits their interpretability.

Another approach to generative clustering is to use the representation of a hypergraph as a bipartite graph and apply a generative model [e.g., (*42*–*44*)] to the latter representation. This approach, while appropriate in many datasets, involves a strong assumption: The memberships of any two nodes in a given hyperedge are independent, conditional on the model parameters. This assumption is natural for certain classes of data. For example, consider an event coattendance network, with nodes representing music enthusiasts and hyperedges representing concerts. Node membership in a hyperedge corresponds to attendance at the specified event. To a reasonable approximation, the decision of two fans to attend a given concert may indeed be independent, conditioned on the popularity of the performers, the location of the venue, and so on. In other datasets, however, the conditional independence assumption is explicitly violated. Multiway social interaction networks give one important class of examples. Interactions such as gossip, for instance, normally take place only between trusted individuals. The presence of a single uninvited outsider may entirely prevent the interaction from taking place. The "all-or-nothing" (AON) structure of these interactions is an important violation of the conditional independence assumptions made by most bipartite generative models. These examples highlight that the task of matching assumptions to higher-order data is an ongoing challenge, for which we benefit from a diversity of distinct tools.

Here, we propose a generative approach to hypergraph clustering based on a degree-corrected hypergraph stochastic blockmodel (DCHSBM). This model generates clustered hypergraphs with heterogeneous degree distributions and hyperedge sizes. We outline an approximate coordinate-ascent maximum likelihood estimation scheme for fitting this model to hypergraph data and show that one stage of this scheme generalizes the well-studied modularity objective for graphs. We derive accompanying Louvain algorithms for

[1]Department of Mathematics, University of California, Los Angeles, 520 Portola Plaza, Los Angeles, CA 90095, USA. [2]Center for Applied Mathematics, Cornell University, 657 Frank H.T. Rhodes Hall, Ithaca, NY 14853, USA. [3]Department of Computer Science, Cornell University, 413B Gates Hall, Ithaca, NY 14853, USA.
*Corresponding author. Email: phil@math.ucla.edu

this class of modularity-like objectives, which are highly scalable in an important special case. We show computationally that hypergraph clustering methods are able to detect planted clusters in regimes in which graph-based methods necessarily fail because of known theoretical limits. We also show that, in datasets with appropriately matched higher-order structure, our generative hypergraph techniques are able to recover clusters correlated to metadata at higher rates than graph-based techniques. Our results highlight the importance of matching generative models to datasets and point toward a number of directions for further work in higher-order network science.

MATERIALS AND METHODS

The DCHSBM

The degree-corrected stochastic blockmodel (DCSBM) is a generative model of graphs with both community structure and heterogeneous degree sequences (*29*). We now extend this model to the case of hypergraphs.

For our model, let n be the number of nodes in a hypergraph. Each node i is assigned to one of $\bar{\ell}$ groups. We let $z_i \in [\bar{\ell}] = \{1, 2, \ldots, \bar{\ell}\}$ denote the group assignment of node i and collect these assignments in a vector $\mathbf{z} \in [\bar{\ell}]^n$. As in the dyadic DCSBM, each node i is assigned a parameter θ_i governing its degree, and we collect these parameters in a vector $\boldsymbol{\theta} \in \mathbb{R}^n$. Let $\mathcal{R}$ represent the set of unordered node tuples, so that each $R \in \mathcal{R}$ is a set of nodes representing the location of a possible hyperedge (following the standard choice for the DCSBM in graphs, we allow $\mathcal{R}$ to include node tuples with repeated nodes). Let $\mathbf{z}_R$ denote the vector of cluster labels for nodes in a given tuple R, and $\boldsymbol{\theta}_R$ the vector of degree parameters.

We use an affinity function Ω to control the probability of placing a hyperedge at a given node tuple R, which depends on the group memberships of the nodes in R. Formally, Ω maps the group assignments $\mathbf{z}_R$ to a non-negative number. If $\Omega(\mathbf{z}_R)$ is large, then there is a higher probability that a hyperedge forms between the nodes in R. In our model, the number of hyperedges placed at $R \in \mathcal{R}$ is distributed as $a_R \sim \text{Poisson}(b_R \pi(\boldsymbol{\theta}_R) \Omega(\mathbf{z}_R))$, where b_R denotes the number of distinct ways to order the nodes of R and $\pi(\boldsymbol{\theta}_R) = \prod_{i \in R} \theta_i$ is the product of degree parameters. The probability of realizing a given value a_R is then

$$\mathrm{P}(a_R | \mathbf{z}, \Omega, \boldsymbol{\theta}) = \frac{e^{-b_R \pi(\boldsymbol{\theta}_R) \Omega(\mathbf{z}_R)} (b_R \pi(\boldsymbol{\theta}_R) \Omega(\mathbf{z}_R))^{a_R}}{a_R!} \tag{1}$$

This edge generation process has the following intuitive interpretation: For each of b_R possible orderings of nodes in R, we attempt to place a Poisson $(\pi(\boldsymbol{\theta}_R)\Omega(\mathbf{z}_R))$ number of hyperedges on this tuple. The result is a weighted hyperedge on the unordered tuple R, whose weight can be any non-negative integer. This is a helpful modeling feature, as many empirical hypergraphs contain multiple hyperedges between the same set of nodes. Even in hypergraph datasets where we only know the presence or absence of hyperedges (but no weights), the Poisson-based model serves as a computationally convenient approximation to a Bernoulli-based model. The probability of realizing a given hyperedge set $\mathbf{A} = (a_R)_{R \in \mathcal{R}}$ is then just the product of probabilities over each $R \in \mathcal{R}$.

Estimation of degree and affinity parameters

There are many methods for inference in general stochastic blockmodels (SBMs) and their relatives, including variational coordinate ascent (*28*), variational belief propagation (*45*, *46*), and Markov Chain Monte Carlo (*26*, *47*). We perform approximate maximum likelihood inference via coordinate ascent. We do so to exploit a recent connection between maximum likelihood inference in the DCSBM and the popular modularity objective for graph clustering (*48*). Our coordinate ascent framework, in which we alternate between estimating parameters and node labels, is a close relative of expectation-maximization (EM) algorithms for blockmodel inference (*45*). Standard versions of EM construct "soft" clusters, in which each node is given a weighted assignment in every possible cluster. "The" cluster for a given node is often taken to be the cluster in which the node has largest weight. In contrast, our approach generates "hard" clusters in which each node belongs to exactly one cluster. Profile likelihood methods offer an alternative framework for maximum likelihood inference (*49*), and their development for hypergraphs is another promising avenue of future work.

In the maximum likelihood framework, we learn estimates $\hat{\mathbf{z}}$ of the node labels, $\widehat{\Omega}$ of the affinity function, and $\hat{\boldsymbol{\theta}}$ of the degree parameters by solving the optimization problem

$$\hat{\mathbf{z}}, \widehat{\Omega}, \hat{\boldsymbol{\theta}} \equiv \underset{\mathbf{z}, \Omega, \boldsymbol{\theta}}{\operatorname{argmax}} \mathrm{P}(\mathbf{A} | \mathbf{z}, \Omega, \boldsymbol{\theta}) \tag{2}$$

where $\mathbf{A}$ is a given dataset represented by a collection of (integer-weighted) hyperedges. As usual, it is easier to work with the log-likelihood, which has the same local optima. The log-likelihood is

$$\mathcal{L}(\mathbf{z}, \Omega, \boldsymbol{\theta}) = \sum_{R \in \mathcal{R}} \log \mathrm{P}(a_R | \mathbf{z}, \Omega, \boldsymbol{\theta}) = Q(\mathbf{z}, \Omega, \boldsymbol{\theta}) + K(\boldsymbol{\theta}) + C \tag{3}$$

where

$$Q(\mathbf{z}, \Omega, \boldsymbol{\theta}) \equiv \sum_{R \in \mathcal{R}} [a_R \log \Omega(\mathbf{z}_R) - b_R \pi(\boldsymbol{\theta}_R) \Omega(\mathbf{z}_R)] \tag{4}$$

$$K(\boldsymbol{\theta}) \equiv \sum_{R \in \mathcal{R}} a_R \log \pi(\boldsymbol{\theta}_R) \tag{5}$$

$$C \equiv \sum_{R \in \mathcal{R}} [a_R \log b_R - \log a_R!] \tag{6}$$

The first term $Q(\mathbf{z}, \Omega, \boldsymbol{\theta})$ is the only part of the log-likelihood that depends on the group assignments $\mathbf{z}$ and affinity function Ω. The second term depends on $\boldsymbol{\theta}$, while the third term depends only on the data $\mathbf{A}$ and can be disregarded for inferential purposes.

In the coordinate ascent approach to maximum likelihood, we alternate between two stages. In the first stage, we assume a current estimate $\hat{\mathbf{z}}$ and obtain new estimates of Ω and $\boldsymbol{\theta}$ by solving

$$\widehat{\Omega}, \hat{\boldsymbol{\theta}} = \underset{\Omega, \boldsymbol{\theta}}{\operatorname{argmax}} \mathcal{L}(\hat{\mathbf{z}}, \Omega, \boldsymbol{\theta}) \tag{7}$$

The resulting pair $\widehat{\Omega}, \hat{\boldsymbol{\theta}}$ can be viewed as maximum likelihood estimates, conditioned on the current estimate $\hat{\mathbf{z}}$ of the label vector $\mathbf{z}$. In the second stage, we assume current estimates $\widehat{\Omega}$ and $\hat{\boldsymbol{\theta}}$ and obtain a new estimate of $\mathbf{z}$ by solving

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmax}} \mathcal{L}(\mathbf{z}, \widehat{\Omega}, \hat{\boldsymbol{\theta}}) \tag{8}$$

We alternate between these two stages until convergence.

There are several identifiability issues that must be addressed. First, permuting the group labels in $\mathbf{z}$ and Ω does not alter the value of the likelihood. We therefore impose an arbitrary order on group labels. Second, the number of possible groups $\bar{\ell}$ can, in principle, be larger than the number of groups present in $\mathbf{z}$. Such a case would correspond to the presence of groups that are statistically possible

but empty in the given data realization. While other treatments are possible, we choose to disregard empty groups and treat $\bar{\ell}$ as equal to the number of distinct labels in an estimate of $\mathbf{z}$. A final form of unidentifiability relates to the scales of $\boldsymbol{\theta}$ and Ω. For a fixed $\boldsymbol{\theta}$ and Ω, we can construct $\boldsymbol{\theta}' \neq \boldsymbol{\theta}$ and $\Omega' \neq \Omega$ such that $\mathcal{L}(\mathbf{z}, \Omega, \boldsymbol{\theta}) = \mathcal{L}(\mathbf{z}, \Omega', \boldsymbol{\theta}')$ (Supplementary Appendix A). To enforce identifiability, we must therefore place a joint normalization condition on either $\boldsymbol{\theta}$ or Ω. We choose to constrain $\boldsymbol{\theta}$ such that

$$\sum_{i=1}^{n} \theta_i \delta(z_i, \ell) = \mathbf{vol}(\ell),\ \ell = 1, \ldots, \bar{\ell} \tag{9}$$

where $\mathrm{vol}(\ell) = \sum_{i=1}^{n} d_i \delta(z_i, \ell)$ and d_i is the (weighted) number of hyperedges in which node i appears. In this expression and below, δ is an indicator function with value 1 if all its inputs are equal and value 0 otherwise.

The usefulness of eq. (9) is that, when $\mathbf{z}$ is known or estimated, the conditional maximum likelihood estimates $\widehat{\boldsymbol{\theta}}$ and $\widehat{\Omega}$ in eq. (7) take simple, closed forms. First, for a fixed label vector $\mathbf{z}$, when using the normalization (9), the maximum likelihood estimate for $\boldsymbol{\theta}$ is (see Supplementary Appendix B)

$$\widehat{\boldsymbol{\theta}} = \mathbf{d} \tag{10}$$

Second, conditioned on $\mathbf{z}$, if Ω takes constant value ω on some set Y of unordered tuples of labels, then the maximum likelihood estimate for ω is (see Supplementary Appendix C)

$$\widehat{\omega} = \frac{\sum_{\mathbf{y} \in Y} \sum_{R \in \mathcal{R}} a_R \delta(\mathbf{z}_R, \mathbf{y})}{\sum_{\mathbf{y} \in Y} \prod_{y \in \mathbf{y}} \mathbf{vol}(y)} \tag{11}$$

In full generality, we can estimate one such ω for every possible label arrangement in the data. Later, we will make natural restrictions on Ω. Although eq. (10) assumes that $\mathbf{z}$ was fixed, it is not necessary to know $\mathbf{z}$ to form the estimate $\widehat{\boldsymbol{\theta}}$. However, forming the estimate $\widehat{\omega}$ via eq. (11) requires that we know or estimate $\mathbf{z}$. It is therefore important to remember that $\widehat{\omega}$ is not a globally optimal estimate, but rather a locally optimal estimate conditioned on the currently estimated group labels.

The formula (11) could also be inserted directly into the full likelihood maximization problem (2), eliminating the parameters corresponding to Ω and producing a lower-dimensional profile likelihood, which could then in principle be optimized directly. This approach has been successful for dyadic blockmodels (*49*), and the development of similar methods for hypergraph blockmodels would be of considerable interest. The advantage of our coordinate ascent framework is that we are able to develop fast heuristics for solving problem (8), by generalizing widely used algorithms for graph clustering hypergraph maximum likelihood Louvain, below. Solving problem (7) in our framework can also be interpreted as evaluating the profile likelihood for a fixed cluster vector $\mathbf{z}$, highlighting the relationship between these approaches.

We now turn to the problem of inferring the label vector $\mathbf{z}$. This problem leads naturally to a class of modularity-type objectives for hypergraph clustering.

Symmetric modularities

Our results from the previous section imply that the estimated degree parameter $\widehat{\boldsymbol{\theta}}$ and piecewise constant affinity function $\widehat{\Omega}$ can be efficiently estimated in closed form, provided an estimate of $\mathbf{z}$. This provides a solution to the first stage (7) of coordinate ascent. We now discuss the second stage (8). From eq. (3), it suffices to optimize Q with respect to $\mathbf{z}$. To do so, it is helpful to impose some additional structure on $\widehat{\Omega}$.

We obtain an important class of objective functions by stipulating that Ω is symmetric with respect to permutations of node labels. In this case, $\Omega(\mathbf{z}_R)$ depends not on the specific labels $\mathbf{z}_R$ in a given node tuple R but only on the number of repetitions of each. Statistically, the corresponding DCHSBM generates hypergraphs in which all groups are statistically identical, conditioned on the degrees of their constituent nodes. Symmetric affinity functions thus give a flexible generalization of the planted partition SBM (*50*, *51*) to the setting of hypergraphs.

Define the function $\phi(\mathbf{z}) = \mathbf{p}$, where p_j is the number of entries of $\mathbf{z}$ in the jth largest group in $\mathbf{z}$, with ties broken arbitrarily. For example, if $\mathbf{z} = (1,1,4,1,2,3,2)$, then $\mathbf{p} = (3,2,1,1)$. We call $\mathbf{p}$ a partition vector. The symmetry assumption implies that Ω is a function of $\mathbf{z}_R$ only through $\mathbf{p} = \phi(\mathbf{z}_R)$. Accordingly, we abuse notation by writing $\Omega(\mathbf{p}) \equiv \Omega(\mathbf{z})$ when $\mathbf{p} = \phi(\mathbf{z})$.

We now define generalized cuts and volumes corresponding to a possible partition vector $\mathbf{p}$ for tuples of k nodes

$$\mathbf{cut}_{\mathbf{p}}(\mathbf{z}) \equiv \sum_{R \in \mathcal{R}^k} a_R \delta(\mathbf{p}, \phi(\mathbf{z}_R)) \tag{12}$$

$$\mathbf{vol}_{\mathbf{p}}(\mathbf{z}) \equiv \sum_{\mathbf{y} \in [\bar{\ell}]^k} \delta(\mathbf{p}, \phi(\mathbf{y})) \prod_{y \in \mathbf{y}} \mathbf{vol}(y) \tag{13}$$

where $\mathcal{R}^k$ is the subset of tuples in $\mathcal{R}$ consisting of k nodes. The function $\mathbf{cut}_{\mathbf{p}}(\mathbf{z})$ counts the number of edges that are split by $\mathbf{z}$ into the specified partition $\mathbf{p}$, while the function $\mathbf{vol}_{\mathbf{p}}(\mathbf{z})$ is a sum-product of volumes over all grouping vectors $\mathbf{y}$ that induce partition $\mathbf{p}$. Let $\mathcal{P}$ be the set of partition vectors on sets up to size $\bar{k}$, the maximum size of hyperedges. We show in Supplementary Appendix D that the symmetric modularity objective can then be written as

$$Q(\mathbf{z}, \Omega, \mathbf{d}) = \sum_{\mathbf{p} \in \mathcal{P}} \left[\mathbf{cut}_{\mathbf{p}}(\mathbf{z}) \log \Omega(\mathbf{p}) - \mathbf{vol}_{\mathbf{p}}(\mathbf{z}) \Omega(\mathbf{p}) \right] \tag{14}$$

For a partition vector $\mathbf{p}$ for tuples of k nodes, direct calculation of $\mathbf{vol}_{\mathbf{p}}(\mathbf{z})$ is a summation of $\bar{\ell}^k$ elements, which can be impractical when either $\bar{\ell}$ or k are large. We show in Supplementary Appendix E, however, that it is possible to efficiently evaluate these sums via a combinatorial identity. We also give a formula for updating volume terms $\mathbf{vol}_{\mathbf{p}}(\mathbf{z})$ when a candidate labeling is modified.

The objective function (14) is related to a recent formulation of the multiway hypergraph cut problem (*52*). They formulate the hypergraph cut objective in terms of splitting functions, which associate penalties when edges are split between two or more clusters. One then aims to minimize the sum of penalties subject to constraints that certain nodes must not lie in the same cluster. Symmetric affinity functions in our framework correspond to signature-based splitting functions in their terminology. Table 1 lists four of many families of affinity functions.

The All-Or-Nothing (AON) affinity function distinguishes only whether a given edge is contained entirely within a single cluster. This affinity function is especially important for scalable computation, and we discuss it further below. The Group Number (GN) affinity depends only on the number of distinct groups represented in an edge, regardless of the number of incident nodes in each one. Special cases of the GN affinity arise frequently in applications. When $f(\|\mathbf{p}\|_0, k) = \exp(\|\mathbf{p}\|_0 - 1)$, the first term of the modularity objective corresponds to a hyperedge cut penalty that is known in the scientific

Table 1. Symmetric affinity functions. Throughout, $k=\|\mathbf{p}\|_0$ is the number of nodes in partition $\mathbf{p}$, ω_{k0} and ω_{k1} are scalars, and f, g, and h are arbitrary scalar functions.

All-or-nothing (AON)	$\Omega(\mathbf{p}) = \begin{cases} \omega_{k1} & \|\mathbf{p}\|_0 = 1 \\ \omega_{k0} & \text{otherwise.} \end{cases}$
Group Number (GN)	$\Omega(\mathbf{p}) = f(\|\mathbf{p}\|_0, k)$
Relative Plurality (RP)	$\Omega(\mathbf{p}) = g(p_1 - p_2, k)$
Pairwise	$\Omega(\mathbf{p}) = h(\sum_{i \neq j} p_i p_j, k)$

computing literature as "connectivity – 1" (*53*), the K–1 metric (*54*), or the boundary cut (*55*). It has also been called fanout in the database literature (*17*). The related Sum of External Degrees penalty (*54*) is also a special case of the GN affinity. The Relative Plurality (RP) affinity considers only the relative difference between the size of the largest group represented in an edge and the next largest group. This rather specialized affinity function is especially appropriate in contexts where groups are expected to be roughly balanced, as we find, for example, in party affiliations in congressional committees. Last, the Pairwise affinity counts the number of pairs of nodes within the edge whose clusters differ. While this affinity function uses similar information to that used in dyadic graph models, there is no immediately apparent equivalence between any dyadic random graph and a DCHSBM with the Pairwise affinity function. There are many more symmetric affinity functions; see table 3 of (*52*) for several other splitting functions that can be used to define affinities.

An important subtlety has been recently raised (*56*) concerning the relationship between blockmodels and modularity derived in (*48*). This consideration also applies to our derivation of eq. (14) above and eq. (15) below. We derived the conditional maximum likelihood estimates (10) and (11) of $\boldsymbol{\theta}$ and Ω under the assumption of a general, unconstrained affinity function Ω. It is not guaranteed that these same estimates maximize the likelihood when additional constraints—such as the symmetry constraint $\Omega(\mathbf{z}) = \Omega(\phi(\mathbf{z}))$—are imposed. In the case of dyadic graphs, eqs. (10) and (11) for estimating $\widehat{\boldsymbol{\theta}}$ and $\widehat{\Omega}$ are only exact under the symmetry assumption on Ω when $\mathbf{vol}(\ell)$ is constant for each $\ell \in [\bar{\ell}]$ (*56*). When the sizes of groups vary, as is typical in most datasets, eqs. (10) and (11) are instead approximations of the exact conditional maximum likelihood estimates. The situation is reminiscent of the tendency of the graph modularity objective to generate clusters of approximately equal sizes (*57*). The objectives and algorithms that we develop below should therefore be understood as approximations to coordinate-ascent maximum likelihood inference, which are exact only in the case that all clusters have equal volumes.

AON modularity

The AON affinity function is of special interest for modeling and computation. This affinity function is a natural choice for systems in which the occurrence of an interaction or relationship depends strongly on group homogeneity.

Inserting the AON affinity function from Table 1 into eq. (14) yields, after some algebra (Supplementary Appendix F), the objective

$$Q(\mathbf{z}, \Omega, \mathbf{d}) = -\sum_{k=1}^{\bar{k}} \beta_k \left[\mathbf{cut}_k(\mathbf{z}) + \gamma_k \sum_{\ell=1}^{\bar{\ell}} \mathbf{vol}(\ell)^k \right] + J(\boldsymbol{\omega}) \qquad (15)$$

where $\beta_k = \log \omega_{k1} - \log \omega_{k0}$, $\gamma_k = \beta_k^{-1}(\omega_{k1} - \omega_{k0})$, and $J(\boldsymbol{\omega})$ collects terms that do not depend on the partition $\mathbf{z}$. We collect $\{\beta_k\}$ and $\{\gamma_k\}$ into vectors $\boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathbb{R}^{\bar{k}}$. We have also defined

$$\mathbf{cut}_k(\mathbf{z}) \equiv m_k - \sum_{R \in \mathcal{R}^k} a_R \delta(\mathbf{z}_R) \qquad (16)$$

In this expression, m_k is the (weighted) number of hyperedges of size k, i.e., $m_k = \Sigma_{R \in \mathcal{R}^k} a_R$. The cut terms $\mathbf{cut}_k(\mathbf{z})$ thus count the number of hyperedges of size k that contain nodes in two or more distinct clusters. This calculation is a direct generalization of a recent derivation for graph modularity (*48*). We recover the standard dyadic modularity objective by restricting to $k = 2$. We call eq. (15) the AON hypergraph modularity.

Recently, the authors of (*40*) proposed a "strict modularity" objective for hypergraphs. This strict modularity is a special case of eq. (15), obtained by choosing ω_{k0} and ω_{k1} such that $\beta_k = 1$ and $\gamma_k = \frac{m_k}{\mathrm{vol}(H)^k}$, where $\mathrm{vol}(H) = \sum_{i=1}^{n} d_i$ is the sum of all node degrees in hypergraph H. However, leaving these parameters free lends important flexibility to our proposed AON objective eq. (15). Tuning $\boldsymbol{\beta}$ allows one to specify which hyperedge sizes are considered to be most relevant for clustering. In email communications, for example, a very large list of recipients may carry minimal information about their social relationships, and it may be desirable to down-weight large hyperedges. Tuning $\boldsymbol{\gamma}$ has the effect of modifying the sizes of clusters favored by the objective, in a direct generalization of the resolution parameter in dyadic modularity (*58*, *59*). It is not necessary to specify the values of these parameters a priori; instead, they can be adaptively estimated via eq. (11).

Hypergraph Maximum Likelihood Louvain

To optimize the modularity objectives (14) and (15), we propose a family of agglomerative clustering algorithms. These algorithms greedily improve the specified objective through local updates to the node label vector $\mathbf{z}$. The structure of these algorithms is based on the widely used and highly performant Louvain heuristic for graphs (*60*). The standard heuristic alternates between two phases. In the first phase, each node begins in its own singleton cluster. Then, each node i is visited and moved to the cluster of the adjacent node j that maximizes the increase in the objective Q. If no such move increases the objective, then i's label is not changed. This process repeats until no such moves exist that increase the objective. In the second phase, a "supernode" is formed for each label. The supernode represents the set of all nodes sharing that label. Then, the first phase is repeated, generating an updated labeling of supernodes, which are then aggregated in the second phase. The process repeats until no more improvement is possible. Because every step in the first phase improves the objective, the algorithm terminates with a locally optimal cluster vector $\mathbf{z}$.

This heuristic generalizes naturally to the setting of hypergraphs. However, the incorporation of heterogeneous hyperedge sizes and general affinity functions considerably complicates implementation. Here, we provide a highly general hypergraph maximum likelihood Louvain (HMLL) algorithm for optimizing the symmetric modularity objective (14). For the important case of the AON affinity, the simplified objective (15) admits a much simpler and faster specialized Louvain algorithm, which we describe in Supplementary Appendix G. As we show in subsequent experiments, this specialized algorithm is highly scalable and effective in recovering ground truth clusters in datasets with polyadic structure plausibly modeled by the AON affinity.

Symmetric HMLL

The first phase of our symmetric HMLL algorithm mirrors standard graph Louvain: Nodes begin in singleton clusters and, in turn, greedily move to adjacent clusters until no more improvement is possible. Phase 2 of graph Louvain reduces edges between clusters into weighted edges between supernodes in a structure-preserving way. However, in the hypergraph case, naively collapsing clusters and hyperedges would discard important information about hyperedge sizes and the way each hyperedge is partitioned across clusters. Therefore, in subsequent stages of our algorithm, we greedily improve the objective by moving entire sets of nodes in the original hypergraph, rather than greedily moving individual nodes. In this way, a set of nodes that was assigned to the same cluster in a previous iteration is essentially treated as a supernode and moved as a unit, without collapsing the hypergraph and losing needed information about hyperedge structure.

Our overall procedure is formalized in algorithms S1 and S2. Algorithm S1 visits in turn each set of nodes S_c that represents a cluster c from a previous iteration. The algorithm evaluates the change ΔQ in the objective function Q associated with moving all of S_c to an adjacent cluster and then carries out the move that gives the largest positive change to the objective. This is repeated until moving a set S_c can no longer improve the objective. The entire symmetric HMLL method is summarized by running algorithm S2, which starts by placing every node in a singleton cluster before calling algorithm S1 for the first time.

Algorithm S1 relies on a function ΔQ that computes the change in the objective Q associated with moving all nodes from S_c to an adjacent cluster. Changes to the second (volume) term in the objective can be computed efficiently using combinatorial identities (Supplementary Appendix E, Proposition 1). Changes to the first (cut) term require summing across all hyperedges incident to a node or set of nodes. At each hyperedge, we must evaluate the affinity $\Omega(\mathbf{p})$ on the current partition $\mathbf{p}$, as well as the affinity $\Omega(\mathbf{p}')$ associated with the candidate updated partition $\mathbf{p}'$. This situation contrasts with the case of the graph Louvain algorithm, in which it is sufficient to check whether a given edge joins nodes in the same or different clusters. The fact that we need to store and update the partition vector $\mathbf{p}$ for each hyperedge is what prevents us from collapsing a cluster of nodes into a monolithic supernode and recursively applying algorithm S2 on a reduced data structure, as customary in graph Louvain.

Thus, while clusters of nodes move as a unit in algorithm S1 as well, it is necessary in this case to operate on the full adjacency data $\mathcal{A}$ at all stages of the algorithm. This can make algorithm S2 slow on hypergraphs of even modest size. Developing efficient algorithms for optimizing the general symmetric modularity objective or various special cases is an important avenue of future work.

All-or-nothing hypergraph maximum likelihood Louvain

When Ω is the AON affinity function, considerable simplification is possible. For each edge, we need not compute the full partition vector $\mathbf{p}$ but only check whether $\|\mathbf{p}\|_0 = 1$. Rather than a general affinity function Ω, we instead supply the parameter vectors $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ appearing in eq. (15). This allows us to compute on considerably simplified data structures. In particular, we are able to follow the classical Louvain strategy of collapsing clusters into single, consolidated supernodes and restrict attention to hyperedges that span multiple supernodes. Because we do not need to track the precise way in which the hyperedges span multiple supernodes, we can forget much of the original adjacency data $\mathcal{A}$ and instead simply store the edge sizes of the hypergraph. These simplifications enable both substantial memory savings and very rapid evaluation of the objective update function ΔQ. We provide pseudocode for exploiting these simplifications in Supplementary Appendix G.

Number of clusters

Like most Louvain-style modularity methods, the user cannot directly control the number of clusters returned by HMLL. One approach to influence the number of clusters is to manually set values for the affinity function Ω or the parameters $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ (if using the AON affinity). Rather than inferring these parameters from data, one can set them a priori and perform a single round of optimization over $\mathbf{z}$. This approach generalizes the common treatment of the resolution parameter in dyadic modularity maximization as a tuning hyperparameter rather than a number to be estimated from data (*58*). Considerable experimentation may be required to obtain the desired number of clusters, and retrieving an exact number may not be possible.

Another approach to influencing the number of clusters is to impose a Bayesian prior on the community labels. In the simplest version of a Bayesian approach, one assumes that each node is independently assigned one of $\bar{\ell}$ labels with equal probability, before sampling edges. The probability of realizing a given label vector $\mathbf{z}$ is then $\bar{\ell}^{-n}$, which generates a term of the form $-n\log\bar{\ell}$ in the log-likelihood $\mathcal{L}$. This term may then be incorporated into Louvain implementations, with the result that greedy moves that reduce the total number of clusters $\bar{\ell}$ are strongly incentivized. The resulting regularized algorithm may then label vector $\mathbf{z}$ with slightly smaller numbers of distinct clusters. This can be useful when it is known a priori that the true number of clusters in the data is small. Our implementation of AON HMLL incorporates this optional regularization term. We use this term only in the synthetic detectability experiments presented below.

RESULTS

Runtime

Dyadic Louvain algorithms are known for being highly efficient in large graphs. Here, we show that AON HMLL can achieve similar performance on synthetic data to graph MLL (GMLL), a variant of the standard dyadic Louvain algorithm in which we return the combination of resolution parameter and partition that yield the highest dyadic likelihood. For a fixed number of nodes n, we consider a DCHSBM-like hypergraph model with $\bar{\ell} = n/200$ clusters and $m = 10n$ hyperedges with size k uniformly distributed between 2 and 4. Each k-edge is, with probability p_k, placed uniformly at random on any k nodes within the same cluster. Otherwise, with probability $1 - p_k$, the edge is instead placed uniformly at random on any set of k nodes. We use this model rather than a direct DCHSBM to avoid the computational burden of sampling edges at each k-tuple of nodes, which is prohibitive when n is large. For the purpose of performance testing, we compute estimates of the parameter vectors $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ (in the case of AON HMLL) and the resolution parameter γ (in the case of GMLL) using ground truth cluster labels. We emphasize that this is typically not possible in practical applications, because the ground truth labels are not known. We make this choice to focus on a direct comparison of runtimes of each algorithm in a situation in which both can succeed. In later sections, we study the ability of HMLL and GMLL to recover known groups in synthetic and empirical data when affinities and resolution parameters are not known.

Algorithm 1: SymmetricHMLLstep($H, \Omega, \mathbf{z}$)

Data: Hypergraph H, affinity function Ω, current label vector $\mathbf{z}$
Result: Updated label vector $\mathbf{z}'$

$\mathcal{C} \leftarrow$ unique($\mathbf{z}$) // unique cluster labels in the initial clustering $\mathbf{z}$
$\mathbf{z}' \leftarrow \mathbf{z}$ // By greedily moving clusters in $\mathbf{z}$, form new clustering $\mathbf{z}'$
improving $\leftarrow$ true
while *improving* **do**
  improving $\leftarrow$ false
  for c *in* $\mathcal{C}$ **do**
    $S_c \leftarrow \{i : z_i = c\}$ // nodes with label c in original clustering $\mathbf{z}$
    $\mathcal{C}' \leftarrow$ unique($\mathbf{z}'$) // cluster labels in the current clustering $\mathbf{z}'$
    // Set of clusters $\mathcal{A}_c$ in $\mathbf{z}'$ that are adjacent to S_c
    $\mathcal{A}_c \leftarrow \bigcup_{e \in E} \{\tilde{c} \in \mathcal{C}' : \exists v \in V$ with $z'_v = \tilde{c}$ and $i \in S_c$ with $v, i \in e\}$
    // maximum Δ and maximizer c' of the change in Q from moving set S_c to an adjacent cluster $\tilde{c}$ in $\mathcal{A}_c$
    $(\Delta, c') \leftarrow \mathrm{argmax}_{\tilde{c} \in \mathcal{A}_c} \Delta Q(H, \Omega, \mathbf{z}', S_c, \tilde{c})$
    // update $\mathbf{z}'$ if improvement found
    if $\Delta > 0$ **then**
      for $i \in S_c$ **do**
        $z'_i \leftarrow c'$
      end
      improving $\leftarrow$ true
    end
  end
end
return $\mathbf{z}'$

Algorithm 2: SymmetricHMLL(H, Ω)

Data: Hypergraph H, affinity function Ω
Result: Label vector $\mathbf{z}$

$\mathbf{z}' \leftarrow \mathbf{z} \leftarrow [n]$ // assign each cluster to singleton
do
  $\mathbf{z} \leftarrow \mathbf{z}'$
  $\mathbf{z}' \leftarrow$ SymmetricHMLLstep($H, \Omega, \mathbf{z}$)
while $\mathbf{z} \neq \mathbf{z}'$
return $\mathbf{z}$

Figure 1 shows runtime, adjusted Rand index (ARI), and number of clusters returned on synthetic hypergraphs when $p_2 = 0.6$, $p_3 = 1/n^3$, and $p_4 = 1/n^4$. These parameter are chosen so that hyperedges of size three and four are rarely (if ever) contained completely inside clusters. Thus, hyperedges of different sizes provide different signal regarding ground truth clusters. For this experiment, we implemented GMLL by computing a normalized clique projection, in which nodes are joined by weighted dyadic edges with weights

$$w_{ij} = \sum_{e: i,j \in e} \frac{1}{|e| - 1} \tag{17}$$

We also performed experiments on an unnormalized clique projection with $w_{ij} = |e : i, j \in e|$ but do not show these results because, in this experiment, the associated MLL algorithm consistently fails to recover labels correlated with the planted clusters.

On smaller instances, HMLL outperforms GMLL in recovering planted clusters, as measured by the ARI. For larger instances, the recovery results are comparable. Although GMLL and HMLL obtain similar accuracy in this experiment, they do so in different ways, with HMLL tending to generate more, smaller clusters than GMLL. The runtimes are nearly indistinguishable, indicating that dyadic clique projections are necessary neither for accuracy nor for performance. We observed other choices of the parameters p_2, p_3, and p_4 in which HMLL substantially outperformed GMLL in cluster recovery and vice versa; however, in each case, the algorithms' respective runtimes tended to differ by only a small constant factor.

In this synthetic experiment, a combination of the two algorithms leads to the strongest recovery results. In addition to independently running each algorithm, we also ran a two-stage algorithm in which GMLL is used to generate an intermediate partition and then HMLL is used to refine it. We emphasize again that these results are obtained on synthetic hypergraphs with preoptimized affinity parameters, and so the effectiveness of the refinement strategy may not generalize to real datasets. In the experiments on empirical data shown below we do not show results from the refinement procedure because the output partition was, in each case, essentially indistinguishable from the output of the dyadic partition. This may reflect the fact that we did not allow the algorithms to learn a priori the affinity parameters associated with the true data labels. Further investigation into the performance of hybrid strategies would be of considerable practical importance.

Dyadic projections and the detectability threshold

Informally, an algorithm is able to detect communities in a random graph model with fixed labels $\mathbf{z}$ when the output labeling $\hat{\mathbf{z}}$ of that algorithm is, with probability bounded above zero, correlated with $\mathbf{z}$. Using arguments from statistical physics, the authors of (*45*) conjectured the existence of a regime in the graph SBM in which no algorithm can successfully detect communities. This conjecture has since been refined and proven in various special cases; see (*61*) for a survey. In the dyadic SBM with two equal-sized planted communities, a necessary condition for detectability in the large-graph limit is

$$\frac{(c_i - c_o)^2}{2(c_i + c_o)} \geq 1 \tag{18}$$

where c_i is the mean number of within-cluster edges attached to a node and c_o is the mean number of between-cluster edges attached to a node. If this condition is not satisfied, then no algorithm can reliably detect communities in the associated graph SBM, although the communities are statistically distinct. This bound limits direct inferential methods, such as Bayesian or maximum likelihood techniques, and methods based on maximization of modularity or other graph objectives (*62*). Several recent papers have considered the detectability problem in the case of uniform hypergraphs (*33*, *35*, *63*). In our model, the presence of edges of multiple sizes complicates analysis. Here, we limit ourselves to an experimental demonstration that the regimes of detectability for the graph SBM and our DCHSBM can differ significantly.

Figure 2 shows two experiments on a simple DCHSBM with two equal-sized communities of 250 nodes each. The affinity $\boldsymbol{\Omega}$ is tuned so that

1) Each node is incident to, on average, five 2-edges and five 3-edges.
2) A fraction p_2 of 2-edges join nodes in the same cluster, while a fraction $1 - p_2$ of 2-edges join nodes in different clusters.
3) A fraction p_3 of 3-edges join nodes in the same cluster, while a fraction $1 - p_3$ of 3-edges join nodes in different clusters.

In this experiment only, both GMLL and AON HMLL discussed below use the Bayesian regularization term $-n\log\bar{\ell}$ in the likelihood objective to encourage each algorithm to form a relatively small number of clusters. In the lefthand panel, we show the ARI of the returned partition $\hat{\mathbf{z}}$ against the true partition $\mathbf{z}$ when using the unnormalized variant of GMLL (results for the normalized variant are similar). This choice reflects the fact that the true number of

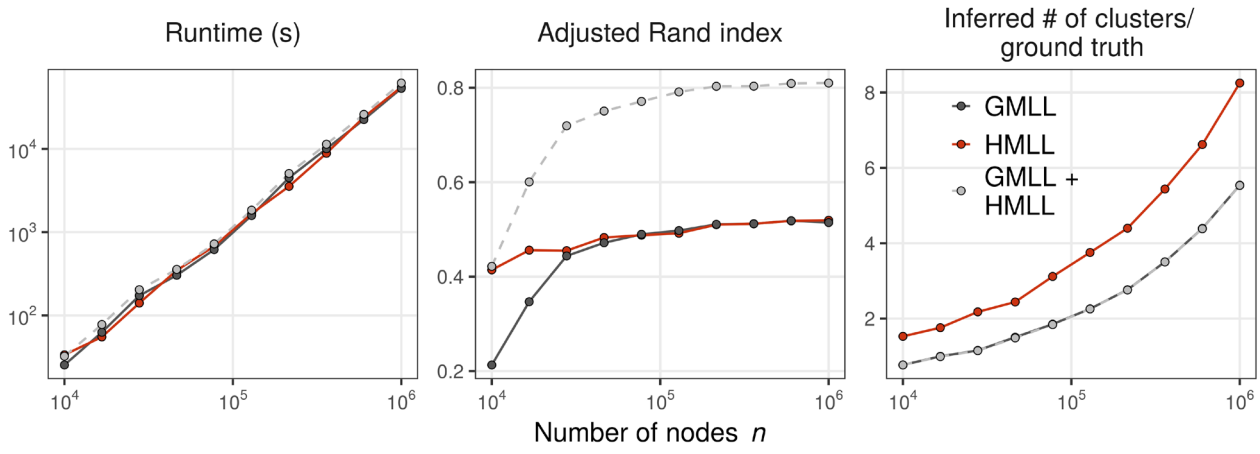

Fig. 1. Runtime, ARI, and number of clusters returned by GMLL and HMLL in a synthetic testbed with optimal affinity parameters. The within-cluster edge placement probabilities are $p_2 = 3/5$, $p_3 = 1/n^3$, and $p_4 = 1/n^4$. We also show in light gray the results obtained by using GMLL as a preprocessing step, whose output partition is then refined by HMLL (light gray).

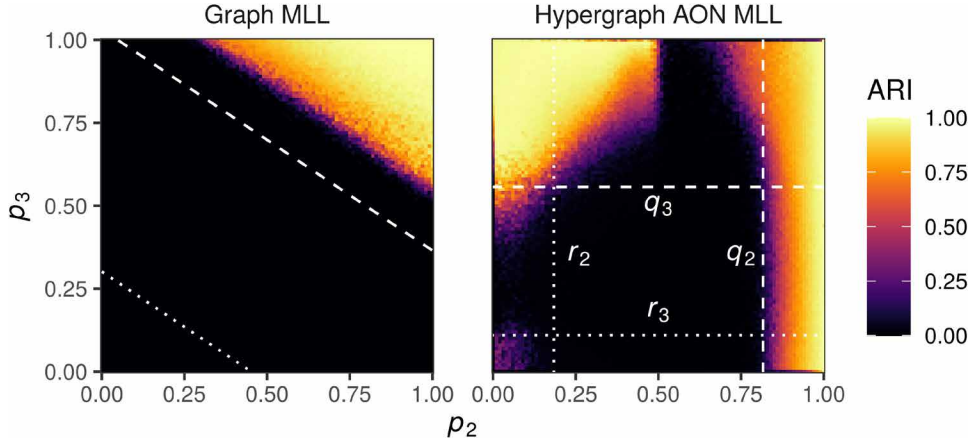

Fig. 2. Detectability experiments in synthetic hypergraphs. For $i = 2,3$, p_i is the proportion of within-cluster edges of size i. Each pixel gives the mean ARI over 20 independently generated DCHSBMs of size $n = 500$ where each node is incident to, on average, five 2-edges and five 3-edges. (**Left**) The recovered partition $\hat{\mathbf{z}}$ is obtained from GMLL. (**Right**) The recovered partition is obtained from AON HMLL (algorithm S1). The dashed and dotted lines give various detectability thresholds as described in the main text. In each panel, the returned partition $\hat{\mathbf{z}}$ is the highest-likelihood partition from 20 alternations between updating $\hat{\mathbf{z}}$ and inference of the affinity parameters. In this experiment only, we incorporate a regularization term $-n\log\bar{\ell}$ in the modularity objective to promote label vectors $\mathbf{z}$ with fewer clusters.

clusters is known and is equal to 2. The dashed and dotted white lines give the boundaries at which eq. (18) holds with equality. The dashed white line gives the detectability threshold for the assortative regime in which nodes are more likely to link with others in the same cluster. Louvain, as an agglomerative algorithm, is well suited for detecting assortative clusterings and is able to detect communities in much, but not all, of this regime. The gap between the theoretical threshold and the performance of Louvain reflects the fact that Louvain, as a stagewise greedy algorithm, has no optimality guarantees. There is also a disassortative detectable region below the dotted white line. The agglomerative structure of graph-based Louvain causes the algorithm to entirely fail here.

In the righthand panel, we show the same experiment using AON HMLL. The dashed lines q_2 and q_3 give assortative detectability thresholds for hypothetical algorithms that entirely ignore 3-edges and 2-edges, respectively, while the dotted lines r_2 and r_3 give the corresponding disassortative thresholds. HMLL is able to detect the planted partition for a range of parameter values in which GMLL is not. These include the case in which edges of certain sizes are largely between-cluster, as shown in the top left (small p_2) and bottom right (small p_3). There is a regime (mid and bottom right) in which the algorithm appears to be constrained by the boundary q_2, suggesting that HMLL is effectively ignoring 3-edges in this regime. As p_3 increases, however, 3-edges become more informative and the partition can be detected for some values $p_2 < q_2$ (top right). There is also a broad regime (top left) in which the hypergraph algorithm is able effectively to use both 2- and 3-edges to detect clusters, even when 2-edges are largely between-cluster. We also observe some very limited ability of HMLL to detect clusters in the regime in which both 2-edges and 3-edges are between-cluster (bottom left). Because HMLL is again an agglomerative algorithm, its performance for fully disassortative partitions such as these is unreliable at best.

Intriguingly, there are also combinations of p_2 and p_3 in which GMLL is able to detect the planted partition while HMLL is not. This may indicate that the pooling of edges of different sizes implied by the dyadic projection can be useful in some regimes. We note again that neither GMLL nor HMLL are optimal inference algorithms. An optimal hypergraph algorithm might significantly extend the detectable regime in the right panel of Fig. 2. We pose the development of these algorithms, as well as their analysis, as highly promising avenues for future research.

Experiments with empirical data

Next, we analyze several hypergraphs derived from empirical data. The first two are hypergraphs of human close-proximity contact interactions (*6*), obtained from wearable sensor data at a primary school (*64*) and a high school (*65*). Nodes are students or teachers, and a hyperedge connects groups of people that were all jointly in proximity to one another. Node labels identify the classrooms to which each student belongs, and the primary school data also includes a teacher associated to each class. Next, we created two hypergraphs from U.S. congressional bill cosponsorship data (*66*, *67*), where nodes correspond to congresspersons and hyperedges correspond to the sponsor and all cosponsors of a bill in either the House of Representatives or the Senate. We constructed another pair of datasets from the U.S. Congress in the form of committee memberships (*68*). Each edge is a committee in a meeting of Congress, and each node again corresponds to a member of the House or a senator. A node is contained in an edge if the corresponding legislator was a member of the committee during the specified meeting of Congress. The 103rd through 115th Congresses are represented, spanning the years 1993–2017. There are again separate datasets for House and Senate members. In all congressional datasets, the node labels give the political parties of the members. We also used a hypergraph of Walmart purchases (*69*), where each node is a product and a hyperedge connects a set of products that were copurchased by a customer in a single shopping trip. Each node has an associated product category label. Last, we constructed a hypergraph where nodes correspond to hotels listed at trivago.com, and each hyperedge corresponds to a set of hotels whose website was clicked on by a user of Trivago within a browsing session. This hypergraph was derived from data released for the 2019 ACM RecSys Challenge contest (*70*). For each hotel, the node label gives the country in which it is located. The datasets vary in size in terms of the number of nodes, hyperedges, hyperedge sizes, and node labels (Table 2).

Model comparison and higher-order structure

It is often stated that higher-order features are important for understanding the structure and function of complex networks. It is less often clarified what kinds of higher-order features are relevant for which networks. Generative modeling provides one way to compare different kinds of higher-order structure. In the DCHSBM, this structure is specified by the affinity function $\boldsymbol{\Omega}$. Comparison of the likelihood functions obtained by each affinity can indicate which one is most plausible as a higher-order generative mechanism of the underlying data. We performed such a comparison using the symmetric affinity functions from Table 1 and the labels for nodes described above. In this setup, we can compute an approximate ML estimate for $\boldsymbol{\Omega}$, given its functional form. To make concrete comparisons, it is necessary to specify the functional forms of the GN, RP, and Pairwise affinities. We use the following parameterizations

$$\Omega(\mathbf{p}) = \omega_{\|p\|_0,k} \qquad \text{(Group Number)}$$

$$\Omega(\mathbf{p}) = \begin{cases} \omega_{k1} & p_1 - p_2 < \frac{k}{4} \\ \omega_{k0} & \text{otherwise} \end{cases} \qquad \text{(Relative Plurality)}$$

$$\Omega(\mathbf{p}) = \begin{cases} \omega_{k1} & \sum_{i \neq j} p_i p_j < \frac{k(k-1)}{4} \\ \omega_{k0} & \text{otherwise} \end{cases} \qquad \text{(Pairwise)}$$

The GN affinity function assigns a separate parameter to each combination of edge size and number of groups. The RP affinity function assigns one parameter for the case that the difference between the largest and second largest groups within an edge exceeds $k/4$, where k is the size of the edge. The Pairwise affinity function assigns one parameter to the case that the total number of dyadic pairs in differing groups exceeds half the possible number of these pairs. RP, which favors edges that the two most common labels are roughly balanced in representation, is qualitatively distinct from AON, GN, and Pairwise, all of which favor edges with homogeneous cluster labels.

Because these affinity functions have different numbers of parameters, we compare them via the Bayesian Information Criterion (BIC) (*71*), which penalizes affinity functions with more parameters than are supported by the data. In computing the BIC, we exclude the n parameters $\boldsymbol{\theta}$, as these are the same in each model and therefore contribute an unimportant additive constant. The AON, RP, and Pairwise affinities each have $2\bar{k}$ parameters. In the case of GN, we compute the number of possible parameters for each edge size k by computing the number of possible groups using the number of

Table 2. Summary study datasets. Shown are the number of nodes n, number of hyperedges m, mean degree $\langle d \rangle$, SD of degree $s(d)$, mean edge size $\langle k \rangle$, SD of edge size $s(k)$, and number of data labels $\bar{\ell}$.

	n	m	$\langle d \rangle$	$s(d)$	$\langle k \rangle$	$s(k)$	$\bar{\ell}$
contact-primary-school	242	12,704	127.0	55.3	2.4	0.6	11
contact-high-school	327	7,818	55.6	27.1	2.3	0.5	9
house-bills	1,494	43,047	274.0	282.7	9.5	7.2	2
senate-bills	293	20,006	493.4	406.3	7.3	5.5	2
house-committees	1,290	340	9.2	7.1	35.2	21.3	2
senate-committees	282	315	19.0	14.7	17.5	6.6	2
walmart-purchases	88,860	65,979	5.1	26.7	6.7	5.3	11
trivago-clicks	171,495	220,758	4.0	7.0	4.2	2.0	160

Table 3. BIC of the DCHSBM using the AON, GN, RP, and Pairwise affinity functions on our full study datasets. Definitions of each affinity function are supplied in Table 1. Lower BIC indicates a more plausible model. The affinity function achieving the lowest BIC in each dataset is shown in bold.

	AON	GN	RP	Pairwise	
contact-high-school	2.2003	**2.1946**	2.4330	2.2003	$\times 10^5$
contact-primary-school	4.1954	**4.1646**	4.3990	4.1954	$\times 10^5$
house-committees	2.7128	2.7128	**2.7119**	2.7127	$\times 10^5$
senate-committees	9.7934	9.7934	**9.7736**	9.7933	$\times 10^4$
house-bills	9.9719	9.9720	10.003	**9.9670**	$\times 10^6$
senate-bills	**3.1925**	3.1926	3.2030	3.1925	$\times 10^6$
walmart-purchases	1.0763	**1.0753**	1.0806	1.0758	$\times 10^6$
trivago-clicks	**1.6854**	1.6866	2.0257	1.6960	$\times 10^8$

distinct labels in the given partition. For example, if the given partition contained only three distinct groups, then we do not posit parameters corresponding to edges containing more than three groups. It would also be reasonable to remove this restriction, in which case there would be k parameters for edges of size k regardless of $\mathbf{z}$.

Table 3 shows the BIC for the DCHSBM using each of these affinity functions. No single affinity function is preferred across all of the study datasets, suggesting the presence of different kinds of polyadic structure. In the two congressional committee datasets, RP achieves the optimal BIC, while in each of the other datasets, one of the three affinities that promotes edge homogeneity is instead preferred. There are also important differences between these three affinities. In house-bills, the Pairwise affinity function achieves the lowest BIC overall, while in walmart-purchases the Pairwise affinity is preferred over all but the GN affinity. This suggests that a model involving only pairwise comparison of node labels can provide relatively strong generative explanations of the data in these cases. This, in turn, suggests that dyadic algorithms may perform at least as well on these datasets as their polyadic counterparts. As we will see below, in both of these datasets, dyadic algorithms can return clusterings more correlated with ground truth than those returned by AON HMLL.

Recovering classes in contact hypergraphs

To test the AON HMLL algorithm itself, we first study its behavior in the contact-primary-school and contact-high-school networks. The comparison of BIC scores from Table 3 suggests that GN may be the most explanatory model of the data, but we instead use AON to take advantage of its considerable computational benefits. We performed 20 alternations between AON HMLL and estimation of the AON parameters and returned the partition with the highest DCHSBM likelihood. We compare the results to two dyadic methods. Each step of the Graph Louvain algorithm alternates between using the standard Louvain algorithm (*60*) to infer clusters and estimating the resolution parameter γ using the approximate maximum likelihood framework of (*48*). Graph Louvain returns the partition that maximizes the classical dyadic modularity objective. We also compare to GMLL, which carries out the same alternation but instead returns the partition that maximizes the approximate log-likelihood of the corresponding planted partition SBM.

Figure 3 compares the performance of each of these algorithms. In the case of contact-primary-school, we consider the ground truth partition to be the one that assigns exactly one teacher to each class. Graph Louvain is able to find partitions of students with clear correlations with the given class labels but conflates two primary school classes and splits several high school classes (left column, top two rows). GMLL is able to perfectly recover the primary school student class labels and misclassifies three high school students. Our proposed AON HMLL is able to correctly recover the given partitions in both datasets.

We can obtain some qualitative insight into the behavior of HMLL by studying the structure of the inferred affinity function Ω. The most intuitive way to do so is through the derived parameters β_k and γ_k from eq. (15). The bottom row of Fig. 3 shows these parameters and the distribution of edge sizes. The dependence of β_k on edge size k provides one explanation of why GMLL succeeds in contact-primary-school but makes several errors in contact-high-school. Under the standard dyadic projection, a k-hyperedge generates $\binom{k}{2}$ 2-edges and therefore appears in the dyadic modularity objective $\binom{k}{2}$ distinct times. In the case of contact-primary-school, the estimated importance parameter β_k is indeed relatively close to $\binom{k}{2}$ (Fig. 3, bottom center). At the optimal partition, the relative weights of edges are therefore distorted relatively little by the clique projection. On the other hand, the estimates for β_k in contact-high-school deviate considerably from $\binom{k}{2}$, especially for $k = 2,3$. Here, small edges feature much more prominently in the polyadic modularity objective than they do in the projected dyadic objective, implying that the latter is a poorer approximation to the former near the optimal partition. This difference may explain the small errors in GMLL in contact-high-school. The bottom-right panel of Fig. 3 compares the inferred value of the size-specific resolution parameter γ_k to $\gamma_{k0} = m_k/\mathbf{vol}(H)^k$, the implicit value used in (*40*). The inferred resolution parameters are consistently larger γ_{k0} and increase with k, highlighting the value of adaptively estimating these parameters in our approach.

Cluster recovery with large hyperedges

In Fig. 4, we study the ability of AON HMLL to recover ground truth communities in several more of our study datasets. Unlike the two contact networks, each of these datasets contains edges of size up to 25 nodes. We have excluded house-committees and senate-committees on the grounds that these datasets are disassortative, indicating that AON is clearly inappropriate. We compare AON HMLL to two variants of GMLL. In the unnormalized variant, we obtain a dyadic graph by replacing each k-edge with a k-clique, thus generating a total of $\binom{k}{2}$ dyadic edges. In the normalized variant, we weight each edge in the k-clique by a factor of $\frac{1}{k-1}$. The normalized

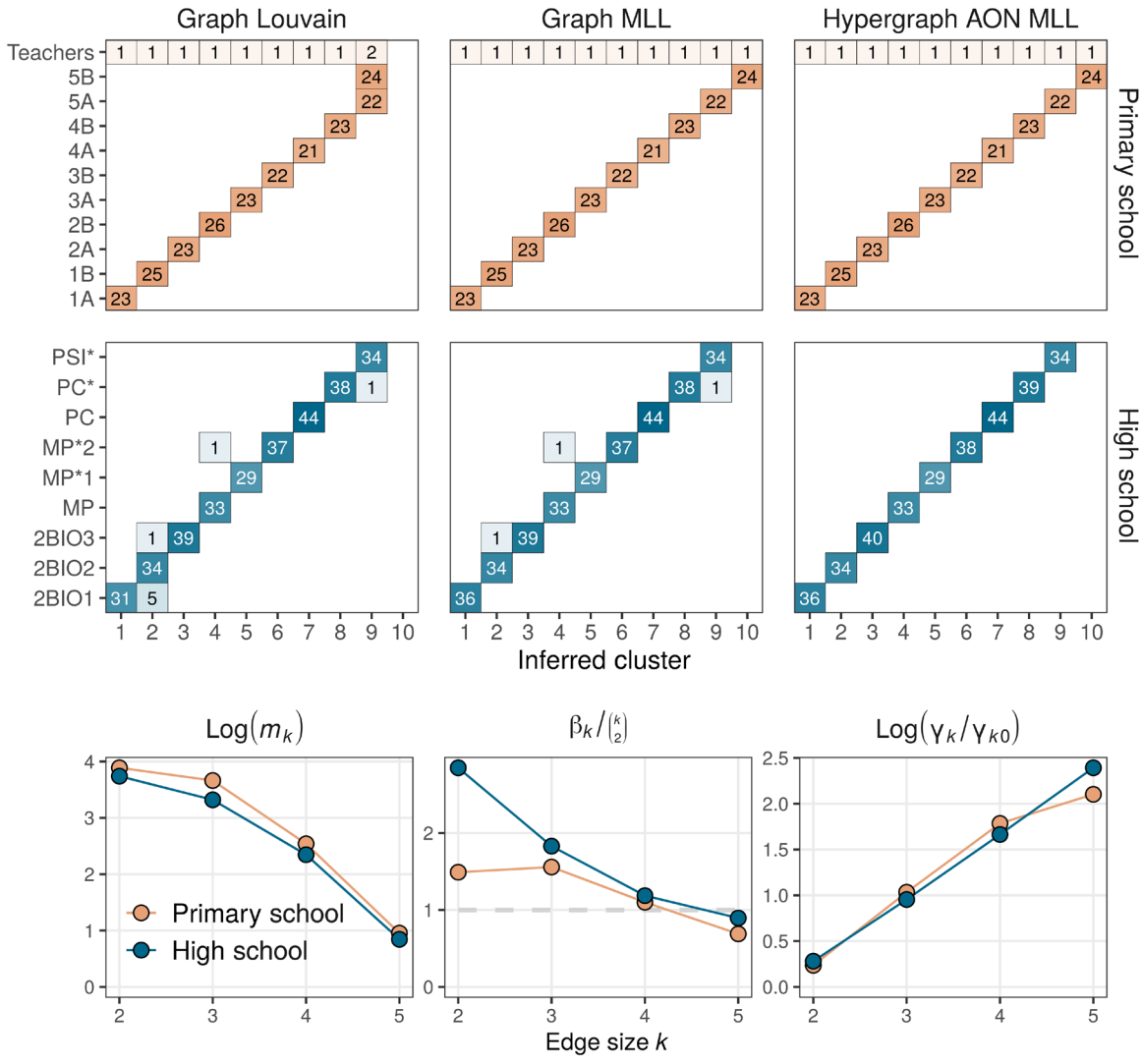

Fig. 3. Comparison of clustering algorithms in contact-primary-school and contact-high-school. For each dataset, we show a partition obtained from the classical graph Louvain modularity maximization heuristic, a partition obtained from GMLL, and partition obtained by AON HMLL. The partition shown is the one that attains the corresponding objective function after 20 rounds of iterative likelihood maximization. Each box records the number of agents with the specified combination of inferred cluster and ground truth label. The bottom row visualizes the number m_k of edges of size k, the inferred size weights β_k, and inferred resolution parameters γ_k as defined in (15). On the far right, $\gamma_{k0} = m_k/\mathbf{vol}(H)^k$.

dyadic degree of each node is then equal to its degree in the original hypergraph. In either case, we then alternate between the dyadic Louvain algorithm for estimating clusters and conditional maximum likelihood inference of the resolution parameter γ. In each trial, we perform 20 iterations of AON HMLL and the two GMLL variants, returning from these the combination of group labels and parameters that achieves the highest likelihood. We then compare the clustering to the ground truth labels via the ARI. We vary the maximum edge size $\bar{k}$ to show how each algorithm responds to the incorporation of progressively larger edges. Because extreme sparsity poses issues for community detection algorithms in general (*61*), we show experiments for progressively denser cores of trivago-clicks and walmart-purchases. The c-core of of a hypergraph H is defined as the largest subhypergraph H_c such that all nodes in H_c have degree at least c.

The results highlight the strong dependence of the performance of AON HMLL on the relative plausibility of the AON affinity function as a generative mechanism for the data (cf. Table 3). In trivago-clicks, the AON affinity function achieved the lowest BIC of all four candidates. Because AON is a more plausible generating mechanism by this metric, it is not unusual that AON HMLL is able to find partitions considerably more correlated with the supplied data labels than those returned by the dyadic variants. In walmart-purchases, on the other hand, the Pairwise affinity is preferred to AON. In this case, AON HMLL performs much worse and, in the 2-core, even returns clusters that are anticorrelated with the supplied labels. As weakly connected nodes are removed and the resulting data become denser, HMLL begins to return correlated clusters. However, the normalized GMLL variant is at least as effective in recovering the data labels. In the two congressional bills datasets, the Pairwise affinity achieves

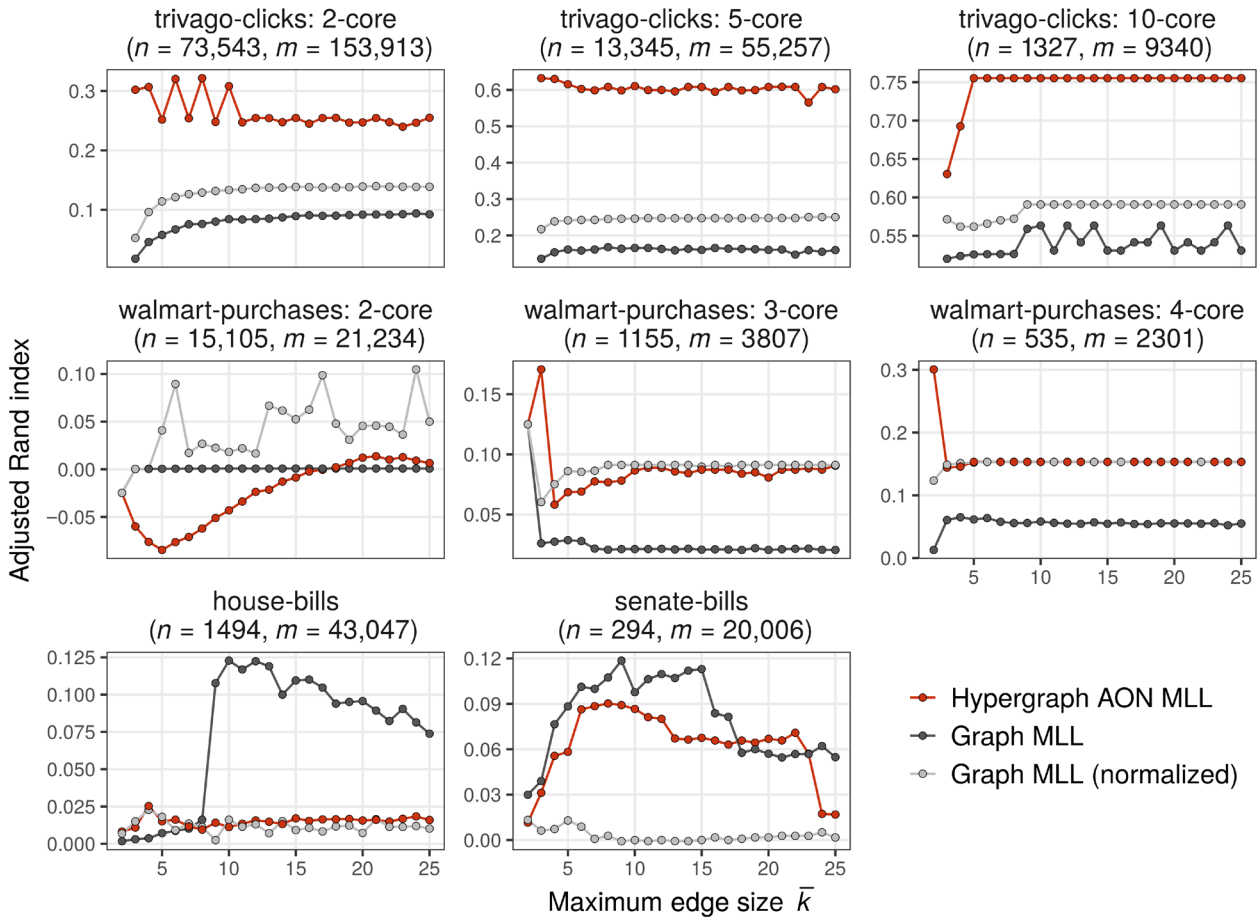

Fig. 4. Comparison of hypergraph AON MLL (algorithm S1) against dyadic likelihood Louvain in data with known clusters. Points give the ARI of the highest-likelihood partition obtained after 20 alternations between partitioning and parameter estimation. The maximum edge size $\bar{k}$ varies along the horizontal axis. In the panel titles, n is the number of nodes and m the number of edges when $\bar{k} = 25$. Note that the vertical axis limits vary between panels.

a lower BIC than AON in the House and a comparable one in the Senate. Echoing this finding, a dyadic method outperforms AON HMLL in each of these cases. Unnormalized GMLL performs best in house-bills and senate-bills, while normalized GMLL is preferable in walmart-purchases. In addition, HMLL is the worst algorithm only in the case of the 2-core of walmart-purchases for small $\bar{k}$. HMLL may therefore be the algorithm of choice in cases when it is not known whether normalized or unnormalized dyadic representations are more appropriate for the data.

When interpreting these recovery results, it is important to contextualize them against the limitations of community detection methods in general and of modularity maximization in particular. There is no "best algorithm" for community detection that does not make implicit assumptions about the structure of the data, and mismatch of algorithms to datasets can generate misleading results (*72*). Even when the data-generating process indeed matches algorithmic assumptions—such as a synthetic dataset generating from an SBM—optimal algorithms may fail to detect planted communities because of sparsity (*45*, *61*). Greedy modularity maximization, including the Louvain variants considered here, only finds one of possibly many local optima (*73*), some of which may be largely uncorrelated with each other. These considerations imply that (i) we cannot rule out the existence of other local optima that might achieve higher scores in any of the three algorithms and (ii) the fact that an algorithm fails to recover a clustering close to the ground truth does not imply that it is "failing" in its stated objective, namely, local likelihood maximization. Overall, our results suggest that, when the assumptions of the DCHSBM with AON affinity are appropriate to the data, AON HMLL can outperform dyadic approaches in recovering ground truth communities. In practice, because we often do not have access to ground truth labels, the question of whether the assumptions are appropriate to the data should be informed by domain expertise.

DISCUSSION

We have proposed a generative approach for clustering polyadic data, grounded in a DCHSBM. From this model, we have derived a symmetric, modularity-like objective, which includes the AON modularity objective as an important special case. This derivation connects hypergraph modularity objectives to concrete modeling assumptions, which can be tuned in response to domain expertise. We have also formulated Louvain-like algorithms for optimizing these objectives, which are highly scalable in the case of the AON affinity function. Embedding this heuristic within an alternating approximate maximum likelihood scheme allows adaptive estimation of both node clusters and affinity parameters. We have shown experimentally that

hypergraph algorithms have markedly different detectability regimes from dyadic algorithms. We have also conducted experiments on empirical data, finding that hypergraph methods are preferred to dyadic ones in datasets where their modeling assumptions are well founded.

Our work points toward many directions of further research. One of these directions is algorithmic. Our greedy coordinate ascent framework for inference in the DCHSBM has several important limitations. First, because we rely on an NP-hard optimization step, global maximization of the likelihood is never assured. Second, even exact maximum likelihood itself is limited as an inference paradigm, as it uses information contained only within a small part of the likelihood landscape. Our method, as an approximation that is exact only when clusters are of roughly equal sizes, may also suffer from estimation bias. Third, the edgewise agglomerative approach embodied by Louvain-style algorithms is limited in applicability to affinity functions that promote homogeneity within edges. Alternative inference paradigms may ameliorate some or all of these limitations. Within the framework of maximum likelihood inference, directly maximizing a profile likelihood offers an intriguing alternative to coordinate ascent (*49*). While all maximum likelihood methods are equivalent insofar as they optimize the same objective function, algorithmic properties such as runtime and propensity to be trapped in undesirable local optima may vary between different approaches. Fully Bayesian treatments (*74*) offer another promising path, although these are sometimes limited in their computational scalability. Variational belief propagation (*45*, *46*) provides an intriguing compromise, achieving considerable scalability in exchange for several approximations. Recent work (*35*) has made progress in this direction, but several questions related to scalability and behavior in nonuniform hypergraphs remain extant. Belief propagation methods for scalable inference with more general affinity functions would be of particular practical interest.

There are also several important directions of theoretical development. One of these is the question of detectability in the DCHSBM. Because the DCHSBM is more flexible than the dyadic DCSBM, the theory of detectability in this model may be substantially more complex. Another direction concerns the properties of the dyadic modularity objective that extend to the hypergraph modularity objectives discussed here. In addition to its role as a comparison against null models (*75*) and as a term in the DCSBM likelihood (*48*), the dyadic modularity also expresses the stability of diffusion processes on graphs (*76*) and the energy of discrete surface tensions defined on graphs (*77*). Extensions of these properties, or explanations of why they fail to generalize, would be helpful for both theorists and practitioners.

SUPPLEMENTARY MATERIALS

Supplementary material for this article is available at http://advances.sciencemag.org/cgi/content/full/7/28/eabh1303/DC1

REFERENCES AND NOTES

1. M. O. Jackson, *Social and Economic Networks* (Princeton Univ. Press, 2008).
2. D. Easley, J. Kleinberg, *Networks, Crowds, and Markets: Reasoning about a Highly Connected World* (Cambridge Univ. Press, 2010).
3. M. E. J. Newman, *Networks: An Introduction* (Oxford Univ. Press, 2010).
4. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, U. Alon, Network motifs: Simple building blocks of complex networks. *Science* **298**, 824–827 (2002).
5. A. R. Benson, D. F. Gleich, J. Leskovec, Higher-order organization of complex networks. *Science* **353**, 163–166 (2016).
6. A. R. Benson, R. Abebe, M. T. Schaub, A. Jadbabaie, J. Kleinberg, Simplicial closure and higher-order link prediction. *Proc. Natl. Acad. Sci. U.S.A.* **115**, E11221–E11230 (2018).
7. R. Lambiotte, M. Rosvall, I. Scholtes, From networks to optimal higher-order models of complex systems. *Nat. Phys.* **15**, 313–320 (2019).
8. F. Battiston, G. Cencetti, I. Iacopini, V. Latora, M. Lucas, A. Patania, J.-G. Young, G. Petri, Networks beyond pairwise interactions: Structure and dynamics. *Phys. Rep.* **874**, 1–92 (2020).
9. L. Torres, A. S. Blevins, D. S. Bassett, T. Eliassi-Rad, The why, how, and when of representations for complex systems. arXiv:2006.02870 [cs.SI] (4 June 2020).
10. P. Li, O. Milenkovic, *35th International Conference on Machine Learning, ICML 2018* [International Machine Learning Society (IMLS), 2018], pp. 4690–4719.
11. G. F. de Arruda, G. Petri, Y. Moreno, Social contagion models on hypergraphs. *Phys. Rev. Res.* **2**, 023032 (2020).
12. R. Sahasrabuddhe, L. Neuhäuser, R. Lambiotte, Modelling non-linear consensus dynamics on hypergraphs. *J. Phys. Complex.* **2**, 025006 (2020).
13. N. Veldt, A. Wirth, D. F. Gleich, Parameterized correlation clustering in hypergraphs and bipartite graphs in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Association for Computing Machinery, 2020), pp. 1868–1876.
14. M. A. Porter, J.-P. Onnela, P. J. Mucha, Communities in networks. *Notices Amer. Math. Soc.* **56**, 1082–1097 (2009).
15. S. Fortunato, D. Hric, Community detection in networks: A user guide. *Phys. Rep.* **659**, 1–44 (2016).
16. G. Ballard, A. Druinsky, N. Knight, O. Schwartz, Hypergraph partitioning for sparse matrix-matrix multiplication. *ACM Trans. Parallel Comput.* **3**, 1–34 (2016).
17. I. Kabiljo, B. Karrer, M. Pundir, S. Pupyrev, A. Shalita, Social hash partitioner: A scalable distributed hypergraph partitioner. *Proc. VLDB Endow.* **10**, 1418–1429 (2017).
18. G. Karypis, R. Aggarwal, V. Kumar, S. Shekhar, Multilevel hypergraph partitioning: Applications in VLSI domain. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **7**, 69–79 (1999).
19. S. Agarwal, J. Lim, L. Zelnik-Manor, P. Perona, D. Kriegman, S. Belongie, in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05 (IEEE Computer Society, 2005), pp. 838–845.
20. D. Zhou, J. Huang, B. Schölkopf, in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06 (MIT Press, 2006), pp. 1601–1608.
21. N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, P. Talukdar, Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Adv. Neural Inf. Process. Syst.*, 1511–1522 (2019).
22. Z. Tian, T. Hwang, R. Kuang, A hypergraph-based learning algorithm for classifying gene expression and arrayCGH data with prior knowledge. *Bioinformatics* **25**, 2831–2838 (2009).
23. P. Li, O. Milenkovic, Submodular hypergraphs: P-Laplacians, Cheeger inequalities, and spectral clustering in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett, Eds. (NeurIPS, 2017), pp. 2308–2318.
24. N. Neubauer, K. Obermayer, Towards community detection in k-partite k-uniform hypergraphs in *Proceedings of the 2009 Workshop on Analyzing Networks and Learning with Graphs* (NeurIPS, 2009), pp. 1–9.
25. C. E. Tsourakakis, J. Pachocki, M. Mitzenmacher, Scalable motif-aware graph clustering in *Proceedings of the 26th International Conference on World Wide Web* (Association for Computing Machinery, 2017), pp. 1451–1460.
26. K. Nowicki, T. A. B. Snijders, Estimation and prediction for stochastic blockstructures. *J. Am. Stat. Assoc.* **96**, 1077–1087 (2001).
27. P. D. Hoff, A. E. Raftery, M. S. Handcock, Latent space approaches to social network analysis. *J. Am. Stat. Assoc.* **97**, 1090–1098 (2002).
28. E. M. Airoldi, D. M. Blei, S. E. Fienberg, E. P. Xing, Mixed membership stochastic blockmodels. *J. Mach. Learn. Res.* **9**, 1981–2014 (2008).
29. B. Karrer, M. E. J. Newman, Stochastic blockmodels and community structure in networks. *Phys. Rev. E* **83**, 016107 (2011).
30. J. Yang, J. Leskovec, in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, Overlapping community detection at scale: A nonnegative matrix factorization approach (Association for Computing Machinery, 2013), pp. 587–596.
31. T. P. Peixoto, Hierarchical block structures and high-resolution model selection in large networks. *Phys. Rev. X* **4**, 011047 (2014).
32. A. Athreya, D. E. Fishkind, M. Tang, C. E. Priebe, Y. Park, J. T. Vogelstein, K. Levin, V. Lyzinski, Y. Qin, Statistical inference on random dot product graphs: A survey. *J. Mach. Learn. Res.* **18**, 8393–8484 (2017).
33. D. Ghoshdastidar, A. Dukkipati, Consistency of spectral partitioning of uniform hypergraphs under planted partition model. *Adv. Neural Inf. Process. Syst.* **27**, 397–405 (2014).
34. C. Kim, A. S. Bandeira, M. X. Goemans, Stochastic block model for hypergraphs: Statistical limits and a semidefinite programming approach. arXiv:1807.02884 [math.PR] (8 July 2018).
35. M. C. Angelini, F. Caltagirone, F. Krzakala, L. Zdeborová, Spectral detection on sparse hypergraphs, in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing, Allerton 2015* (IEEE, 2016), pp. 66–73.

36. Z. T. Ke, F. Shi, D. Xia, Community detection for hypergraph networks via regularized tensor power iteration. arXiv:1909.06503 [stat.ME] (14 September 2019).
37. P. S. Chodrow, Configuration models of random hypergraphs. *J. Complex Netw.* **8**, cnaa018 (2020).
38. M. E. J. Newman, M. Girvan, Finding and evaluating community structure in networks. *Phys. Rev. E* **69**, 026113 (2004).
39. T. Kumar, S. Vaidyanathan, H. Ananthapadmanabhan, S. Parthasarathy, B. Ravindran, Hypergraph clustering by iteratively reweighted modularity maximization. *Appl. Netw. Sci.* **5**, 52 (2020).
40. B. Kamiński, V. Poulin, P. Prałat, P. Szufel, F. Théberge, Clustering via hypergraph modularity. *PLOS ONE* **14**, e0224307 (2019).
41. N. Veldt, A. Wirth, D. F. Gleich, A correlation clustering framework for community detection in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Association for Computing Machinery, 2020), pp. 1868–1876.
42. D. B. Larremore, A. Clauset, A. Z. Jacobs, Efficiently inferring community structure in bipartite networks. *Phys. Rev. E* **90**, 012805 (2014).
43. M. Gerlach, T. P. Peixoto, E. G. Altmann, A network approach to topic models. *Sci. Adv.* **4**, eaaq1360 (2018).
44. T.-C. Yen, D. B. Larremore, Community detection in bipartite networks with stochastic blockmodels. *Phys. Rev. E* **102**, 032309 (2020).
45. A. Decelle, F. Krzakala, C. Moore, L. Zdeborová, Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Phys. Rev. E* **84**, 066106 (2011).
46. P. Zhang, C. Moore, Scalable detection of statistically significant communities and hierarchies, using message passing for modularity. *Proc. Natl. Acad. Sci. U.S.A.* **111**, 18144–18149 (2014).
47. T. P. Peixoto, Merge-split Markov chain Monte Carlo for community detection. *Phys. Rev. E* **102**, 012305 (2020).
48. M. E. J. Newman, Equivalence between modularity optimization and maximum likelihood methods for community detection. *Phys. Rev. E* **94**, 052315 (2016).
49. P. J. Bickel, A. Chen, A nonparametric view of network models and newman–girvan and other modularities. *Proc. Natl. Acad. Sci. U.S.A.* **106**, 21068–21073 (2009).
50. M. Jerrum, G. B. Sorkin, The Metropolis algorithm for graph bisection. *Discrete Appl. Math.* **82**, 155–175 (1998).
51. A. Condon, R. M. Karp, Algorithms for graph partitioning on the planted partition model. *Random Struct. Algor.* **18**, 116–140 (2001).
52. N. Veldt, A. R. Benson, J. Kleinberg, Hypergraph cuts with general splitting functions. arXiv:2001.02817 [cs.DS] (9 January 2020).
53. M. Deveci, K. Kaya, B. Uçar, Ü. V. Çatalyürek, Hypergraph partitioning for multiple communication cost metrics: Model and methods. *J. Parallel Distrib. Com.* **77**, 69–83 (2015).
54. G. Karypis, V. Kumar, Multilevel *k*-way hypergraph partitioning. *VLSI Design* **11**, 285–300 (2000).
55. B. Hendrickson, T. G. Kolda, Graph partitioning models for parallel computing. *Parallel Comput.* **26**, 1519–1534 (2000).
56. L. Zhang, T. P. Peixoto, Statistical inference of assortative community structures. *Phys. Rev. Res.* **2**, 043271 (2020).
57. D. F. Gleich, M. W. Mahoney, in *Handbook of Big Data*, P. Bühlmann, P. Drineas, M. Kane, M. van der Laan, Eds. (Chapman & Hall/CRC Handbooks of Modern Statistical Methods, CRC Press, 2016).
58. J. Reichardt, S. Bornholdt, Statistical mechanics of community detection. *Phys. Rev. E* **74**, 016110 (2006).
59. N. Veldt, D. F. Gleich, A. Wirth, in *Proceedings of the 2018 World Wide Web Conference*, WWW '18 (International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018), p. 439–448.
60. V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks. *J. Stat. Mech.* P10008 (2008).
61. E. Abbe, Community detection and stochastic block models: Recent developments. *J. Mach. Learn. Res.* **18**, 1–86 (2018).
62. R. R. Nadakuditi, M. E. J. Newman, Graph spectra and the detectability of community structure in networks. *Phys. Rev. Lett.* **108**, 188701 (2012).
63. D. Ghoshdastidar, A. Dukkipati, Consistency of spectral hypergraph partitioning under planted partition model. *Ann. Statist.* **45**, 289–315 (2017).
64. J. Stehlé, N. Voirin, A. Barrat, C. Cattuto, L. Isella, J.-F. Pinton, M. Quaggiotto, W. V. den Broeck, C. Régis, B. Lina, P. Vanhems, High-resolution measurements of face-to-face contact patterns in a primary school. *PLOS ONE* **6**, e23176 (2011).
65. R. Mastrandrea, J. Fournet, 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).
66. J. H. Fowler, Connecting the Congress: A study of cosponsorship networks. *Political Anal.* **14**, 456–487 (2006).
67. J. H. Fowler, Legislative cosponsorship networks in the US House and Senate. *Soc. Netw.* **28**, 454–465 (2006).
68. C. Stewart, J. Woon, Congressional committee assignments, 103rd to 115th congresses, 1993–2017 (2021).
69. I. Amburg, N. Veldt, A. Benson, Clustering in graphs and hypergraphs with categorical edge labels in *Proceedings of The Web Conference 2020* (Association for Computing Machinery, 2020), pp. 706–717.
70. J. Adamczak, Y. Deldjoo, F. B. Moghaddam, P. Knees, G. P. Leyson, P. Monreal, Session-based hotel recommendations dataset: As part of the ACM recommender system challenge 2019. *ACM Trans. Intell. Syst. Technol.* **12**, 1–20 (2020).
71. G. Schwarz, Estimating the dimension of a model. *Ann. Statist.* **6**, 461–464 (1978).
72. L. Peel, D. B. Larremore, A. Clauset, The ground truth about metadata and community detection in networks. *Sci. Adv.* **3**, e1602548 (2017).
73. B. H. Good, Y.-A. de Montjoye, A. Clauset, Performance of modularity maximization in practical contexts. *Phys. Rev. E* **81**, 046106 (2010).
74. T. P. Peixoto, Bayesian stochastic blockmodeling, in *Advances in Network Clustering and Blockmodeling*, P. Doreian, V. Batagelj, A. Ferligoj, Eds. (John Wiley & Sons, Ltd, 2019), pp. 289–332.
75. M. E. J. Newman, Modularity and community structure in networks. *Proc. Natl. Acad. Sci. U.S.A.* **103**, 8577–8582 (2006).
76. J.-C. Delvenne, S. N. Yaliraki, M. Barahona, Stability of graph communities across time scales. *Proc. Natl. Acad. Sci. U.S.A.* **107**, 12755–12760 (2010).
77. Z. M. Boyd, M. A. Porter, A. L. Bertozzi, Stochastic block models are a discrete surface tension. *J. Nonlinear Sci.* **30**, 2429–2462 (2020).

Acknowledgments: We are grateful to T. de Paula Peixoto for pointing out the distinction between exact and approximate maximum likelihood estimation of θ and Ω as discussed at the end of the "Symmetric modularities" section. **Funding:** This research was supported, in part, by ARO Award W911NF19-1-0057, ARO MURI, NSF Award DMS-1830274, and JP Morgan Chase & Co. **Author contributions:** P.S.C., N.V., and A.R.B. all conceived the research, developed algorithms, performed experiments, and wrote the manuscript. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the paper are present in the paper and/or the Supplementary Materials. Software and data sufficient to reproduce and extend the experiments and analysis presented are available at the following repository: https://github.com/PhilChodrow/HypergraphModularity. The data are also hosted in packaged format at www.cs.cornell.edu/~arb/data/#hyperlabels.

Submitted 17 February 2021
Accepted 24 May 2021
Published 7 July 2021
10.1126/sciadv.abh1303

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

Generative hypergraph clustering: From blockmodels to modularity

Philip S. ChodrowNate VeldtAustin R. Benson

Sci. Adv., 7 (28), eabh1303. • DOI: 10.1126/sciadv.abh1303