PLOS COMPLEX SYSTEMS

RESEARCH ARTICLE

FastEnsemble: Scalable ensemble clustering on large networks

Yasamin Tabatabaee, Eleanor Wedell, Minhyuk Park, Tandy Warnow*

Siebel School of Computing and Data Science, University of Illinois Urbana-Champaign, Urbana, Illinois, United States of America

* warnow@illinois.edu

OPEN ACCESS

Citation: Tabatabaee Y, Wedell E, Park M, Warnow T (2025) FastEnsemble: Scalable ensemble clustering on large networks. PLoS Complex Syst 2(10): e0000069. https://doi.org/10.1371/journal.pcsy.0000069

Editor: Tobias Braun, Leipzig University: Universitat Leipzig, GERMANY

Received: February 24, 2025

Accepted: August 28, 2025

Published: October 1, 2025

Data availability statement: The code and scripts used in this study are available at https://github.com/ytabatabaee/fast-ensemble. The data are available at https://github.com/ytabatabaee/ensemble-clustering-data.

Funding: TW received a grant from the US National Science Foundation, number 2402559. The funder did not play any role in the study design, data collection and analysis, decision to publish, or preparation of the manuscript.

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

Abstract

Many community detection algorithms are inherently stochastic, leading to variations in their output depending on input parameters and random seeds. This variability makes the results of a single run of these algorithms less reliable. Moreover, different clustering algorithms, optimization criteria (e.g., modularity and the Constant Potts model), and resolution values can result in substantially different partitions on the same network. Consensus clustering methods, such as Ensemble Clustering for Graphs (ECG) and FastConsensus, have been proposed to reduce the instability of non-deterministic algorithms and improve their accuracy by combining a set of partitions resulting from multiple runs of a clustering algorithm. In Complex Networks and their Applications 2024, we introduced FastEnsemble, a new consensus clustering method; here we present a more extensive evaluation of this method. Our results on both real-world and synthetic networks show that FastEnsemble produces more accurate clusterings than two other consensus clustering methods, ECG and FastConsensus, for many model conditions. Furthermore, FastEnsemble is fast enough to be used on networks with more than 3 million nodes, and so improves on the speed and scalability of FastConsensus. Finally, we showcase the utility of consensus clustering methods in mitigating the effect of resolution limit and clustering networks that are only partially covered by communities.

Author summary

Consensus (ensemble) clustering methods, such as FastConsensus and Ensemble Clustering for Graphs (ECG), combine partitions from multiple runs of the same clustering algorithm, in order to improve stability and accuracy of the output partition. In this study, we present a new ensemble clustering method, FastEnsemble, and show that it provides improved accuracy under many conditions compared to FastConsensus and ECG. We show results using FastEnsemble with Leiden optimizing modularity or the Constant Potts model (CPM) and the Louvain algorithm. We show that FastEnsemble and other consensus clustering methods can reduce the effect of resolution limit for both modularity- and CPM-optimization. Finally, we demonstrate that consensus clustering

methods can improve community detection over modularity-optimization using Leiden on networks with both clusterable and unclusterable regions.

Introduction

Community detection methods are commonly used to analyze the community structure of complex networks, where a community is a set of nodes that satisfies criteria such as being dense (more edges than expected), well-connected (i.e., not having a small edge cut) [1], and reasonably separable from the rest of the network. Because of the broad applicability of community detection, many community detection methods have been developed that vary in approach (e.g., spectral clustering, heuristics for NP-hard discrete optimization problems, or machine learning approaches), objective (e.g., disjoint clustering or overlapping clustering) and type of network they are designed for (e.g., citation networks, biological networks, and social networks) [2–8].

One difficulty in using community detection algorithms is that most do not produce a unique output on the same network in multiple runs. In most cases, this variability arises from the stochastic nature of the algorithm, which incorporates randomness in the clustering process. As a result, the output can vary depending on factors such as random seeds, initial conditions, and tie-breaking rules used in the algorithm [9,10]. For instance, in the Leiden algorithm [11], changes in random seeds can substantially affect the final clustering [12].

On the other hand, even when the clustering algorithm is deterministic, the output may vary based on parameters given to the method, such as the resolution parameter for clustering optimizing modularity [13] or the Constant Potts Model (CPM) [11,14]. In many cases, it is not immediately clear which optimization function or algorithmic parameters will yield the best partition for a given network. This unpredictability, combined with the challenge of selecting the most suitable optimization criteria and parameters, highlights the need for systematic methods to evaluate and compare partitions, either qualitatively or quantitatively. Alternatively, combining information from multiple partitions can lead to a more robust and representative community structure of the network.

To address these challenges, consensus (or ensemble) clustering approaches have been proposed [9,15–23] with the goal of reducing the noise in the final clustering, which arises from the stochasticity of methods. Previous studies have shown that these consensus approaches could lead to more robust and stable partitions, and improve the accuracy of the output clustering [9,16,17].

A class of consensus clustering methods, introduced in [9], take a network G as input and run a clustering algorithm (such as Louvain [24,25] with different random seeds) on it n_p times to get n_p different partitions. These partitions are then analyzed to construct a co-classification matrix, which captures how frequently each pair of nodes are co-clustered. Using this matrix, a new weighted network G' is created and subsequently re-clustered n_p times. This iterative process continues until G' stabilizes, converging to a stationary network. Several variations of this consensus approach have been proposed in the literature [9,16,17].

Scalability remains a challenge for these approaches, as constructing the co-classification matrix is computationally intensive when the network is large. FastConsensus [16] addresses this issue by employing a sampling technique, computing the co-classification matrix only for a subset of node pairs. Another recent and promising consensus method is Ensemble Clustering for Graphs (ECG) [26], which simplifies the process compared to FastConsensus by combining partitions in a single step rather than through iterative refinements.

In a recent paper published in Complex Networks and their Applications 2024 [27], we introduced FastEnsemble, a new ensemble clustering method. While FastEnsemble shares design similarities with ECG and FastConsensus, FastEnsemble is designed to support both modularity and CPM optimization, whereas ECG and FastConsensus only work with modularity optimization. Additionally, FastEnsemble eliminates much of the technical complexity of FastConsensus, allowing it to scale efficiently to large networks. To evaluate its performance, we tested a simplified version of FastEnsemble using Leiden with both modularity and CPM-based optimization on large synthetic networks generated with the Lancichinetti-Fortunato-Radicchi (LFR) benchmark software [28,29]. We compared FastEnsemble against FastConsensus and ECG [30,31] in terms of accuracy and scalability and demonstrated cases where FastEnsemble provided an advantage over these methods. Furthermore, we demonstrated that consensus clustering methods can help mitigate the resolution limit problem [32] and improve clustering accuracy on networks where only a portion of the network has community structure.

In this extended study, we expand on our previous work in several directions. While [27] focused on a limited set of networks for algorithm design experiments, varying only in terms of the mixing parameter, we expand our analysis to include synthetic networks with a wider range of densities and sizes. Additionally, whereas the original study demonstrated the impact of the resolution limit only for modularity-based optimization, we show that CPM-based optimization is also susceptible to the resolution limit at sufficiently small resolution values. We further extend our experiments by incorporating additional networks, including both real-world networks and synthetic networks partially composed of Erdős-Rényi graphs and tree-of-cliques networks. Finally, unlike [27], which only used Leiden for modularity optimization, we include the Louvain algorithm in our experiments to ensure a fair comparison with ECG and FastConsensus, both of which also utilize Louvain.

Preliminaries

In this section, we introduce the notation and concepts related to networks and clustering used throughout this paper.

Notation and definitions. Let $N = G(V, E)$ be a network where V denotes the set of nodes and E denotes the set of edges, and let $n = |V|$ and $m = |E|$. A partition or clustering $\mathcal{P}$ of N divides the set of nodes V into k non-overlapping sets $C_1, C_2, \ldots, C_k$ such that each vertex belongs to exactly one cluster. We use clustering and partition interchangeably throughout this paper, and refer to each C_i as a cluster or community.

In a synthetic network, we will have known ground truth communities. In this study we will constrain all such communities to be internally connected, i.e., to not be comprised of two or more components.

For a fixed partition $\mathcal{P}$, let d_i^{in} indicate the degree of node v_i inside its own community and d_i^{out} indicate the degree of v_i outside its community. The total degree of v_i is therefore $d_i = d_i^{in} + d_i^{out}$. The estimated *mixing parameter* of the network for the partition $\mathcal{P}$ is defined as

$$\hat{\mu}_{(N,\mathcal{P})} = \frac{1}{n} \sum_{i \in \{1,\ldots,n\}} \frac{d_i^{out}}{d_i^{in} + d_i^{out}}, \tag{1}$$

which is equivalent to the average ratio of the number of neighbors of a node outside its community to its total degree. When $\mathcal{P}$ is the ground-truth community structure of a network, the mixing parameter serves as an indicator of clustering difficulty for that network; small mixing parameters signify networks that are generally easy to cluster [28], whereas large mixing

parameters correspond to networks that have less clear boundaries around their clusters and are therefore more difficult to cluster correctly.

For a set of partitions $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{n_p}$ on network N with n vertices, the *co-classification* or *consensus* matrix A is an $n \times n$ matrix where each row and each column corresponds to a vertex in N. The entry A_{ij} represents the proportion of the n_p partitions in which nodes v_i and v_j are co-clustered together [9].

Fast ensemble clustering

In this section, we describe the algorithm and implementation of FastEnsemble.

Algorithm. The input is a network N. FastEnsemble uses five algorithmic parameters: an integer n_p that indicates the number of partitions, a threshold $0 \leq t \leq 1$ for removing weak edges in the consensus matrix, the clustering method M that is applied n_p times to the input network N, the final clustering method M', and a Boolean parameter "weight" that determines whether to consider weights in the final step.

FastEnsemble has the following steps:

- Step 1: It first uses the specified clustering method M to generate n_p partitions of N. It then computes the entries of the co-classification matrix corresponding to the edge set of N.
- Step 2: It then builds a new network on the same node and edge set but with the edges weighted by the entries in the co-classification matrix, i.e., the fraction of the clusterings in which the endpoints of an edge are in the same cluster. If a given edge has weight less than t, then the edge is removed from the network; hence the new network N' can have fewer edges than the original network.
- Step 3: If weight = True, then N' is clustered using M'. If weight=False, then the edge weights are ignored before M' is applied to N'.

As an example, suppose the clustering method M is Louvain, the final clustering method M' is Leiden optimizing modularity, $t = 0.5$, $np = 10$, and weight=True. In Step 1, FastEnsemble clusters the input network N, $n_p = 10$ times using Louvain. We assign weights to each edge of N as follows: for every edge (x,y) in the network, letting k be the number of times the pair of nodes x,y are co-clustered (i.e., appear together in a cluster), the edge (x,y) is assigned weight $k/10$. In Step 2, every edge whose weight is less than $t = 0.5$ is removed from the network, which can reduce the number of edges. Hence, the network that is clustered in Step 3 will contain a subset (and possibly proper subset) of edges of the input network, and these edges will have values between 0 and 1. In Step 3, this edge-weighted network is clustered using Leiden optimizing modularity, producing the final clustering.

Some comments about the algorithmic parameters may be helpful. Although considering edge weights may be desirable, the user may wish to cluster the final network without reference to the edge weights; therefore, we allow the user to ignore the edge weights in Step 3. Increasing the number n_p of partitions can enhance accuracy and stability but comes with a computational cost. To ensure scalability for large networks, we set the default value of $n_p = 10$. We choose the default value for the parameter t based on a set of algorithm design experiments.

Strict consensus. We refer to a special case of FastEnsemble that uses $t = 1$ as *Strict Consensus Clustering* in the experiments. In this variant, an edge (x,y) remains in the weighted network if and only if x and y are co-clustered in all n_p partitions.

Implementation. FastEnsemble is a generalized framework that can be used with a single clustering method (the default version) or with a combination of clustering methods (its

advanced version, see Sect A, S1 Appendix). It is currently implemented for use with Leiden optimizing modularity (referred to as "Leiden-mod" in the experiments), Leiden optimizing CPM ("Leiden-CPM"), and the Louvain algorithm, which optimizes modularity. However, additional clustering methods can be easily incorporated into the implementation.

Runtime analysis. The first step of FastEnsemble runs the specified clustering method (Leiden or Louvain in the current implementation) n_p times. The runtime of both Louvain and Leiden are $O(L|E|)$ where L is the total number of iterations of the algorithm and $|E|$ is the number of edges in the network [33]. Therefore, the runtime for this step is $O(n_pL|E|)$. Given these clusterings, FastEnsemble computes the entries of the co-classification matrix corresponding to edges of N in $O(n_p|E|)$ by looking at the proportion of times each edge $e \in E$ appears in the n_p partitions. Therefore, the total runtime of step 1 is $O(n_pL|E|)$. In Step 2, FastEnsemble builds a new network using the co-classification matrix, which can be done in $O(|E|)$ time. The third step involves applying the final clustering method on the new weighted network, which has a time complexity of $O(L|E|)$ when the clustering method is Leiden or Louvain. Therefore, the overall runtime of FastEnsemble is $O(n_pL|E|)$.

Performance study

Networks

We used both real-world and synthetic networks, some available from prior studies, and some generated for this study. Table 1 provides a summary of empirical statistics of the networks, including network size and mixing parameters [34]. Networks that have mixing parameters of 0.5 or larger are considered challenging to cluster while networks with much smaller mixing parameters are generally easy to cluster [28,35].

Algorithm design experiments. For the algorithm design experiment, we generated LFR networks using parameters similar to those used in [16], but with a modified exponent for the cluster size distribution to better match properties of real-world networks (see also Sect B.1.1 in S1 Appendix). The default model condition in this dataset consists of synthetic networks with 10,000 nodes, an average degree of 10, and estimated mixing parameter values ranging from 0.196 to 0.978 (note that the model mixing parameters, which are used to generate the networks, are drawn from $0.1, 0.2, \ldots, 0.9$, but the resultant mixing parameters are different). We vary the network density (i.e., average node degree) between 5 and 20 and number of nodes between 1,000 and 100,000 to create additional networks that allow us to evaluate FastEnsemble under a range of model conditions. In total, the algorithm design dataset has 45 model conditions, and each model condition has one replicate (i.e., one network). These networks have mixing parameters between 0.196 to 0.978 (Table 1).

Testing experiments. The testing experiments used two different real-world networks and several sets of synthetic networks. The real-world networks are taken from SNAP [36], and include two of the networks from that collection (DLBP and Amazon Products) that have real-world ground-truth communities. The synthetic networks come in different sets. One set, taken from [1], contains LFR [28] networks based on parameters obtained from five real-world networks clustered using Leiden-mod or Leiden-CPM. The five real-world networks are cit_hepph, the Curated Exosome Network (CEN), Open Citations (OC), wiki_topcats, and cit_patents. Two of these LFR networks based on CPM clustering contained a large percentage of ground truth clusters that were internally disconnected and were therefore excluded from the experiments in [1] as well as from this study. Additionally, LFR failed to generate a network for one model condition from a Leiden-CPM clustering. Thus, there are 5 LFR networks that are based on Leiden-mod clusterings and 22 LFR networks that are based on Leiden-CPM clusterings. The networks based on Leiden-mod clusterings have small mixing

Table 1. Empirical statistics of the networks used in this study.

Network	Expt.	# nodes	# edges	avg. mixing param.
Synthetic Networks				
LFR algorithm design [27],(*)	1	1,000-1,000,000	5708-600,227	0.196-0.978
LFR cit_hepph MOD [1]	2	34,546	~ 431*K*	0.155
LFR wiki_topcats MOD [1]	2	1,791,489	~ 24*M*	0.199
LFR cen MOD [1]	2	3,000,000	~ 21*M*	0.180
LFR OC MOD [1]	2	3,000,000	~ 55*M*	0.129
LFR cit_patents MOD [1]	2	3,774,768	~ 16*M*	0.114
LFR cit_hepph CPM [1]	3	34,546	~ 431*K*	0.086-0.781
LFR wiki_topcats CPM [1]	3	1,791,489	~ 24*M*	0.379-0.793
LFR cen CPM [1]	3	3,000,000	~ 21*M*	0.402-0.646
LFR OC CPM [1]	3	3,000,000	~ 55*M*	0.407-0.871
LFR cit_patents CPM [1]	3	3,774,768	~ 16*M*	0.211-0.807
Ring-of-cliques [27]	4	90-10,000	4140-460,000	0.02
Tree-of-cliques (*)	4	90-5,000	4139-229,999	0.018
Erdős-Rényi [27]	5	1000	470-50,025	0.625-1.0
Erdős-Rényi+LFR [27]	5	2000	4776-53,917	0.486-0.572
Erdős-Rényi+ring (*)	5	2000	5100-54,470	0.40-0.51
Real-world Networks				
Amazon Products [36]	6	334,863	925,872	0.845
DBLP [36]	6	317,080	1,049,866	0.991

Notes: We report the number of nodes, number of edges, and the average mixing parameter for each network; when a network collection is indicated, we provide the range of these values. The mixing parameter is measured for the "ground truth" community structure; for Erdős-Rényi graphs, we assume the ground truth clustering has each node forming its own community. The rows for Experiment 3 each represent up to five different networks, each generated based on a Leiden-CPM clustering with different resolution values of the specified real-world network. Mixing parameters for real-world networks used in Experiment 6 (bottom two rows) are derived from the top 5000 clusters according to [38], based on cluster quality. For networks from prior publications, we provide a citation to that publication; an asterisk (*) indicates that the networks are newly created for this study. The LFR training data are partly from a prior publication and partly new.

https://doi.org/10.1371/journal.pcsy.0000069.t001

parameters ranging from 0.114–0.199, while the networks based on Leiden-CPM clusterings have mixing parameters that range from 0.086 to 0.871 (Table 1). The five LFR networks based on Leiden-mod clusterings of real-world networks are used to evaluate the modularity-based consensus clustering methods ECG, FastConsensus, and FastEnsemble using Leiden-mod in Experiment 2. The 22 networks based on Leiden-CPM clusterings are used to evaluate FastEnsemble using Leiden-CPM, in comparison to Leiden-CPM, in Experiment 3.

We also included LFR synthetic networks where the resolution limit [32] is known to cause a problem for modularity-based clustering; these were used to evaluate both modularity-based clusterings and CPM-based clusterings in Experiment 4. There are six ring-of-cliques networks and five tree-of-cliques networks in this collection. The ring-of-cliques networks have n cliques of size 10, each connected to the cliques on the two sides by a single edge. The tree-of-cliques networks are formed by taking a random tree on n nodes and replacing each node by a clique of size 10. The mixing parameters for these networks are very small, in the 0.018–0.02 range (Table 1).

We studied 14 networks that have at least half of the nodes not in any clusters in Experiment 5. Some of these networks are Erdős-Rényi graphs [37], and others are hybrid networks that contain Erdős-Rényi graphs as subnetworks. By construction, half of each hybrid network has no community structure (i.e., every node is in a singleton cluster) and the other half has very strong community structure, as reflected by a very low mixing parameter. The combination of these two subgraphs produces mixing parameters in the range 0.40–0.572 (Table 1).

The testing experiments also include two real-world networks taken from the SNAP repository [36]; these networks come with ground-truth clusters that have been evaluated for cluster quality by Yang and Leskovec [38], so that the top 5000 clusters are provided. We use those networks with their top 5000 clusters in a final experiment. Since these ground truth clusters are overlapping, to enable a fair comparison to an estimated clustering, we modify the ground truth clusters to make them pairwise disjoint by removing all nodes that appear in two or more of these top 5000 clusters. Each clustering method is applied to the entire network, but then evaluated only with respect to these modified ground-truth clusters.

Methods

We evaluated FastEnsemble, ECG, and FastConsensus used with base methods for modularity optimization (Louvain for FastEnsemble and ECG, and Leiden-mod for FastConsensus). Since ECG and FastConsensus consider weighted edges, for these analyses we also ran FastEnsemble with weights on the edges in the final clustering step. We included Leiden-mod for a baseline comparison. We also evaluated FastEnsemble used with Leiden-CPM as the base method, and compared it to Leiden-CPM; to keep this comparison simple, we did not include weights on the edges in the final clustering step for FastEnsemble.

ECG is similar to FastEnsemble, as both follow a two-step process: first, generating multiple partitions using a clustering algorithm with different random seeds, and second, combining these partitions into a final clustering by assessing the fraction of times each node pair is co-clustered and then applying a clustering algorithm to the resulting weighted network.

However, there are two key differences between ECG and FastEnsemble. First, ECG assigns a predefined minimum edge weight to edges that are not part of a 2-core (i.e., a subnetwork where every node is adjacent to at least two other nodes in the subnetwork) in the original graph. In contrast, FastEnsemble assigns high weights to edges whose endpoints are frequently co-clustered across partitions, regardless of their inclusion in a 2-core. As a result, ECG is less likely to co-cluster node pairs in the final consensus clustering if they do not belong to 2-cores in the input network, whereas FastEnsemble does not impose this structural constraint. Second, prior to the final clustering step, ECG is restricted to the Louvain algorithm (which optimizes modularity), whereas FastEnsemble has been implemented to work with Leiden or Louvain for modularity optimization, and with Leiden optimizing CPM. Furthermore, although not examined in this study, FastEnsemble can integrate partitions from two or more clustering algorithms (see Sect A in S1 Appendix).

Evaluation criteria

We evaluated accuracy on networks with known ground truth community structure using Normalized Mutual Information (NMI), Adjusted Mutual Information (AMI), and Adjusted Rand Index (ARI), as implemented in the Scikit-learn [39] library. For analyses on real-world networks (DBLP and Amazon Products), the NMI and AMI calculations were done using the graph-tool [40] library. Additionally, in some experiments, we report cluster size distributions to gain deeper insights into clusters. To further assess clustering accuracy, we computed false negative and false positive error rates. Treating both the true and estimated clusterings as equivalence relations—each defined by a set of node pairs where (x,y) belongs to the relation if and only if nodes x and y are in the same cluster—we define:

- False negatives (FN): Pairs present in the true clustering but missing in the estimated clustering.
- False positives (FP): Pairs present in the estimated clustering but absent in the true clustering.
- True positives (TP): Pairs present in both the true and estimated clusterings.
- True negatives (TN): Pairs absent from both clusterings.

Using these definitions, we report the False Negative Rate (FNR), False Positive Rate (FPR), and the F1-score computed as

$$FNR = \frac{FN}{FN + TP}, \quad FPR = \frac{FP}{FP + TN}, \quad F_1 = \frac{2TP}{2TP + FP + FN} \tag{2}$$

Please see Table 2 for the definitions of acronyms we use in this paper.

Experiments

We conducted six experiments, as outlined below. We evaluated accuracy by comparing the results to the ground truth community structure. For all experiments except ones on large networks, all analyses were allocated four hours of runtime and 64GB of memory without parallelism on the University of Illinois Campus Cluster. Any instances where a method failed to complete within this time limit were recorded.

- Experiment 1: We set the default for the threshold parameter t in FastEnsemble based on experiments using modularity optimization on a collection of algorithm design datasets. We also performed a set of experiments comparing FastEnsemble to ECG and FastConsensus on these algorithm design datasets.
- Experiment 2: We evaluated modularity-based consensus pipelines with respect to both accuracy and runtime on five LFR synthetic networks from [1], which are based on five real-world networks clustered using Leiden-mod. These networks have up to $\sim$ 3.8M nodes.
- Experiment 3: We evaluated FastConsensus used with Leiden-CPM with respect to accuracy and runtime on 22 LFR synthetic networks from [1], which are based on five real-world networks clustered using Leiden-CPM with different resolution parameters. These networks have up to $\sim$ 3.8M nodes.

Table 2. List of abbreviations.

Abbreviation	Definition
AMI	Adjusted Mutual Information
ARI	Adjusted Rand Index
CEN	Curated Exosome Network
CPM	Constant Potts Model
ECG	Ensemble Clustering for Graphs [26]
FE	FastEnsemble
FNR	False Negative Rate
FPR	False Positive Rate
Leiden-CPM	Leiden optimizing CPM
Leiden-Mod	Leiden optimizing Modularity
LFR	Lancichinetti-Fortunato-Radicchi [28]
NMI	Normalized Mutual Information
OC	Open Citations
SC	Strict Consensus

https://doi.org/10.1371/journal.pcsy.0000069.t002

- Experiment 4: We assessed the robustness of different modularity-based and CPM-based clustering methods to the resolution limit using ring-of-cliques and tree-of-cliques networks with up to 100K nodes.
- Experiment 5: We evaluated modularity-based consensus pipelines on networks where at least half of the network is an Erdős-Rényi graph and so the network has at most half of the nodes in non-singleton clusters.
- Experiment 6: We explored modularity- and CPM-based clusterings on two real-world networks from the SNAP repository.

Some of these experiments focused exclusively on modularity-based clusterings, while others examined CPM-based clusterings. Experiments 1 and 3 used networks with a range of mixing parameters, Experiments 2 and 4 used networks with low mixing parameters, and Experiment 5 examined networks with moderate to high mixing parameters (Table 1).

Results

Experiment 1: Algorithm design experiment

This experiment has two parts. In Experiment 1a, we set the default value for the threshold parameter t in FastEnsemble, so that edges with support below t in the co-classification matrix are removed from the weighted network. In Experiment 1b, we compared the default setting for FastEnsemble to ECG and FastConsensus.

Experiment 1a: Setting the default threshold value. In Fig 1 (left), we examine AMI, ARI, and NMI accuracy for FastEnsemble using four different settings for the threshold t (0.2,0.5,0.8, and 0.9) on networks with varying mixing parameters. For all settings of the threshold t and all criteria, the accuracy was highest when the mixing parameter was small and decreased as the mixing parameter increased; this is expected [28,35]. However, the choice of threshold impacted the accuracy for FastEnsemble. When considering AMI, there was little difference between FastEnsemble variants as the threshold is changed, except on the small mixing parameters, where the two larger threshold values ($t = 0.8$ and $t = 0.9$) had an advantage. For the other two criteria (ARI and NMI), the best accuracy across all networks was obtained using threshold values of $t = 0.8$ and $t = 0.9$, with $t = 0.9$ slightly outperforming $t = 0.8$ in terms of NMI and $t = 0.8$ slightly outperforming $t = 0.9$ in terms of ARI for moderate to high resolution values.

Evaluating results on one of the LFR networks with mixing parameter $\mu = 0.5$ and allowing t to vary between $0.1, 0.2, \ldots, 0.9, 1$ (Fig 1 (right)), FastEnsemble accuracy peaked at different threshold values, depending on the accuracy criterion. For AMI, it peaked at $t = 0.7$, and was slightly lower at $t = 0.6$ and $t = 0.8$. For ARI, it peaked at $t = 0.8$ and was only slightly lower at $t = 0.7$ and $t = 0.9$. For NMI, it peaked at $t = 0.8$ through $t = 1.0$, and was slightly lower at $t = 0.7$. Based on these results, we set the default value for t to be 0.8.

Experiment 1b: Comparing default FastEnsemble to ECG and FastConsensus on algorithm design networks. As seen in Fig 2 (top), accuracy declined for all methods as the model mixing parameter increases. For the default model condition, i.e., networks with 10,000 nodes and an average degree of 10, ECG achieved the highest accuracy for the two smallest mixing parameters (0.1 and 0.2). However, when the mixing parameter was 0.3 or higher, FastEnsemble outperformed the other methods. Both ECG and FastEnsemble consistently matched or surpassed Leiden-mod in accuracy. FastConsensus improved upon Leiden-mod for larger mixing parameters but was less accurate for smaller ones (μ values below 0.5). However, FastConsensus failed to converge in 14 out of the 45 model conditions (Table A in

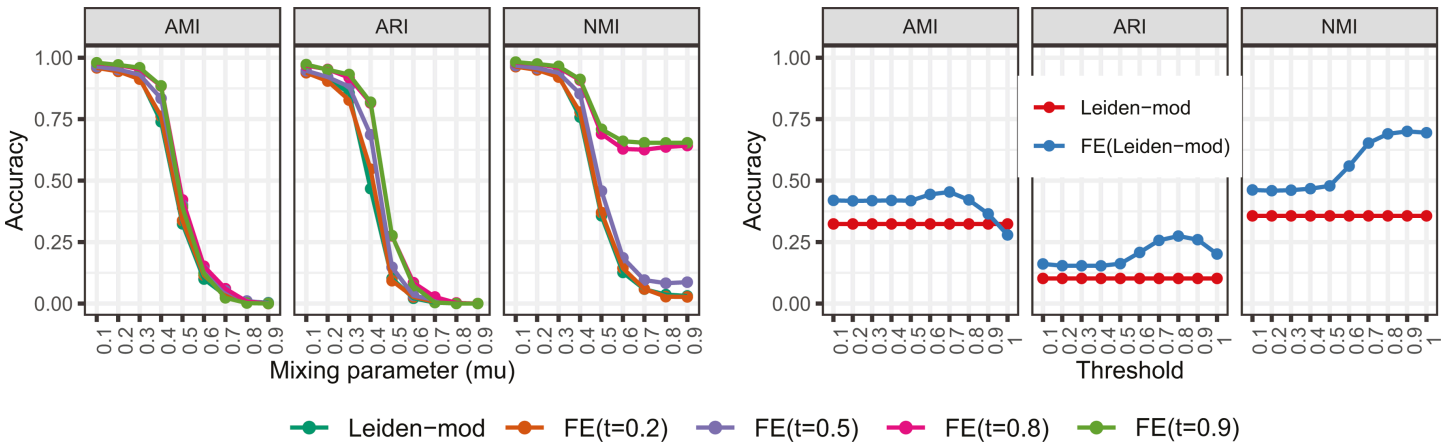

Fig 1. Experiment 1a: Setting the default value for *t* in FastEnsemble. Each plot shows AMI, ARI and NMI accuracy for Leiden-mod and FastEnsemble using four different threshold values on the default algorithm design networks with 10,000 nodes and average degree 10. Left: Accuracy as a function of the model mixing parameter (x-axis). Right: Accuracy as a function of the threshold value on the networks with model mixing parameter 0.5. FE stands for FastEnsemble. There is one network for each shown mixing parameter (left) or threshold (right); hence no error bars are shown.

https://doi.org/10.1371/journal.pcsy.0000069.g001

S1 Appendix) within the allotted four-hour time limit, including on all networks with 100K nodes.

We explored the impact of network density (i.e., average node degree). Based on average degrees of real-world networks taken from the Neutzschleuder catalog [41] with the same number of nodes, an average degree of 5 is sparse and an average degree of 20 is very dense (S1 Appendix, Fig A). Therefore, we let the average degree vary from 5 to 20. Results (Fig 2 (middle)) reveal the following trends. FastConsensus shows somewhat erratic responses to changes in density, sometimes improving and sometimes becoming less accurate as density increases (especially for high mixing parameters). For FastEnsemble and ECG, the impact of increasing density tended to be positive for both AMI and ARI accuracy, but was variable for NMI accuracy. We also note that the gap in accuracy between methods decreased as the density increased.

We next evaluated the impact of network size (number of nodes) as a function of the clustering method, mixing parameter, and accuracy criterion (Fig 2 (bottom)). FastConsensus failed to complete on many larger networks, and was not as accurate as either ECG or FastEnsemble when it did complete, so we limit the discussion to FastEnsemble, ECG, and Leiden-mod. Across all three accuracy criteria, Leiden-mod was never more accurate than ECG or FastEnsemble, and the gap between Leiden-mod and the consensus clustering methods increased as the network size increased. In comparing ECG and FastEnsemble, we see that FastEnsemble generally maintained its accuracy as network size increased, and in some cases improved in accuracy. ECG showed a similar response to increases in network size, but sometimes reduced in accuracy for the networks with the highest mixing parameters. Finally, although ECG and FastEnsemble were often close in accuracy for the smaller networks, as the network size increased, FastEnsemble tended to improve more than ECG, resulting in FastEnsemble either matching or improving on ECG for larger networks.

We also explored stability for ECG, FastEnsemble, and Leiden-mod on the default Algorithm Design datasets (average degree 10 and network size 10,000, and with varying mixing parameters). We ran each clustering method twice and measured the similarity between the two clusterings using AMI, ARI, and NMI. As seen in Fig 3, both FastEnsemble and ECG had

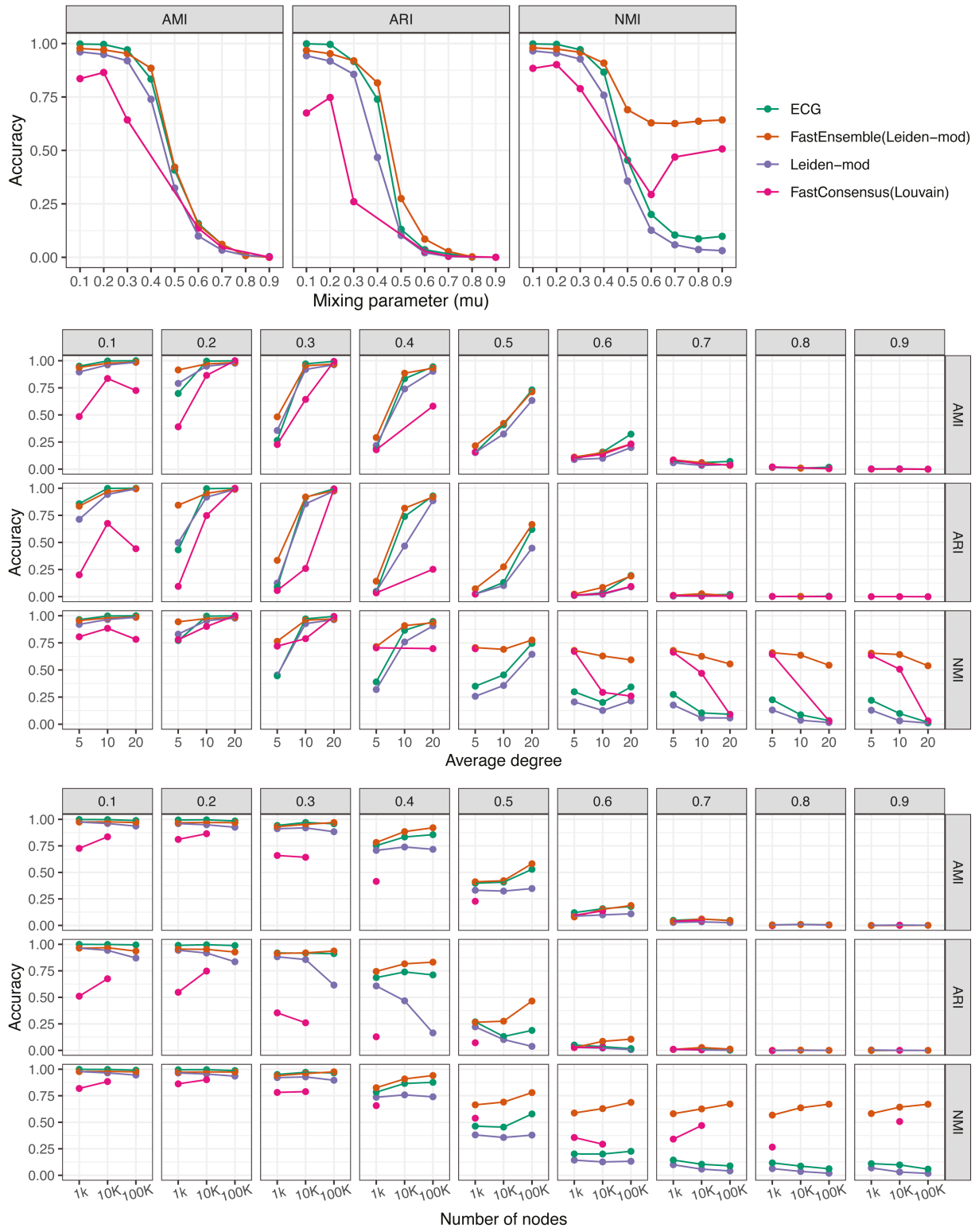

Fig 2. Experiment 1b: Evaluating modularity-based consensus clustering pipelines on the algorithm design datasets as a function of the mixing parameter. Results are shown for three consensus clustering methods and also Leiden-mod on the algorithm design datasets. Top: Accuracy on the default algorithm design datasets as the mixing parameter for the network changes (values on the x-axis). Middle: Accuracy on algorithm design datasets as the average node degree and mixing parameter changes. Bottom: Accuracy on the algorithm design datasets as the network size and mixing parameter changes. Note: FastConsensus failed to converge within four hours in several model conditions (Table A in S1 Appendix), including all networks of size 100K. Each model condition corresponds to one network; hence no error bars are shown.

https://doi.org/10.1371/journal.pcsy.0000069.g002

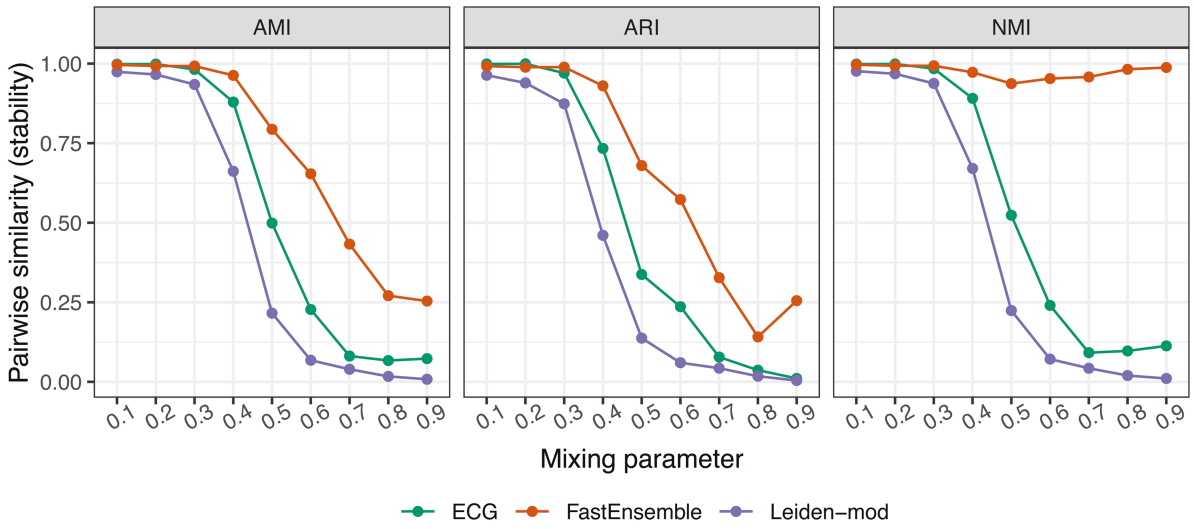

Fig 3. Evaluating stability across multiple stochastic runs on the algorithm design datasets. The methods compared are ECG, FastEnsemble, and Leiden-mod. Each method was run twice on the same input network with different random seeds and in default mode. We report pairwise similarity (measured in terms of AMI, ARI and NMI), that is a measure of stability, between the output clusterings from the two different runs for each method. The datasets used are the algorithm design dataset from Experiment 1 for conditions with the default average degree (10) and network size (10,000); values on the x-axis are the mixing parameter values for generating the synthetic networks.

https://doi.org/10.1371/journal.pcsy.0000069.g003

better stability than Leiden-mod for all three similarity measures and mixing parameters. We also see that ECG and FastEnsemble had the same stability for the smallest mixing parameters (at most 0.3), but FastEnsemble had better stability for all higher mixing parameters.

Experiment 1c: Ablation study. We performed an ablation study where we varied each of the algorithmic parameters of FastEnsemble in turn, keeping the other parameters in their default setting. The parameters we varied are: *np* (number of partitions), *t* (threshold for keeping an edge), the base clustering method (Leiden-mod or Leiden-CPM), and whether to consider the edge weights in the final clustering. Full results are provided in S1 Appendix, Sect B2 and Figs B–E, and summarized here.

We observed that there was sometimes a benefit to having *np*>10 (its default value), but when there was an advantage, it tended to be small. Furthermore, using 50 or 100 partitions increased the runtime. The impact of changing the base method from Leiden-mod to Leiden-CPM depended on the resolution parameter. For large values of the resolution parameter, the impact could be large (and in favor of Leiden-mod), but if the resolution parameter was small (e.g., 0.001), then the two methods provided essentially the same accuracy, with only a few cases where they differed. And, as noted, considering weights in the final clustering (within Leiden-mod as the base method) did not lead to a noticeable change in accuracy on these datasets.

The threshold *t* had a larger impact than these other parameters, but the impact depended on the mixing parameter of the network: using thresholds between 0.8 and 1.0 for networks with small mixing parameters (at most 0.3) and thresholds between 0.7 or 0.8 for networks with larger mixing parameters provided the best accuracy. Since this depends on the "true community structure", which cannot be known in advance, the use of 0.8 is a reasonable default setting for this parameter. One outcome of this analysis is that the Strict Consensus (which is when the threshold is 1.0) is not recommended for real-world analyses, since real-world networks are unlikely to have very small mixing parameters.

Experiment 2: Results of modularity-based clustering on LFR networks based on real-world networks

In this experiment, we evaluated the accuracy and scalability of different consensus clustering methods on the LFR networks from [1] that were generated using modularity-based clusterings of five large real-world networks. These LFR networks range in size from ∼ 35K to ∼ 3.8M nodes and have mixing parameters that range from 0.114 to 0.199 (Table 1).

Among all consensus clustering methods, the only network they completed on within four hours was cit_hepph, the smallest LFR synthetic network with approximately 35K nodes. On this network, all three methods achieved near-perfect AMI, ARI and NMI scores (Fig 4 (left)).

We then extended the runtime limit to 48 hours for the four remaining networks. FastEnsemble and ECG successfully completed on all the networks within this time limit, but FastConsensus completed on only one additional network, wiki_topcats, see Fig 4 (left). All methods had near perfect accuracy for all three criteria on cit_hepph. ECG and FastEnsemble both had high NMI and AMI accuracy on the other networks, with the same relative performance (i.e., both essentially perfect accuracy on cit_hepph and CEN, and a slight advantage to FastEnsemble on the other networks). Both methods were also consistently more accurate than Leiden-mod.

Results for ARI show near-perfect accuracy for all methods on the cit_hepph network, but larger differences between methods on the other networks. FastConsensus completed on only one other network, wiki_topcats, where it had the best accuracy of all three consensus methods. A comparison between ECG and FastEnsemble shows FastEnsemble better on three networks, tied on one, and less accurate on one.

The methods differed in terms of runtime on these networks (Fig 4 (right), Table B in S1 Appendix). While all methods completed very quickly on the smallest network, cit_hepph (under one minute for ECG, FastEnsemble, and Leiden-mod, and 22 minutes for FastConsensus), runtimes on the larger networks were larger. Only Leiden-mod was fast on the larger networks, finishing in under 4 minutes on all networks. FastConsensus was the slowest of all the consensus methods, and managed to complete on only one of these larger networks, requiring 14.5 hours. ECG and FastEnsemble completed on all networks, with runtimes ranging from 6 to 36 hours for ECG and from 7.5 to 28 hours for FastEnsemble).

Experiment 3: Results on CPM-based clustering on LFR networks based on clustered real-world networks

In this experiment, we evaluated the accuracy and runtime of FastEnsemble using Leiden-CPM in comparison to Leiden-CPM on 22 LFR networks from [1]. These networks are based on 5 real-world networks that are clustered using Leiden-CPM for varying resolution parameters, and range in size from ∼ 35K to ∼ 3.8M nodes. For NMI and ARI criteria, FastEnsemble consistently achieved accuracy that is at least as high as Leiden-CPM for all 22 networks and often surpassed it, particularly when used with small resolution values (Fig 5). Results for AMI show nearly the same trends, except that Leiden-CPM was more accurate than FastEnsemble(Leiden-CPM) for the two largest tested resolution parameters, but here too we see large improvements for the smallest resolution parameters.

In this experiment, networks generated using parameters from CPM-clusterings with low resolution values had *low* mixing parameters (Fig F in S1 Appendix). Thus, the results shown here suggest that FastEnsemble used with Leiden-CPM provides an advantage over Leiden-CPM for networks with moderate to large mixing parameters, and usually–but not always–for networks with small mixing parameters.

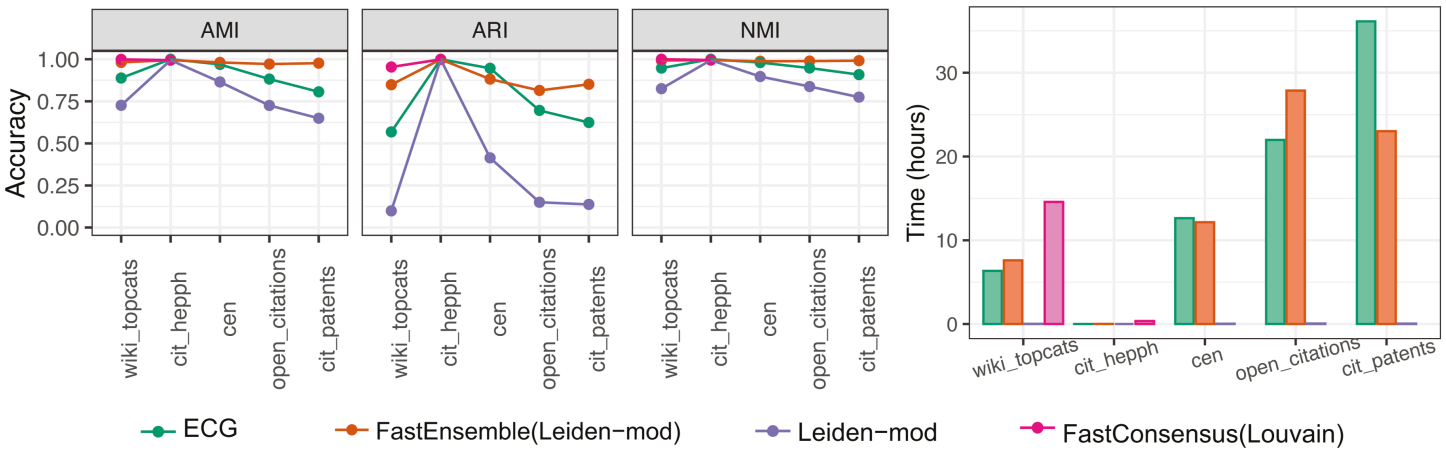

Fig 4. Experiment 2: Evaluating modularity-based consensus clustering pipelines on synthetic networks based on clustered real-world networks. Results are for modularity-based clustering methods on LFR networks from [1], each based on a Leiden-modularity clustering of a real-world network. Left: Accuracy (AMI, ARI, and NMI). Right: Runtime (in hours). FastConsensus failed to converge on three networks (CEN, open_citations, cit_patents) within the allotted 48 hours. The legend applies to all subfigures. Each named network corresponds to one synthetic network; hence no error bars are shown.

https://doi.org/10.1371/journal.pcsy.0000069.g004

The comparison of runtimes shows that FastEnsemble was, on average, much slower compared to Leiden-CPM (Fig 5 (bottom)). Nevertheless, FastEnsemble completed on all the networks in this collection in at most 2.6 hours (see Table C, S1 Appendix), and typically less, thus demonstrating that it can process large networks (up to 3.8 million nodes) in a reasonable time.

Experiment 4: The resolution limit

The resolution limit for modularity was first defined in [32], which proved that in some cases, an optimal modularity-based clustering may not identify what are intuitively the "obvious" communities, especially when those communities are small. As an example, [32] proposed the family of ring-of-cliques networks, which are defined by the clique size k and the number n of cliques. In these networks, the cliques are arranged in a ring and connected to adjacent cliques by a single edge. The study in [32] shows that when $n \geq k(k-1)+2$, the optimal modularity-based clustering will group multiple cliques into a single cluster, rather than returning the obviously preferred clustering where each clique is considered a separate community.

Robustness to resolution limit for modularity optimization: Here we examine whether consensus clustering methods can address this vulnerability of modularity-based clustering from an empirical perspective, using ring-of-cliques networks where each clique is of size $k = 10$ but the number n of cliques is allowed to vary. According to the previous paragraph, when $n \geq 10 \times 9 + 2 = 92$ then an optimal modularity clustering will group two or more of the cliques together. Hence, we examine values of n that are both smaller and larger than $n = 91$ in this experiment. The methods evaluated include Leiden-mod, FastConsensus, ECG, FastEnsemble, and two variants of Strict Consensus, differing in the number of partitions (np) used.

For $n = 90$ clusters, all methods produced clusterings where each clique was returned as a separate community, as desired (Fig 6). However, as the number of clusters increased but not their sizes, then Leiden-mod started merging cliques together, as predicted by the theory from [32]. We also see that the consensus clustering methods (i.e., FastConsensus, ECG,

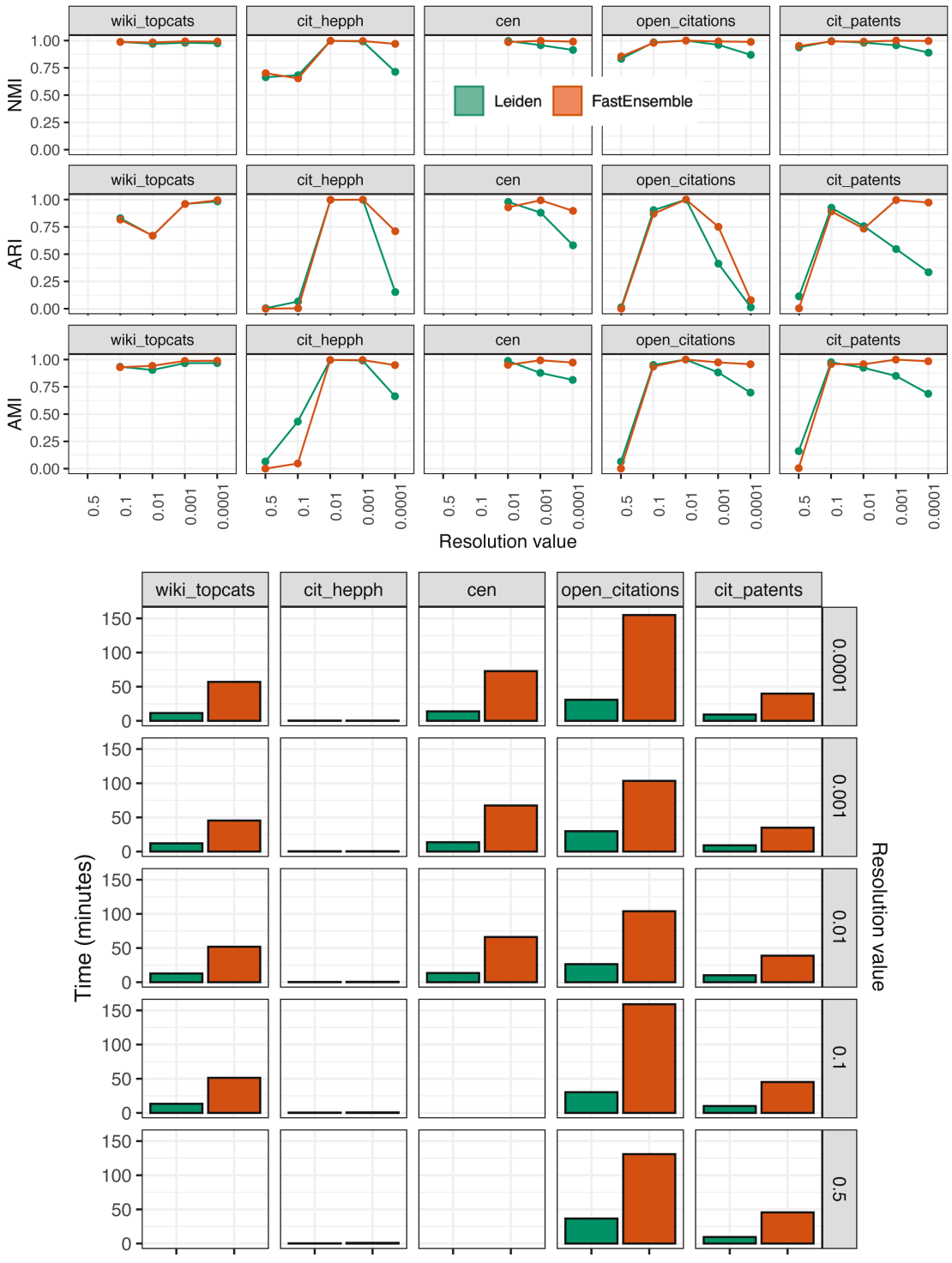

Fig 5. Experiment 3: Comparison of FastEnsemble(Leiden-CPM) and Leiden-CPM on synthetic networks based on clustered real-world networks. The LFR networks are from [1] and are generated from a real-world network clustered using Leiden optimizing CPM for a specific resolution parameter value. The clustering methods studied are Leiden-CPM and FastEnsemble using CPM, each used with the same resolution parameter value as specified for the given LFR network. Top: Accuracy (NMI, ARI, and AMI). Bottom: Runtime (in minutes). The legend applies to all subfigures. Results are not shown for three conditions: LFR graphs with a large fraction of disconnected ground truth clusters (the two CEN networks) or when the LFR software failed to create a network for the provided parameters (the wiki_topcats network). Each combination of named network and resolution parameter corresponds to one synthetic network; hence no error bars are shown.

https://doi.org/10.1371/journal.pcsy.0000069.g005

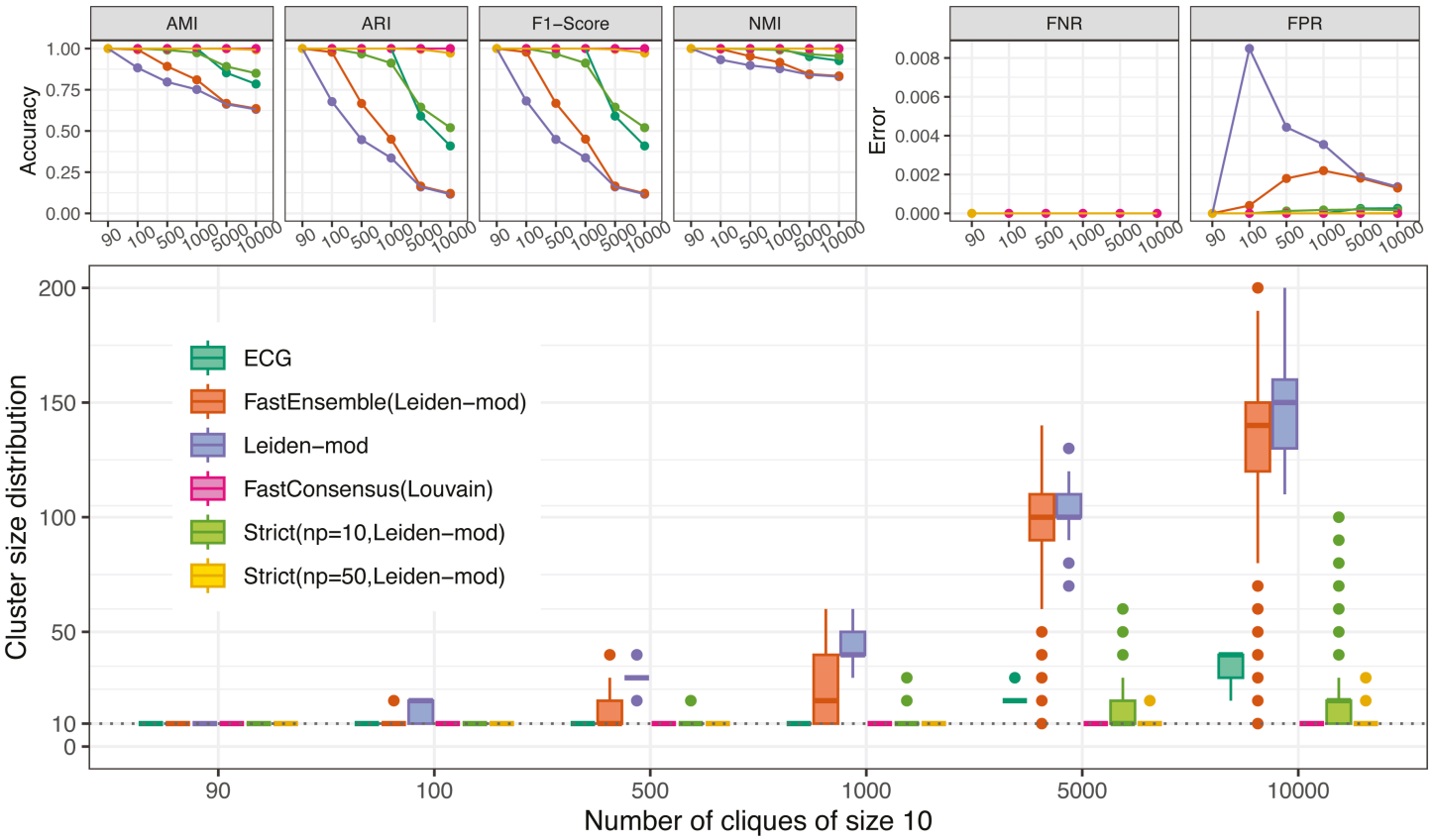

Fig 6. Experiment 4: Accuracy of modularity-based consensus clustering methods on ring-of-cliques networks of varying sizes. Each ring-of-cliques networks connects n cliques of size 10 in a ring. The methods compared are Leiden-mod, ECG, FastConsensus, FastEnsemble, and Strict Consensus (with two numbers np of partitions). Top left: Accuracy (AMI, ARI, NMI, F1-score) as a function of n. Top right: Error metrics (FNR and FPR) as a function of n. Bottom: Cluster size distribution as a function of n (the dotted line indicates the true distribution). The legend applies to all subfigures. There is one synthetic network for each number of cliques; hence no error bars are shown.

https://doi.org/10.1371/journal.pcsy.0000069.g006

FastEnsemble, and Strict Consensus) reduced the tendency to merge cliques into clusters, but some were more beneficial than others.

In particular, for large numbers of cliques, FastConsensus and the Strict Consensus variant with np = 50 had the best accuracy, Leiden-mod had the worst accuracy, and the other methods had intermediate accuracy. ECG was somewhat more accurate than Strict Consensus with $np = 10$ and FastEnsemble had lower accuracy than both, especially for the large numbers of clusters, where it was nearly as poor as Leiden-mod.

Note that all the methods returned zero FNR, indicating that no clique in the ring-of-cliques network was ever split apart (Fig 6 (top right)). On the other hand, the methods differed in terms of FPR, with Leiden-mod having high FPR except for $n = 90$. Again, FastEnsemble was almost as poor as Leiden-mod when there was a large number of cliques, while the other consensus methods had much lower FPR values.

We examined the impact of increasing the number of partitions np within FastEnsemble, to see if an increase would improve accuracy. This experiment, shown in Fig G, S1 Appendix, does show that increasing this value improved accuracy for FastEnsemble. However, even with the improvement, FastEnsemble was still less accurate than the Strict Consensus variants we studied.

Results on tree-of-cliques networks exhibited slightly different trends (Fig 7). As with the ring-of-cliques networks, Leiden-mod had the worst accuracy, followed by FastEnsemble(Leiden-mod), and FastConsensus had the best accuracy. However, on these networks, ECG strictly improved on the two Strict Consensus variants, which is different from what we saw on the ring-of-cliques networks.

Robustness to resolution limit for CPM-based optimization: In contrast to the theory for modularity, [42] established that for every setting of the resolution parameter r, there will be a value N so that every optimal CPM(r) clustering of a ring-of-cliques network with $n \geq N$ cliques of size k will return the individual cliques as clusters. However, our experimental results show that for large enough numbers of cliques of size 10 and small resolution values, Leiden-CPM grouped cliques together into clusters (Fig 8). This vulnerability occurred for all of the small values for the resolution parameter r, but disappeared when $r \geq 0.01$.

Unlike in modularity-based experiments, increasing the number of cliques had little effect on the accuracy of the methods or the cluster size distribution. This suggests that, for CPM-optimization, cliques tend to be co-clustered into groups of the same size, regardless of the overall network size (e.g., Leiden-CPM returns clusters containing approximately 4 to 5 cliques when $r = 0.001$, Fig 8). Using FastEnsemble provided an improvement over Leiden-CPM for all three accuracy measures (AMI, ARI, and NMI) and also improved the cluster size distribution. Finally, applying the Strict Consensus with Leiden-CPM had the best results of all methods, successfully identifying individual cliques as distinct clusters.

Note that for a ring-of-cliques network with n cliques of size k, the mixing parameter with respect to the ground-truth community structure is equal to $\frac{2}{k^2}$ (see Sect C in S1 Appendix for derivation). When $k = 10$, this value becomes $\hat{\mu} = 0.02$, which agrees with Table 1. This means that the mixing parameter of a ring-of-cliques networks only depends on the size of the cliques (and not their count), and except when cliques are extremely small (e.g., at most

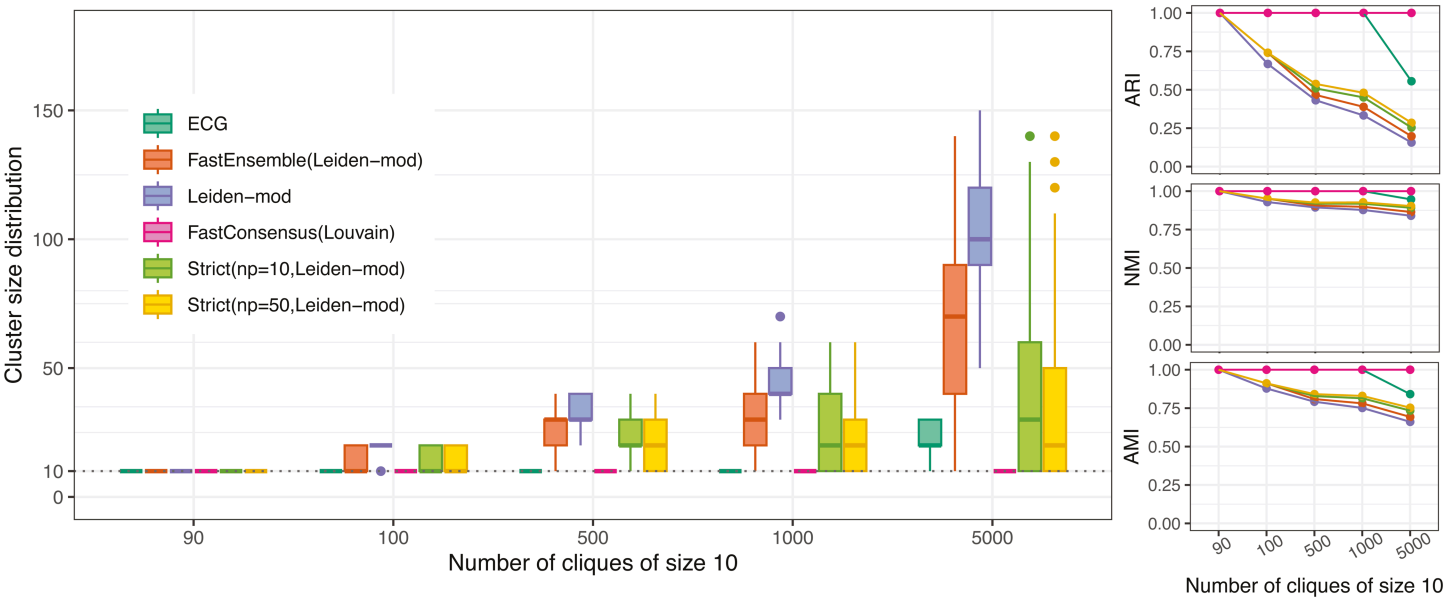

Fig 7. Experiment 4: Evaluating modularity-based clustering methods on tree-of-cliques networks. Each tree-of-cliques network has varying number of cliques of size 10 connected in a tree structure. Left: Cluster size distribution, as a function of the number of cliques (dotted line is the true distribution). Right: Clustering accuracy (AMI, ARI and NMI) as a function of the number of cliques. The legend applies to all subfigures. There is one synthetic network for each number of cliques; hence no error bars are shown.

https://doi.org/10.1371/journal.pcsy.0000069.g007

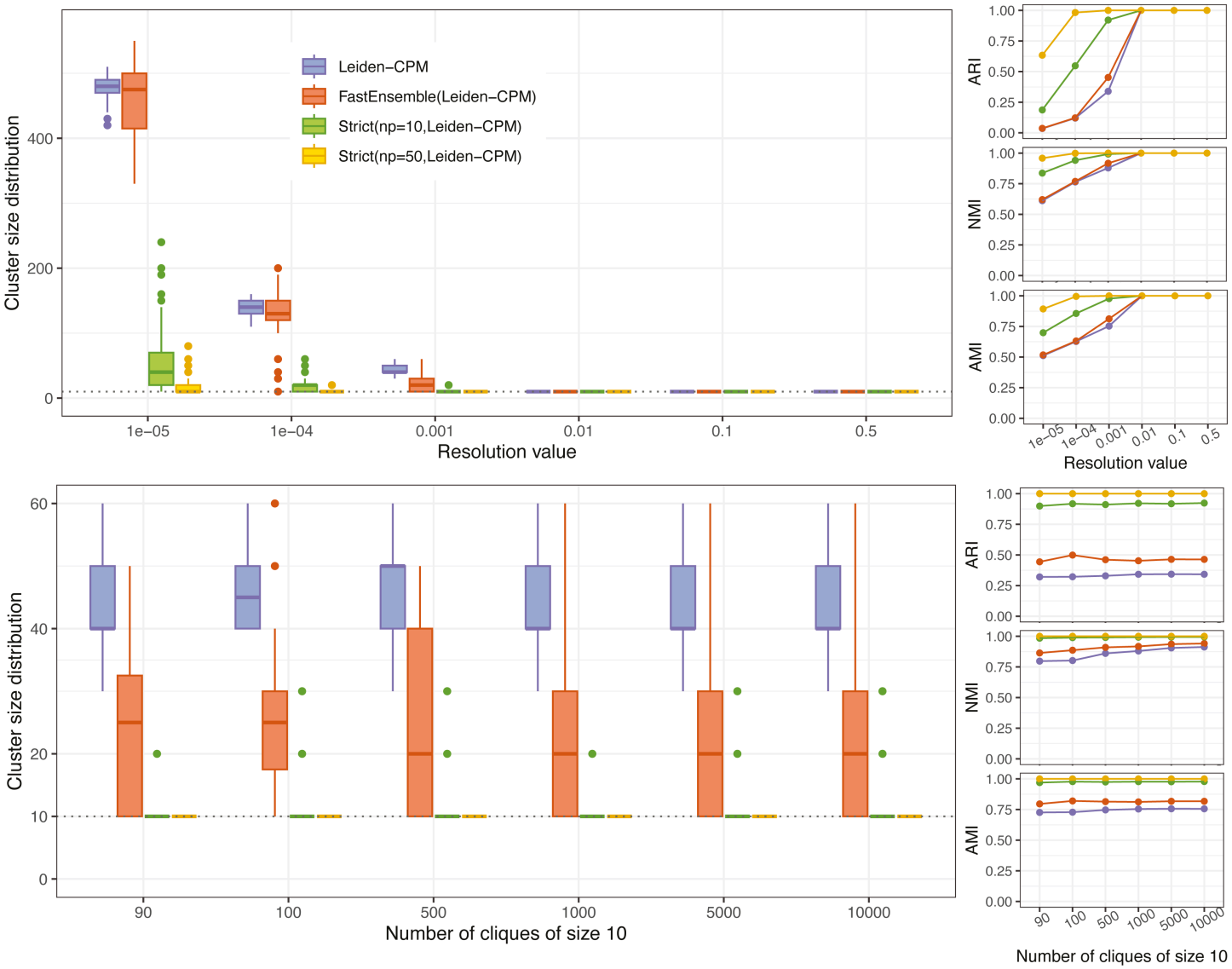

Fig 8. Experiment 4: Evaluating CPM-based clustering methods on ring-of-cliques networks. Results are shown for Leiden-CPM, FastEnsemble with Leiden-CPM, and the Strict Consensus with Leiden-CPM with 10 or 50 partitions (indicated by np). Top left: cluster size distribution as a function of the resolution parameter. Top right: Clustering accuracy as a function of the resolution parameter. Bottom left: Cluster size distribution as a function of the number of cliques (dotted line indicates the true distribution). Bottom right: Cluster accuracy as a function of the number of cliques, using a resolution value of $r = 0.001$. The legend applies to all subfigures. There is one synthetic network for each resolution parameter value (top) or number of cliques (bottom); hence no error bars are shown.

https://doi.org/10.1371/journal.pcsy.0000069.g008

four nodes), the mixing parameter is very low. The lower accuracy of FastEnsemble compared to ECG and FastConsensus in this context aligns with the findings on the algorithm design dataset.

Experiment 5: Clustering networks that have only partial community structure

While Erdős-Rényi graphs may exhibit regions that appear to be valid communities based on metrics such as modularity scores, we follow the discussion in [43] and treat Erdős-Rényi

graphs as lacking any true community structure. Thus, we do not consider any cluster of size greater than 1 to be valid in an Erdős-Rényi graph. We used these graphs to assess the extent to which consensus clustering pipelines can correctly reject spurious community structures by producing no or very few non-singleton clusters. Additionally, we constructed hybrid networks that combine Erdős-Rényi graphs with LFR networks and ring-of-cliques networks to examine whether clustering methods can correctly restrict detected communities to subnetworks with well-defined community structures. To evaluate these aspects, we analyzed both the cluster size distribution and overall clustering accuracy.

On Erdős-Rényi graphs, the cluster size distribution and accuracy for each method was very impacted by the density p (Fig 9 (top)). In particular, while cluster sizes tended to be small for the smallest tested value for p, ECG and Leiden-mod produced fairly large clusters even at relatively small values for p, and so did FastConsensus at slightly larger values. In contrast, the clusters produced by FastEnsemble and the two Strict Consensus variants decreased in size as the density p increases. The clustering AMI, NMI and ARI accuracy results also reflect these trends. For ARI and AMI, all methods other than the two Strict Consensus variants had very poor accuracy at all values for p, but the two Strict Consensus variants improved as p increases and attained high accuracy for the larger values for p. NMI results show all methods had fairly high accuracy for the smallest tested value for p, but Leiden-Mod, ECG, and FastConsensus degraded as p increases, while FastEnsemble and the two Strict Consensus variants improved as p increases.

We observe somewhat different trends in networks that combine Erdős-Rényi graphs with LFR networks (Fig 9 (middle)). The LFR subnetwork has 14 ground-truth communities with sizes that range from 45 to 96 (Sect B.1.4 in S1 Appendix) and its mixing parameter is 0.14. Therefore, the correct community structure should have half the nodes in singleton clusters and the other half in 14 clusters that do not exceed 100 nodes. The cluster size distributions seen in Fig 9 (middle) seem reasonably accurate for all methods for the very lowest density values for the Erdős-Rényi graphs, and then accuracy decreased. Specifically, with the exception of FastConsensus, for the middle density values, all methods produced large clusters, and some even produced clusters of size 1000. Upon inspection, the clusters of size 1000 were verified to be the Erdős-Rényi graphs. At the highest density values, the cluster size distribution for most methods dropped closer to the true values, but Leiden-mod continued to produce very large clusters, including one of size 1000. An examination of AMI, NMI and ARI accuracy shows interesting trends that reflect the cluster size distribution accuracy. For ARI and AMI, the method with consistent but poor accuracy across all density values was FastConsensus. Leiden-mod started with high accuracy and then dropped to a very low accuracy as the density increases, and never regained accuracy. ECG was similar to Leiden-mod in starting at high accuracy and then decreasing to low accuracy, but it regained some accuracy as the density increases. FastEnsemble and the two Strict Consensus variants showed a surprising trend of starting high, dropping down to a low value, and then going back to a high value, though the Strict Consensus variants returned to the high value at lower density values than FastEnsemble. Results for NMI were similar as for ARI, but the accuracy scores were higher.

We also examined Erdős-Rényi graphs combined with ring-of-cliques networks, which have mixing parameter 0.02 (Fig 9 (bottom)). On these networks, the true cluster size distribution has half the nodes in clusters of size 10 and the other half in singleton clusters. There were similar trends for cluster size distributions, with good accuracy at the lowest density and then all methods (other than FastConsensus) grouping all the nodes in the Erdős-Rényi graph into one cluster for the intermediate density values.

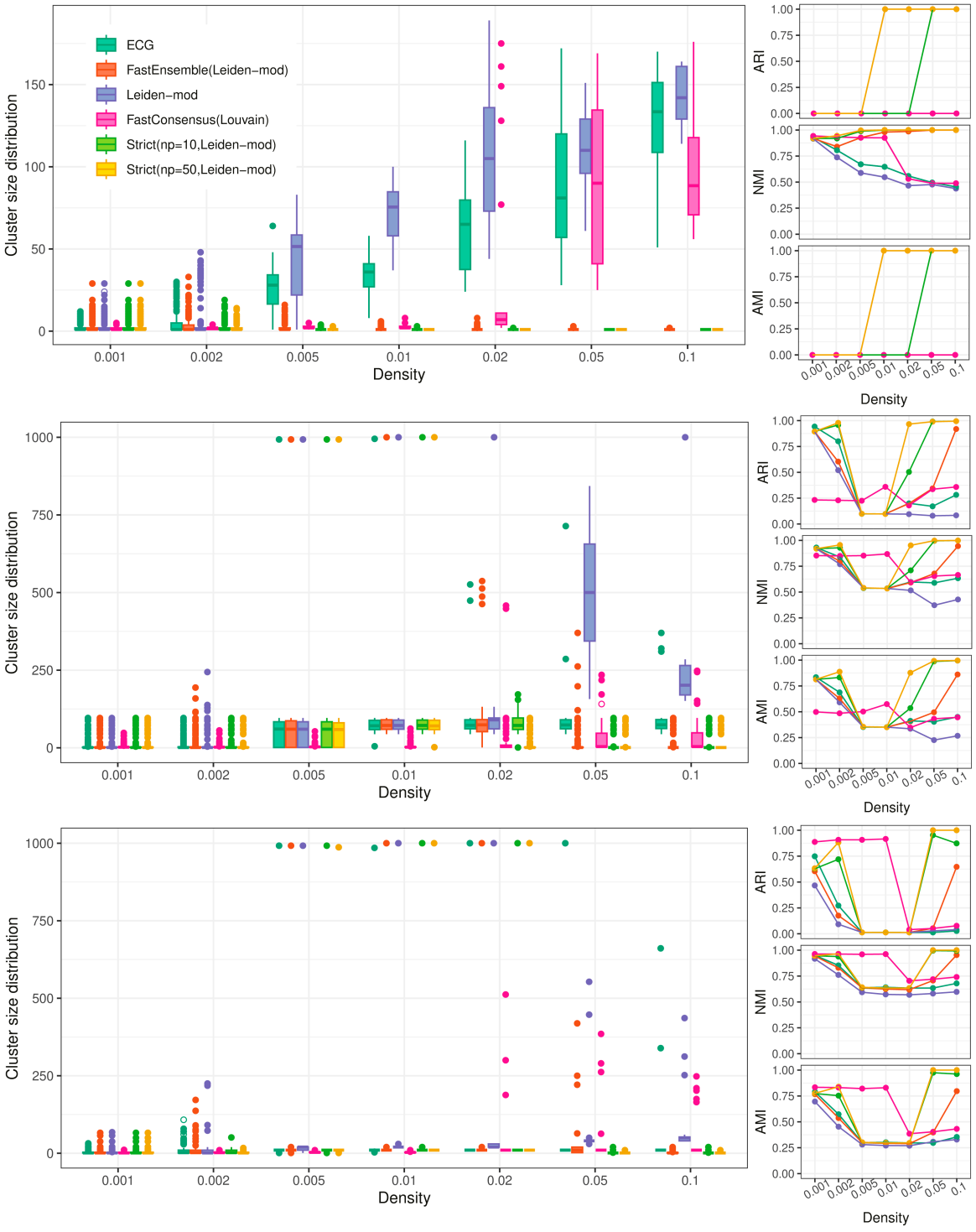

Fig 9. Experiment 5: Clustering networks that are only partially clusterable. Accuracy and cluster size distributions of modularity-based consensus clustering on networks with clusterable and unclusterable components. The unclusterable portion is created using Erdős-Rényi graphs with various densities, and the clusterable portion includes strong community structure created using LFR graphs or ring-of-cliques. Each row (top, middle, bottom) shows the cluster size distribution on the left and the clustering accuracy on the right (AMI, ARI and NMI), as a function of the density of the Erdős-Rényi graph. Top: Erdős-Rényi graphs with 1000 nodes and various densities. Middle: Erdős-Rényi graph of size 1000 attached to an LFR graph of size 1000 with 14 communities (2000 nodes in total), with sizes ranging from 45 to 96. Bottom: Erdős-Rényi graph attached to a ring of 10-cliques of size 1000 (2000 total nodes). The legend applies to all subfigures. There is one synthetic network for each density value; hence no error bars are shown.

https://doi.org/10.1371/journal.pcsy.0000069.g009

On these networks and for all accuracy criteria, the best accuracy depended on the density for the Erdős-Réyni graphs. For density values below 0.02, the best accuracy was obtained by FastConsensus, but for larger density values the best accuracy was obtained by the two Strict Consensus variants. For most density values, FastEnsemble and ECG showed similar accuracy (only better than Leiden-mod), but FastEnsemble was more accurate than ECG at the two highest tested density values.

Overall, for this experiment the relative accuracy between methods depended very much on the density of the Erdős-Rényi graph as well as the structure of the clusterable subnetwork. FastConsensus did poorly in one setting (when the Erdős-Rényi graph was paired with LFR networks) and well in the other (when it was paired with the ring-of-cliques network). FastEnsemble did not have very good accuracy in either setting for middle density values, but did well at the highest density values. The two Strict Consensus variants, however, did well at the highest density values, and were generally more accurate than the other methods. Nevertheless, no method did well at all density values, and no method dominated the others.

Experiment 6: Clustering real-world networks

We evaluated ECG and FastEnsemble on two real-world networks, DBLP and Amazon Products, from the SNAP [36] collection. We also ran FastConsensus on these networks, but it failed to converge within 48 hours, and so we restrict our discussion to ECG and FastEnsemble.

These networks come with ground truth communities that have been ranked by [38] according to six different cluster quality metrics. We used the top-scoring 5000 communities for each network. Since these clusters are overlapping, we made them pairwise-disjoint before using them for evaluation purposes by removing all nodes from the clusters that had membership in two or more of these top 5000 communities. Each clustering method was applied to the entire network, and then evaluated only with respect to these modified clusters, using AMI, ARI, and NMI criteria. To perform this evaluation for a given clustering, the set of clusters induced on the nodes in these clusters was computed and then compared to the given clustering. Results on these networks are shown in Fig 10.

Note that AMI, ARI, and NMI accuracies for all clusterings are higher on the Amazon Products network than the DBLP network, indicating that the Amazon Products network is easier to cluster than the DBLP network.

Examining the modularity-based clusterings, ECG was a bit more accurate for all three criteria than FastEnsemble on the Amazon Products network (and both were more accurate than Leiden-mod). Interestingly, the trends are different on the DBLP network: for all three criteria, ECG was a bit less accurate than FastEnsemble on the DBLP network, and was even less accurate than Leiden-mod for ARI accuracy. Thus the clustering methods perform differently on these two networks.

CPM-based clusterings are restricted to Leiden-CPM and FastEnsemble used with Leiden-CPM, as ECG does not work with Leiden-CPM. Here we see a different outcome: for both networks, for every fixed resolution parameter value, and for each of the three criteria, FastEnsemble and Leiden-CPM had nearly identical accuracy, indicating that FastEnsemble did not change the clustering very much. The only case where there was any visible change in accuracy is for the smallest resolution parameter (0.0001), where FastEnsemble was slightly more accurate than Leiden-CPM on DBLP for AMI and NMI, and very slightly less accurate for ARI. Finally, the choice of resolution parameter affected ARI and NMI accuracy, with

Fig 10. AMI, ARI and NMI accuracy on two real-world networks. We show accuracy results for modularity-based clusterings (left) and CPM-based clusterings (right) on two real-world networks from the SNAP repository, Amazon Products and DBLP. We show AMI (top), ARI (middle), and NMI (bottom) accuracy of the computed clustering with respect to a pairwise-disjoint version of the top 5000 ground truth clusters based on cluster quality, according to [38]. Each bar indicates one analysis, and so error bars are not shown.

https://doi.org/10.1371/journal.pcsy.0000069.g010

smaller resolution values better for ARI and larger resolution values better for NMI; changes in the resolution value did not seem to particularly affect AMI accuracy.

Discussion

We used both simulated and real-world networks to explore three consensus clustering methods, ECG, FastConsensus, and FastEnsemble. For nearly all conditions we explored, each of the methods improved on the accuracy of the base clustering method (i.e., modularity optimization for all three methods, and also Leiden-CPM for FastEnsemble). The results for ECG and FastConsensus are consistent with prior studies [16,26] and are expected. We saw that FastConsensus was the slowest of the three methods, and was unable to complete analyses within 48 hours on several large networks, while the other two methods were able to complete on all the networks we explored within 48 hours. Of significant importance, no consensus clustering method was consistently more accurate than the others, thus raising the question of how to choose between methods.

Comparing FastConsensus, ECG, and FastEnsemble

We begin with a discussion of FastConsensus. Because FastConsensus was unable to complete on the largest networks, our discussion is limited to its performance on a subset of the networks we explored. There were two experiments where FastConsensus had the best accuracy: ring-of-cliques and tree-of-cliques networks in Experiment 4 and some of the networks formed by combining Erdős-Réyni networks with LFR or ring-of-cliques networks (Experiment 5). In the other experiments, FastConsensus was rarely the most accurate tested method in our experiments. Furthermore, when it was the most accurate tested method, it tied for most accurate with one or both of ECG and FastEnsemble. Given this, and considering that FastConsensus was unable to scale to large networks, we restrict the rest of the discussion to a comparison between FastEnsemble and ECG.

A comparison between ECG and FastEnsemble can be made on all the networks we explored. Experiment 1 examined the algorithm design datasets, and explored how average degree, mixing parameter, and network size impacted the absolute and relative accuracy of the two methods. On the default algorithm design datasets (10,000 nodes with average degree 10), ECG was typically more accurate than FastEnsemble when the mixing parameter was low (at most 0.3) and FastEnsemble was more accurate when the mixing parameter was higher.

Having ECG be more accurate than FastEnsemble when the mixing parameter is small enough, and otherwise FastEnsemble being more accurate, occurs in many of the other experiments, but is not universally found. For example, in Experiment 1, when we varied the network size or average degree, ECG stopped being more accurate than FastEnsemble, although the mixing parameter remained low. In addition, all the networks in Experiment 2 have low mixing parameters, and yet ECG was not reliably more accurate than FastEnsemble on these networks. Thus, there are networks with low mixing parameters where ECG is not as accurate as FastEnsemble. The converse also occurs: networks with moderate mixing parameters where ECG is more accurate than FastEnsemble. For example, ECG was more accurate than FastEnsemble on Experiment 5 networks produced by combining Erdős-Réyni graphs with LFR graphs or ring-of-cliques graphs, which have moderately large mixing parameters (at least 0.4). We also saw one real-world network (Amazon Products) where ECG was more accurate than FastEnsemble, and this network has a very large mixing parameter (0.845).

To understand the trends on these networks, we need to consider how we compute mixing parameters. We begin with an examination of the mixing parameter calculation and per-node values (see Figs I–M in S1 Appendix). To calculate the mixing parameter of a network, the mixing parameters for each of the nodes are averaged. When all the nodes in the network are in communities, this approach makes sense. However, consider instead networks that

either have no valid communities (e.g., Erdős-Réyni graphs) or where much or all of the network is not in a valid community (the case where we combine an Erdős-Réyni graph with an LFR graph or a ring-of-cliques in Experiment 5 and the two real-world networks studied in Experiment 6).

Focusing on the per-node mixing parameters for networks formed by combining Erdős-Rényi graphs and LFR or ring-of-cliques, note that the nodes in the Erdős-Rényi network that are not isolated nodes each have mixing parameter 1.0, while the isolated nodes have mixing parameter 0.0. The nodes in the clusterable subnetworks that are paired with the Erdős-Rényi networks mostly have very low mixing parameters, with average values less than 0.15. As a result, the *average* mixing parameter for the networks that are formed by pairing an Erdős-Rényi network with an LFR network or a ring-of-cliques network are in the moderate to high range of 0.4 to 0.57, but the distribution of mixing parameters is *bimodal*: between 50% and 81% are small and the remaining ones are all maximally large at 1.0 (see Fig L in S1 Appendix).

This is a very different kind of distribution than we have for the networks studied in Experiments 1–4 (see Figs H–K in S1 Appendix), which are for networks that have all or nearly all the nodes within communities. For the networks in Experiments 1–4, the per-node mixing parameters are typically concentrated around the mean with low variance, and even if they have wide variance, they are nevertheless not bimodal.

We see a similar pattern for the mixing parameter values in Experiment 6, which addressed recovery of the top 5000 clusters in two real-world networks. The overall mixing parameters for these networks are large (0.845 for Amazon Products and 0.991 for DBLP) for both networks (Table 1). However, most of the nodes in these networks are not in these modifications to the top 5000 clusters; each such node is therefore considered unclustered (equivalently, in a cluster of size 1). However, the clustered subnetwork for each of the real-world networks have much smaller mixing parameters: 0.381 for DBLP and 0.085 for Amazon Products.

In other words, for both the real-world and synthetic networks in this study, the relative accuracy of ECG and FastEnsemble to some extent tracks the average mixing parameter of the *clustered subnetwork*, rather than the average mixing parameter of the entire network: networks with only partial community structure that have bimodal distributions for the per-node mixing parameter *may be* better clustered using ECG if the clustered subnetwork has a low mixing parameter and better clustered using FastEnsemble when the clustered subnetwork has a moderate or high average mixing parameter. However, this is just a hypothesis, and future work is needed to understand the conditions under which ECG or FastEnsemble should be preferred.

Given the algorithmic similarities between ECG and FastEnsemble, the variations in accuracy under certain conditions—sometimes favoring ECG and other times FastEnsemble are particularly interesting. One possible explanation is that FastEnsemble, by default, employs Leiden-mod, whereas ECG uses Louvain. However, as shown in Figs M–O in S1 Appendix, there is no difference in accuracy between FastEnsemble used with Louvain and FastEnsemble used with Leiden-mod for the Algorithm Design datasets, the ring-of-cliques networks, and the tree-of-cliques networks. This suggests that the difference in accuracy between FastEnsemble and ECG is not a result of the choice between Leiden-mod and Louvain. A more notable distinction is ECG's reliance on 2-core-based edge weighting, which FastEnsemble does not require. Further research is needed to better understand these differences and their impact on the relative performance of these methods.

Computational performance

By design, the three consensus methods we explored (ECG, FastEnsemble, and FastConsensus) are slower than their base methods. Therefore, the focus here is on the relative computational performance of the three methods, as well as scalability to large networks.

The major observation is that FastConsensus was the most computationally expensive method: it failed to converge in 14 out of the 45 model conditions in Experiment 1 (Table A in S1 Appendix) within the allotted four-hour time limit, including on all networks with 100K nodes. FastConsensus also failed to complete within the allowed 48 hours on three of the five Experiment 2 networks (i.e., the LFR networks based on clustered real-world networks from [1]) and the two real-world networks from Experiment 6. In contrast, ECG and FastEnsemble were able to complete on all the tested networks when given 48 hours. In addition, on those networks where FastConsensus was able to converge, it was much slower than both ECG and FastEnsemble, especially on the networks with more than 1,000,000 nodes.

A comparison between ECG and FastEnsemble(Leiden-mod) shows that neither is consistently faster than the other. Furthermore, both completed within the allowed time on every network we explored. On the five large LFR networks studied in Experiment 2 (taken from [1]) that range from $\sim$ 35 thousand to $\sim$ 3.8 million nodes, FastEnsemble is slower on three networks and faster on two (Table B, S1 Appendix). However, a closer analysis shows that ECG required up to 36 hours on these networks, while FastEnsemble finished within 28 hours on each of the networks. In addition, the two methods are reasonably close in runtime on four of the five networks, and only far apart on one: the LFR cit_patents network, which has $\sim$ 3.8 million nodes and is the largest of the networks we explored. On that network, ECG uses $\sim$ 36 hours while FastEnsemble uses $\sim$ 23 hours. Thus, although neither dominates the other, these preliminary results suggest that possibly FastEnsemble may have an advantage for runtime.

Finally, the runtime of FastEnsemble used with Leiden-CPM is worth examining, even though a comparison cannot be made to either ECG or FastConsensus (which can only be used with modularity optimization). On the 22 LFR networks from Experiment 3 (which are based on Leiden-CPM clusterings of 5 real-world networks, and range up to $\sim$ 3.8 million nodes), FastEnsemble finishes in under 2.6 hours on every network (Table C, S1 Appendix). This reduced runtime, compared to when FastEnsemble was used with Leiden-mod, is likely due to Leiden-CPM being faster on these networks than Leiden-mod.

Strict consensus

The Strict Consensus is FastEnsemble with the threshold t set to 1.0; we studied two versions that differ only in how many partitions are used. Because $t = 1$ in the Strict Consensus, unless a pair of nodes are co-clustered in every partition, the weight on the edge will be 0; thus, the Strict Consensus variants are designed to be very conservative. The Strict Consensus was explored in two experiments: Experiment 4, which examined networks that presented a challenge for the resolution limit, and Experiment 5, which examined networks that had only partial community structure. In these experiments, the Strict Consensus had very good accuracy, including the best accuracy of all methods on the Experiment 4 networks and on the Erdős-Réyni networks from Experiment 5. This is not surprising. We also observed that the Strict Consensus was among the better methods for the combination of Erdős-Réyni graphs with other subnetworks in Experiment 5.

Nevertheless, the ablation study, performed in Experiment 1, revealed that the Strict Consensus is only beneficial (compared to the base method) when the mixing parameter is at most 0.3. For networks with higher mixing parameters, it can result in worsened accuracy. Taken together, these results suggest that the Strict Consensus is capable of producing highly

accurate clusterings when the purpose is to avoid false discovery of communities, and even then only when the communities that are sought have low mixing parameters.

Conclusions

This study introduced FastEnsemble, a new consensus clustering method that can be used with Louvain or Leiden for optimizing modularity or with Leiden for optimizing under the constant Potts model. Our study using a wide range of synthetic networks showed that FastEnsemble generally matches or improves the accuracy of its base method. We also established that FastEnsemble is fast enough to use on large networks, including some with nearly 3.8 million nodes.

The comparison between FastEnsemble and two established consensus methods, ECG and FastConsensus, shows that only FastEnsemble and ECG are able to run on large networks within reasonable timeframes (e.g., 48 hours). The relative accuracy of the three methods depends on the network and its community structure, so that no method outperforms the others under all conditions. Understanding the conditions under which FastEnsemble is more reliable than ECG, and vice-versa, remains open, and requires future study.

This study suggests other directions for future work. The main focus of this study was using consensus methods for modularity optimization, but our study also explored (in Experiment 3) using FastEnsemble with Leiden-CPM. Given its speed and good accuracy in that experiment, additional investigation into the potential for this approach is merited. FastEnsemble also needs to be compared to new consensus clustering methods [10,23], with respect to both accuracy and computational performance.

We examined these consensus clustering methods for the problem of disjoint clustering, but other types of clustering outputs (i.e., hierarchical clusterings or overlapping clusterings) are very relevant to real-world network analysis, and should be considered. While both ECG and FastEnsemble are fast enough to use on relatively large networks with millions of nodes, techniques for speeding up FastEnsemble and other consensus methods could be explored, such as finding the 2-core of the network before applying the consensus method [44].

The difference in trends for clustering networks that are entirely covered by communities and those that are only partially covered by communities indicates the need to explore accuracy on a wider range of synthetic networks. This difference could be particularly relevant to real-world networks, as some studies have argued that real-world networks are not entirely covered by communities [1,45,46]. Given the difficulty in knowing the ground truth community structure in real-world networks, synthetic network generators that are designed to produce networks with only partial community structure are needed. ABCD+o [45], RECCS [47], and EC-SBM [48] are network simulators that explicitly allow for outliers (i.e., nodes that are not in any non-singleton community) and aim to produce realistic simulated networks. Thus, future work should examine clustering accuracy using these simulators.

To ensure a fair comparison with other consensus methods, our study focused solely on a version of FastEnsemble that employs multiple runs of a *single* algorithm. We did not investigate the advanced version that enables the combination of different clustering algorithms and multi-resolution ensemble clustering. Future research should explore this functionality to determine the conditions under which combining different algorithms (e.g., Leiden-mod and Leiden-CPM) yields superior performance compared to multiple runs of each algorithm individually. Additionally, further studies should examine a broader range of networks and algorithmic combinations to gain deeper insights into these trends and identify potential variants of FastEnsemble that may achieve higher accuracy than the current default version while still

providing good computational performance and scalability. Finally, extending this approach to overlapping clustering may be simple: it only requires using clustering methods that produce overlapping clusters instead of methods that produce disjoint clusters and in computing the edge-weights appropriately (e.g., using the fraction of the input clusterings that put the two nodes together in at least one cluster). However, extending this to hierarchical clustering is more complicated, and is left to future work.

Supporting information

S1 Appendix. Supplementary materials document. This PDF document contains additional details about the data generation, commands for running software, and additional results provided in 3 supplementary tables and 15 supplementary figures.
(PDF)

Author contributions

Conceptualization: Yasamin Tabatabaee, Eleanor Wedell, Tandy Warnow.

Data curation: Yasamin Tabatabaee.

Formal analysis: Yasamin Tabatabaee, Eleanor Wedell, Minhyuk Park, Tandy Warnow.

Funding acquisition: Tandy Warnow.

Investigation: Yasamin Tabatabaee, Eleanor Wedell, Minhyuk Park, Tandy Warnow.

Methodology: Yasamin Tabatabaee, Eleanor Wedell, Tandy Warnow.

Project administration: Tandy Warnow.

Resources: Tandy Warnow.

Software: Yasamin Tabatabaee, Eleanor Wedell, Minhyuk Park.

Supervision: Tandy Warnow.

Validation: Yasamin Tabatabaee.

Visualization: Yasamin Tabatabaee.

Writing – original draft: Yasamin Tabatabaee.

Writing – review & editing: Tandy Warnow.

References

1. Park M, Tabatabaee Y, Ramavarapu V, Liu B, Pailodi VK, Ramachandran R, et al. Well-connectedness and community detection. PLOS Complex Syst. 2024;1(3):e0000009. https://doi.org/10.1371/journal.pcsy.0000009
2. Yang Z, Algesheimer R, Tessone CJ. A comparative analysis of community detection algorithms on artificial networks. Sci Rep. 2016;6:30750. https://doi.org/10.1038/srep30750 PMID: 27476470
3. Fortunato S. Community detection in graphs. Physics Reports. 2010;486(3–5):75–174. https://doi.org/10.1016/j.physrep.2009.11.002
4. Jin D, Yu Z, Jiao P, Pan S, He D, Wu J, et al. A survey of community detection approaches: from statistical modeling to deep learning. IEEE Trans Knowl Data Eng. 2021;:1–1. https://doi.org/10.1109/tkde.2021.3104155
5. Liu Y, Xia J, Zhou S, Yang X, Liang K, Fan C, et al. A survey of deep graph clustering: taxonomy, challenge, application, and open resource. arXiv preprint 2022. https://doi.org/10.48550/arXiv.2211.12875
6. Ding L, Li C, Jin D, Ding S. Survey of spectral clustering based on graph theory. Pattern Recognition. 2024;151:110366. https://doi.org/10.1016/j.patcog.2024.110366

7. Li J, Lai S, Shuai Z, Tan Y, Jia Y, Yu M, et al. A comprehensive review of community detection in graphs. Neurocomputing. 2024;600:128169. https://doi.org/10.1016/j.neucom.2024.128169
8. Berahmand K, Saberi-Movahed F, Sheikhpour R, Li Y, Jalili M. A comprehensive survey on spectral clustering with graph structure learning. arXiv preprint 2025. https://doi.org/10.48550/arXiv.2501.13597
9. Lancichinetti A, Fortunato S. Consensus clustering in complex networks. Sci Rep. 2012;2:336. https://doi.org/10.1038/srep00336 PMID: 22468223
10. Morea F, De Stefano D. Enhancing stability and assessing uncertainty in community detection through a consensus-based approach. arXiv preprint 2024.https://doi.org/10.48550/arXiv.2408.02959
11. Traag VA, Waltman L, van Eck NJ. From Louvain to Leiden: guaranteeing well-connected communities. Sci Rep. 2019;9(1):5233. https://doi.org/10.1038/s41598-019-41695-z PMID: 30914743
12. Boyack KW, Klavans R. An improved practical approach to forecasting exceptional growth in research. Quantitative Science Studies. 2022;3(3):672–93. https://doi.org/10.1162/qss_a_00202
13. Newman MEJ, Girvan M. Finding and evaluating community structure in networks. Phys Rev E Stat Nonlin Soft Matter Phys. 2004;69(2 Pt 2):026113. https://doi.org/10.1103/PhysRevE.69.026113 PMID: 14995526
14. Ronhovde P, Nussinov Z. Local resolution-limit-free Potts model for community detection. Phys Rev E Stat Nonlin Soft Matter Phys. 2010;81(4 Pt 2):046114. https://doi.org/10.1103/PhysRevE.81.046114 PMID: 20481793
15. Strehl A, Ghosh J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research. 2002;3(Dec):583–617.
16. Tandon A, Albeshri A, Thayananthan V, Alhalabi W, Fortunato S. Fast consensus clustering in complex networks. Phys Rev E. 2019;99(4–1):042301. https://doi.org/10.1103/PhysRevE.99.042301 PMID: 31108682
17. Jeub LGS, Sporns O, Fortunato S. Multiresolution consensus clustering in networks. Sci Rep. 2018;8(1):3259. https://doi.org/10.1038/s41598-018-21352-7 PMID: 29459635
18. Goder A, Filkov V. Consensus clustering algorithms: comparison and refinement. 2008 Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments (ALENEX). Society for Industrial and Applied Mathematics; 2008. p. 109–17. https://doi.org/10.1137/1.9781611972887.11
19. Li T, Ding C. Weighted consensus clustering. In: Proceedings of the 2008 SIAM International Conference on Data Mining. 2008. p. 798–809.
20. Lock EF, Dunson DB. Bayesian consensus clustering. Bioinformatics. 2013;29(20):2610–6. https://doi.org/10.1093/bioinformatics/btt425 PMID: 23990412
21. van Dongen S. Fast multi-resolution consensus clustering. Cold Spring Harbor Laboratory; 2022. https://doi.org/10.1101/2022.10.09.511493
22. Zhang P, Moore C. Scalable detection of statistically significant communities and hierarchies, using message passing for modularity. Proc Natl Acad Sci U S A. 2014;111(51):18144–9. https://doi.org/10.1073/pnas.1409770111 PMID: 25489096
23. Hussain MT, Halappanavar M, Chatterjee S, Radicchi F, Fortunato S, Azad A. Parallel median consensus clustering in complex networks. Sci Rep. 2025;15(1):3788. https://doi.org/10.1038/s41598-025-87479-6 PMID: 39885235
24. De Meo P, Ferrara E, Fiumara G, Provetti A. Generalized Louvain method for community detection in large networks. In: 2011 11th International Conference on Intelligent Systems Design and Applications. 2011. p. 88–93. https://doi.org/10.1109/isda.2011.6121636
25. Que X, Checconi F, Petrini F, Gunnels JA. Scalable Community Detection with the Louvain Algorithm. In: 2015 IEEE International Parallel and Distributed Processing Symposium. 2015. p. 28–37. https://doi.org/10.1109/ipdps.2015.59
26. Poulin V, Théberge F. Ensemble clustering for graphs: comparisons and applications. Applied Network Science. 2019;4(1):51.
27. Tabatabaee Y, Wedell E, Park M, Warnow T. Fast ensemble: a new scalable ensemble clustering method. In: International Conference on Complex Networks and their Applications. 2024. p. 57–70.
28. Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs for testing community detection algorithms. Phys Rev E Stat Nonlin Soft Matter Phys. 2008;78(4 Pt 2):046110. https://doi.org/10.1103/PhysRevE.78.046110 PMID: 18999496
29. Tabatabaee Y. Real network emulation using LFR graphs. 2023. https://github.com/ytabatabaee/emulate-real-nets
30. Théberge F. Ensemble clustering for graphs. 2019. https://github.com/ftheberge/Ensemble-Clustering-for-Graphs

31. Théberge F. Graph Partition and Measures; 2023. https://pypi.org/project/partition-igraph/
32. Fortunato S, Barthélemy M. Resolution limit in community detection. Proc Natl Acad Sci U S A. 2007;104(1):36–41. https://doi.org/10.1073/pnas.0605965104 PMID: 17190818
33. Sahu S, Kothapalli K, Banerjee DS. Fast Leiden algorithm for community detection in shared memory setting. In: Proceedings of the 53rd International Conference on Parallel Processing. 2024. p. 11–20. https://doi.org/10.1145/3673038.3673146
34. Newman MEJ. Mixing patterns in networks. Phys Rev E Stat Nonlin Soft Matter Phys. 2003;67(2 Pt 2):026126. https://doi.org/10.1103/PhysRevE.67.026126 PMID: 12636767
35. Jiang H, Liu Z, Liu C, Su Y, Zhang X. Community detection in complex networks with an ambiguous structure using central node based link prediction. Knowledge-Based Systems. 2020;195:105626. https://doi.org/10.1016/j.knosys.2020.105626
36. Leskovec J, Krevl A. SNAP datasets: Stanford large network dataset collection. 2014. http://snap.stanford.edu/data
37. Erdős P, Rényi A. On random graphs. Publicationes Mathematicae. 1959;6(3–4):290–7.
38. Yang J, Leskovec J. Defining and evaluating network communities based on ground-truth. In: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics. 2012. p. 1–8. https://doi.org/10.1145/2350190.2350193
39. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in python. The Journal of Machine Learning Research. 2011;12:2825–30.
40. Peixoto TP. The graph-tool python library. figshare. 2014; https://doi.org/10.6084/m9.figshare.1164194
41. Peixoto TP. The Netzschleuder network catalogue and repository. 2020. https://networks.skewed.de
42. Traag VA, Van Dooren P, Nesterov Y. Narrow scope for resolution-limit-free community detection. Phys Rev E Stat Nonlin Soft Matter Phys. 2011;84(1 Pt 2):016114. https://doi.org/10.1103/PhysRevE.84.016114 PMID: 21867264
43. Lancichinetti A, Fortunato S. Limits of modularity maximization in community detection. Phys Rev E Stat Nonlin Soft Matter Phys. 2011;84(6 Pt 2):066122. https://doi.org/10.1103/PhysRevE.84.066122 PMID: 22304170
44. Peng C, Kolda TG, Pinar A. Accelerating community detection by using k-core subgraphs. arXiv preprint 2014. arXiv:1403.2226
45. Kaminski B, Prałat P, Théberge F. Artificial benchmark for community detection with outliers (ABCD o). Applied Network Science. 2023;8(1):25.
46. Miasnikof P, Shestopaloff AY, Raigorodskii A. Statistical power, accuracy, reproducibility and robustness of a graph clusterability test. Int J Data Sci Anal. 2023;15(4):379–90. https://doi.org/10.1007/s41060-023-00389-6
47. Anne L, Vu-Le T-A, Park M, Warnow T, Chacko G. RECCS: realistic cluster connectivity simulator for synthetic network generation. Advs Complex Syst. 2025;28(05). https://doi.org/10.1142/s0219525925400041
48. Vu-Le T-A, Anne L, Chacko G, Warnow T. EC-SBM synthetic network generator. Appl Netw Sci. 2025;10(1). https://doi.org/10.1007/s41109-025-00701-2
49. Tabatabaee Y, Wedell E, Park M, Warnow T. Github site for FastEnsemble clustering. 2024. https://github.com/ytabatabaee/fast-ensemble
50. Tabatabaee Y, Wedell E, Park M, Warnow T. Github site for datasets for FastEnsemble. 2025. https://github.com/ytabatabaee/ensemble-clustering-data