Well-Connected Communities in Real-World and Synthetic Networks

Minhyuk Park,[1†] Yasamin Tabatabaee,[1†] Vikram Ramavarapu,[1†]
Baqiao Liu,[1] Vidya K. Pailodi,[1] Rajiv Ramachandran,[1] Dmitriy Korobskiy,[2]
Fabio Ayres,[3] George Chacko,[1,4*] Tandy Warnow[1*]

[1]Department of Computer Science, University of Illinois Urbana-Champaign, IL 61801, USA
[2]NTT DATA, McLean, VA 22102, USA
[3]Insper Institute, São Paulo, Brazil
[4]Grainger College of Engineering, University of Illinois Urbana-Champaign, IL 61801, USA

[*]To whom correspondence should be addressed; E-mail: chackoge@illinois.edu; warnow@illinois.edu
[†]Contributed equally to this manuscript.

Integral to the problem of detecting communities through graph clustering is the expectation that they are "well connected". In this respect, we examine five different community detection approaches optimizing different criteria: the Leiden algorithm optimizing the Constant Potts Model, the Leiden algorithm optimizing modularity, Iterative K-Core Clustering (IKC), Infomap, and Markov Clustering (MCL). Surprisingly, all these methods produce, to varying extents, communities that fail even a mild requirement for well connectedness. To remediate clusters that are not well connected, we have developed the "Connectivity Modifier" (CM), which, at the cost of coverage, iteratively removes small edge cuts and re-clusters until all communities produced are well connected. Results from real-world and synthetic networks illustrate a tradeoff users make between well connected clusters and coverage, and raise questions about the "clusterability" of networks and models of community structure. CM is available on github.

Introduction Community detection is of broad interest and is typically posed as a graph partitioning problem, where the input is a graph and the objective is a partitioning of its vertices into disjoint subsets, so that each subset represents a community (*1–3*). The terms community and cluster overlap heavily, so we use them interchangeably herein. Our interest in community detection is for the purpose of identifying research communities from the global scientific literature, so we are especially focused on methods that can scale to large networks consisting of documents linked by citation (*4–7*).

A unifying definition of community does not exist but a general expectation is that the vertices within a community are better connected to each other than to vertices outside the community (*8*), implying greater edge density within a community. However, a cluster may be dense while still having a small edge cut (*9*). Therefore, the minimum edge cut size (min cut) for a community should not be small (*10*). Thus, *edge density* and *well-connectedness*, i.e., not having a small edge cut, are two *separable* and expected properties of communities.

The Leiden algorithm (*10*), which builds upon the Louvain algorithm (*11*), is commonly used for community detection, with default quality function the Constant Potts Model (CPM) (*12*). Clusters produced by CPM-optimization have the desirable property that if the edge cut splits the cluster into components A and B, then the edge cut will be at least $r \times |A| \times |B|$ (*10, Supplementary Materials*), where r is a user-provided resolution parameter. This guarantee is strong when the edge cut splits a cluster into two components of approximately equal size, but is weaker when it produces an imbalanced split and weakest when the cut separates a single node from the remaining nodes in the cluster. Importantly, the guarantee depends on r, and small values of r produce weak bounds. Finally, we note that this guarantee applies to CPM-optimal clusterings but not to clusterings found by heuristics.

In using the Leiden software optimizing CPM, we observed that it produces clusters with small min cuts on seven different networks of varied origin ranging in size from approximately

34,000 to 75 million nodes. We also observed that the number of clusters with small min cuts increases as the resolution parameter decreases. Intrigued by this observation, we performed a broader study to evaluate the extent to which clusters produced by algorithms of interest meet even a mild standard for a well connected cluster.

To evaluate whether a cluster is well connected, we use a slow growing function $f(n)$ so that a cluster with n nodes whose min cut size is at most $f(n)$ will not be considered well connected. By design, we ensure that (i) $f(n)$ grows more slowly than the lower bound on the min cut size for clusters in CPM-optimal clusterings in the Leiden algorithm and (ii) that $f(n)$ provides a meaningful lower bound on the small-to-moderate values of n where the bound in (*10*) is weak. We selected $f(n) = \log_{10} n$ for this function.

We constructed min cut profiles from four additional clustering methods with different optimality criteria on the seven networks above: Leiden with modularity (*1*) as quality function; the *k-core* based Iterative k-core Clustering (IKC) (*4*) using $k = 10$; and two flow-based methods, Infomap (*13*) and Markov clustering (MCL) (*14*). None of these methods offer guarantees of well connected clusters. While only IKC and Leiden optimizing either CPM or modularity scaled to the largest network we studied, all tested methods produced poorly connected clusters (i.e., clusters with min cuts of size at most $f(n)$) on these networks. *These observations reveal a gap between the expectation of well connected clusters and what is actually being produced by these community finding methods.* These findings also raise questions about the "clusterability" (*15*) of networks and whether only portions of a network exhibit community structure.

For practical remediation, we developed the Connectivity Modifier (CM), available at `https://github.com/illinois-or-research-analytics/cm_pipeline` (*16*). CM takes a clustering as input and returns well connected clusters that are also at least a minimum size B, where the setting for B as well as the definition of "well connected" can be set by the user (our default setting uses the function $f(n)$ given above and $B = 11$). CM presently

provides support for Leiden optimizing either CPM or modularity and IKC, the methods that scaled to the largest network we studied. On real-world networks, using CM in conjunction with the Leiden method produces well connected clusters but with a reduction in node coverage. We observed similar results of lower magnitude with IKC. Analyses of synthetic networks with ground truth communities show somewhat different trends, revealing intriguing differences between synthetic and real-world networks.

The rest of this manuscript is organized as follows. First, we present an initial study on a large citation network showing conditions under which Leiden clusters are not well connected. Next we present comparable results from additional methods and networks. We then describe the design of Connectivity Modifier (CM) and show the impact of using CM on clusterings by Leiden and IKC on real world and synthetic networks. Finally, we close with a discussion of our findings.

Results

In a study of the Open Citations network (*17*) consisting of 75,025,194 nodes, we computed the min cut of all clusters generated using the Leiden algorithm optimizing either CPM or modularity (Fig. 1). Designating singleton clusters and very small clusters as not being of practical interest, we report node coverage throughout this manuscript as the percentage of nodes in clusters of size at least 11; the exception to this is Figure 4, where we report node coverage based on non-singleton clusters. For CPM, we used five different resolution values ($0.5, 0.10, 0.01, 0.001,$ and 0.0001) that resulted in node coverage values ranging from 6-99%.

For Leiden-CPM clusterings, we see that as the resolution value is decreased, (i) node coverage increases, (ii) the frequency of small mincuts increases, and (iii) cluster sizes increase (Fig. 1). Clustering under modularity is most similar to clustering under CPM at the lowest resolution value used. Strikingly, 98.7% of 2,184 clusters produced under modularity had a

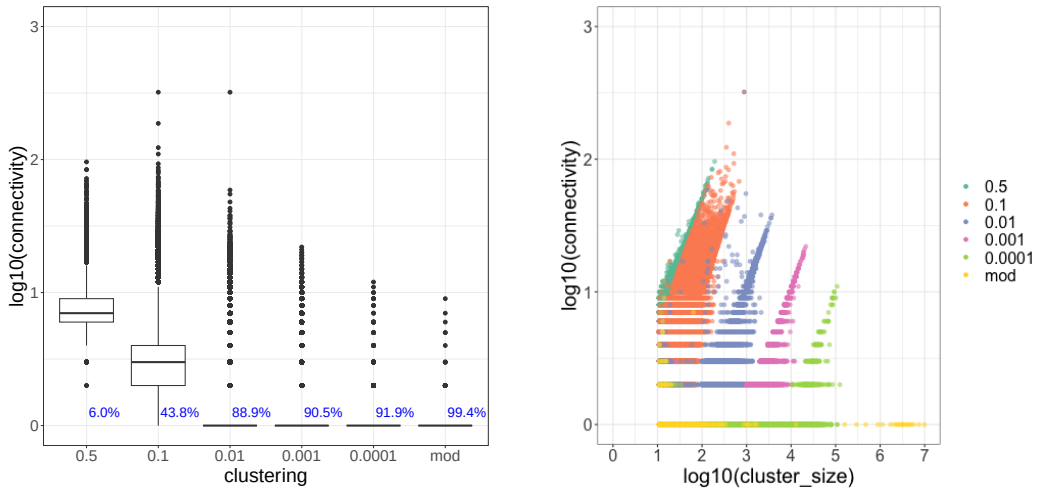


Figure 1: *Node coverage, connectivity, and size distribution of clusters generated by Leiden optimizing either CPM or modularity on the Open Citations network (75,025,194 nodes).* Connectivity (y-axis) is the minimum edge cut size (min cut) of each cluster. Node coverage, the percentage of nodes in clusters of at least size 11, is reported in blue text. Results are shown for clusters of at least size 11 from Leiden optimizing either CPM at five different resolution values or modularity. Higher node coverage is associated with reduced connectivity. Within each clustering, larger clusters are better connected although lower resolution values and modularity trend towards larger clusters that are less well connected.

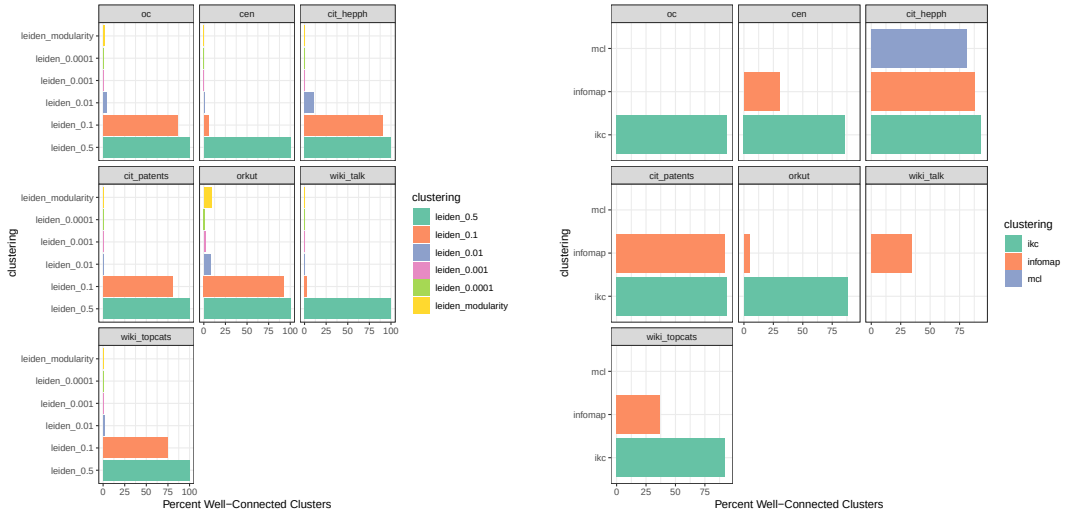
minimum edge cut size of 1 (i.e., could be split by removing a single edge) while accounting for 99.4% node coverage. Additionally, where node coverage is high, clusters tend to be larger and fewer although intermediate resolution values in the range we used result in an increase in the number of clusters (Fig. 1 and Table S1). Thus, these data illustrate a tradeoff that users make with the Leiden algorithm between small clusters, lower node coverage, and few small min cuts (achieved by CPM-optimization with larger resolution values) versus larger clusters, higher node coverage, and many more small min cuts (achieved by modularity-optimization or CPM-optimization with small resolution values).

While a large fraction of small min cuts intuitively signals "poorly-connected" clusters, there are different ways of formalizing this notion. Here we offer a formal definition for the purposes of this study. Briefly, we consider functions $f(n)$ with the interpretation that if a

cluster of size n has an edge cut of size at most $f(n)$ then the cluster will be considered poorly connected. We want $f(n)$ to grow very slowly so that it serves as a mild bound. We also want $f(n) \geq 1$ for all n that are large enough for the cluster to be considered a potential community. From three examples of slowly growing functions (Materials and Methods), we choose $f(n) = \log_{10} n$, the function that imposes the mildest constraint on large clusters and grows more slowly than the bound in (*10*).

In addition to the Open Citations network, we clustered six other networks, ranging in size from 34,546 nodes to 13,989,436 nodes, with Leiden, IKC, Infomap, and MCL and computed the percentage of clusters that we consider well connected (i.e., whose min cuts were greater than $f(n)$). Under the conditions used, Leiden and IKC ran to completion on all seven networks, although IKC did not return any clusters from wiki_talk because a 10-core does not exist in this sparse network. Infomap failed on the largest network, and MCL returned output only from the smallest network (cit_hepph) we analyzed (Fig. 2).

For Leiden clustering optimizing CPM (Fig. 1), the frequency of well connected clusters decreases with resolution value, and results from modularity are similar to the lowest resolution value for CPM that was tested. IKC completed on all networks returning well connected clusters that varied between 85.9% and 94% of the total number of clusters. In comparison to Leiden, IKC clustering resulted in lower node coverage, which is consistent with its more conservative formulation. Infomap produced well connected clusters varying from 5% (orkut) to 92.4% (cit_patents). MCL ran only on the cit_hepph network with 81.3% of the clusters being well connected. Interestingly, while neither Leiden nor IKC generated disconnected clusters, Infomap generated disconnected clusters for some networks (maximized at 75% for orkut) and MCL also generated disconnected clusters (7.2%) on the cit_hepph network. These observations reveal the widespread existence of clusters that are not well connected, with the extent dependent on clustering method and network.



(a) Leiden CPM & Modularity

(b) IKC, Infomap, MCL

Figure 2: *Percentage of Well Connected Clusters.* Clustering of seven networks by five different community finding approaches. Networks analyzed are Open Citations (75,025,194), Curated Exosome Network (13,989,436), cit_hepph (34,546), and cit_patents (3,774,768) are citation networks; orkut (3,072,441) is a social network; wiki_talk (2,394,385) and wiki_topcats (1,791,489) are Wikipedia communication and hyperlink networks respectively. Only Leiden and IKC ran to completion on all networks although IKC did not return any clusters from the wiki_talk network. Infomap completed on all but Open Citations. MCL completed only on cit_hepph.

Towards well connected clusters, we designed the Connectivity Modifier (CM) (*16, 18*) a remediation tool that can be used to modify a given clustering to ensure that each final cluster is well connected (Materials and Methods). Based on our preliminary findings, we chose to initially evaluate CM paired with the Leiden and IKC methods since these two clustering methods were sufficiently scalable and did not produce disconnected clusters. We also restricted our attention to the two largest networks, Open Citations and CEN.

We implemented CM in a pipeline (Fig. 3) that presently takes a Leiden or IKC clustering as input. A pre-processing (filtering) step discards very small clusters, i.e., those of size at most 10, but this bound can be changed by the user. Tree clusters are also discarded in this pre-processing step (given our definition of $f(n)$, any tree of size 10 or larger is not well connected). CM then

iteratively computes and removes any min cuts of size at most $f(n)$ and re-clusters until only well connected clusters remain. A post-processing step removes any small clusters of size at most 10 that may have resulted from repeated cutting.

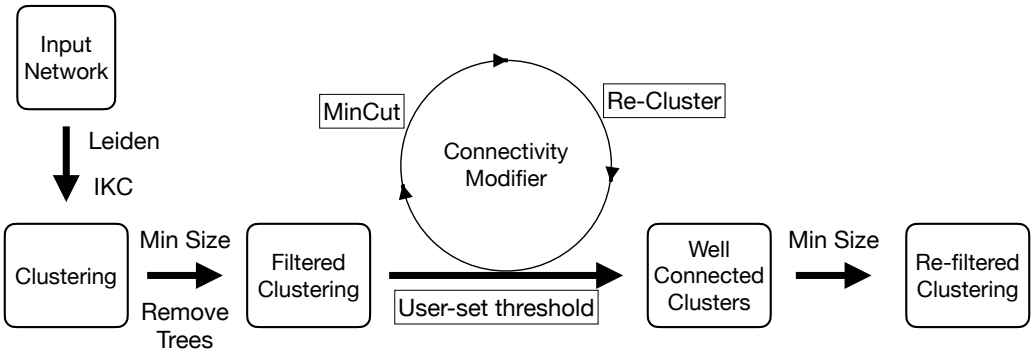
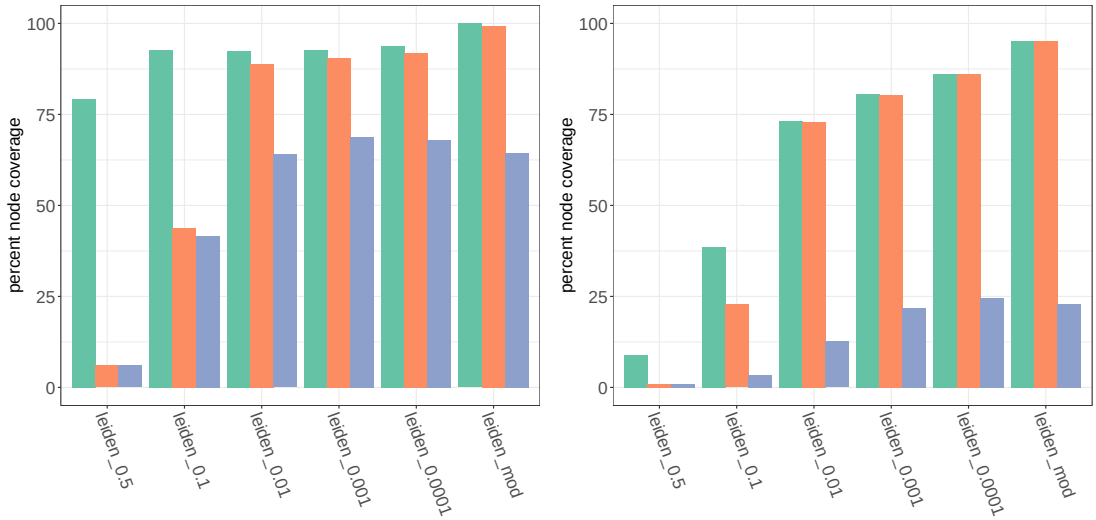


Figure 3: *Connectivity Modifier Pipeline Schematic.* The four-stage pipeline depends on user-specified algorithmic parameters: B, the minimum allowed size of a cluster, and $f(n)$, a bound on the minimum edge cut size for a cluster with n nodes, and clustering method. *Stage 1*: a clustering is computed. *Stage 2*: clusters are pre-processed by removing trees and those clusters of size less than B. *Stage 3*: the CM is applied to each cluster, removing edge cuts of sizes at most $f(n)$, reclustering, and recursing. *Stage 4*: clusters are post-processed by removing those of size less than B. All clusters returned are well connected according to $f(n)$ and have size at least B. Our study explored default settings with $B = 11$ and $f(n) = \log_{10} n$.

.

Effect of CM on node coverage We assessed the impact of CM on node coverage, here specifically examining clusters of size at least two. As above, we examined Leiden clustering of Open Citations and CEN networks using CPM-optimization at five resolution values and also using modularity, examining the change in node coverage before and after CM treatment (Fig. 4). The results are resolution dependent and network sensitive. For CPM-optimization, the impact of filtering out clusters of size at most 10 is large for the two larger resolution values, but then decreases as the resolution value decreases. In contrast, the impact of the Connectivity Modifier (the middle component of the CM pipeline that iteratively finds and removes small edge cuts and reclusters) also depends on the resolution parameter, with a minimal impact for the large resolution values and an increasing impact as the resolution value decreases. Modularity returned results most similar to CPM-optimization with the smallest tested resolution value. Since the pre-processing (filtering) and post-processing both remove all clusters of size at most 10, the node coverage reported for these stages are with respect to clusters of size at least 11. Given this, we observe that post-CM node coverage is low compared to pre-CM for both networks and clustering methods, and was smallest when using CPM-optimization with resolution value $r = 0.5$ and largest when using CPM-optimization with one of the two smallest resolution values, $r = 0.001$ for CEN and $r = 0.0001$ for Open Citations. Overall, post-CM node coverage of any Leiden clustering never exceeded 24.6% for CEN and 68.7% for Open Citations.

Cluster fate To understand the nature of the modifications effected by CM, we further classified the Leiden clusters based on the impact of CM-processing: *extant, reduced, split, and degraded*, where "extant" indicates that the cluster was not modified by CM, "reduced" indicates that the cluster is reduced in size, "split" indicates that the cluster was divided into at



(a) Open Citations (b) CEN

Figure 4: *Reduction in node coverage after CM treatment of Leiden clusters.* The Open Citations (left panel) and CEN (right panel) networks were clustered using the Leiden algorithm under CPM at five different resolution values or modularity. Node coverage (defined as the percentage of nodes in cluster of size at least 2) was computed for Leiden clusters • (lime green), Leiden clusters with trees and/or clusters of size 10 or less filtered out • (soft orange), and after CM treatment of filtered clusters • (desaturated blue).

least two smaller clusters, and "degraded" indicates that the cluster was reduced to singletons or a cluster of size at most 10. We report the fraction in each category (Fig. 5) for six different clustering conditions on the Open Citations and CEN networks. When using CM with CPM-optimization with a very large resolution value, most clusters are extant, the fraction of extant clusters decreases for CPM-optimization as the resolution value decreases, and is low also for modularity (Fig. 2).

An interesting trend with split clusters is seen in Figure 5, indicating the initial cluster contained two or more well connected clusters; this is similar to the observation by Fortunato and Barthélemy in (*19*) of modularity's behaviour on a ring-of-cliques, where modularity returns clusters that contain two or more cliques, instead of returning the individual cliques, under some conditions. Here we see that clusters that are split by CM unsurprisingly occur in both

networks with modularity, but also occur in a noticeable way for CPM-optimization with the smallest resolution value for the Open Citations network. Finally, we note that this occurs for CPM-optimization with other resolution values on both networks (Table S5 and Figure S1). The most extreme cluster fate is of course when it is degraded to singletons, which occurs in modularity clusterings and CPM-based clusterings at lower resolution values; however, the degree to which this occurs depends on the network and resolution value in a way that does not suggest any particular pattern (Figure S1).

Clustering with IKC We examine the impact of CM on IKC (with $k = 10$) on the Open Citations and CEN networks (Fig. 6). In comparison to Leiden, IKC-clustering results in relatively low node coverage, 23.6% and 3.8% in the case of the Open Citations and CEN networks respectively. CM treatment of these clusterings also has a small effect on node coverage. We also see that most clusters are extant and some are split, but none are reduced or degraded.

In summary, the response to CM-processing differed across methods. On the CEN and Open Citations networks, and for both IKC and Leiden under modularity or CPM, node coverage in a post-CM clustering did not exceed 68%, either because CM-processing reduces node coverage substantially from the initial clustering in the case of Leiden with modularity or CPM-clustering with smaller resolution values or because the initial clustering was conservative and already had low node coverage. This suggests the possibility that in these real-world networks, only a fraction of the nodes may be in clusters that are sufficiently well connected and sufficiently large. In other words, real-world networks may not be fully covered by what we consider "valid" communities.

Analysis of synthetic networks Given our hypothesis that the large reduction in node coverage that we observe on real-world networks may be the result of them not being universally covered by well connected true communities, we examined synthetic networks, noting that

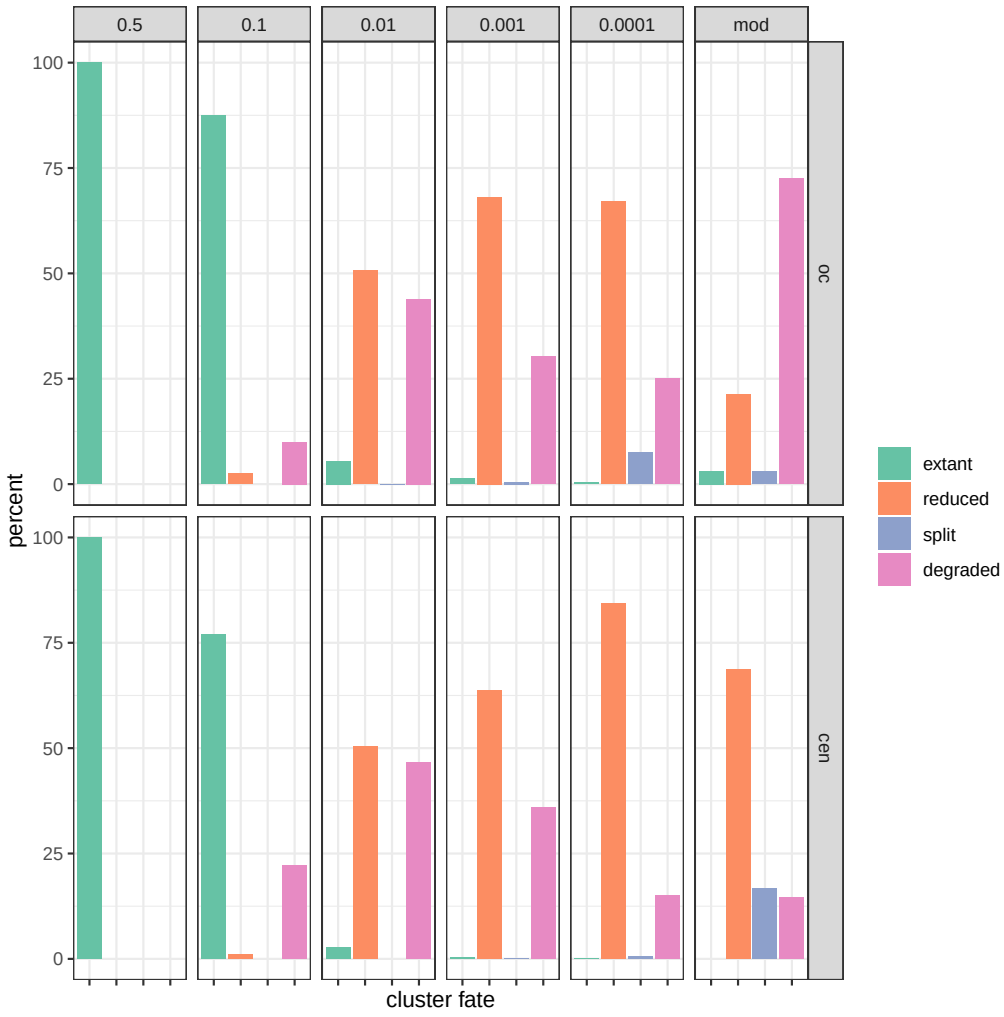
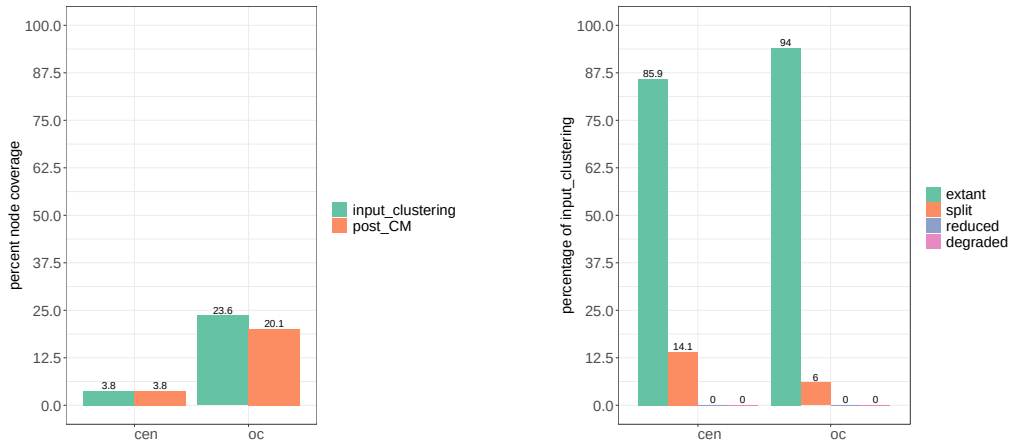


Figure 5: *Cluster Fate for Leiden.* CM-processed clusters are classified as extant (unaffected by CM), reduced (replaced by a single smaller cluster), split (replaced by two or more clusters), or degraded (replaced entirely by singletons). Top: Cluster Fate for Open Citations (OC); Bottom: Cluster fate for the CEN. Data are shown for CM treatment of Leiden clusters, under either CPM for 5 different resolution values or modularity. Cluster counts are expressed as a percentage of the input clusters.

synthetic networks generated using LFR software (*20*) assign every node to a non-singleton ground-truth community. If node coverage after CM-processing of LFR networks were similar to real-world networks, it would argue against this hypothesis. Examining synthetic networks also enables us to evaluate the impact of CM-processing on accuracy.



(a) Node coverage

(b) Cluster Fate

Figure 6: *Node coverage (a) and cluster fate (b) for IKC clusters modified by CM on the Open Citations and CEN.* For IKC on these two networks, node coverage is small on both the CEN and Open Citations, and slightly reduced by CM on the Open Citations network. We also see that most clusters are extant (unaffected by CM-processing) and some are split (replaced by two or more smaller clusters), but none are reduced (replaced by a single smaller cluster) or degraded (replaced by only singletons).

For this experiment, we computed statistics for the Leiden clusterings of the 7 real-world networks we explored, Open Citations, CEN, and 5 networks from the SNAP collection (*21*)), and used them as input to the LFR software (*20*) (see Materials and Methods). Using this software, we were not able to build synthetic networks for the CEN and Open Citations with the number of nodes in the empirical networks; therefore, for these two specific cases, we constructed the LFR networks to 3M nodes. For some combinations of network and Leiden clustering, we were unable to produce any synthetic networks, for example, we were unable to produce LFR networks for the Orkut social network. We produced a collection of 34 LFR networks with ground truth communities that had similar empirical statistics as their corresponding real-world networks (Supplementary Materials Section S3). We clustered each of these 34 LFR networks using the same clustering method used to provide empirical statistics to LFR.

Results in Figure 7 show node coverage after clustering for both the empirical network and

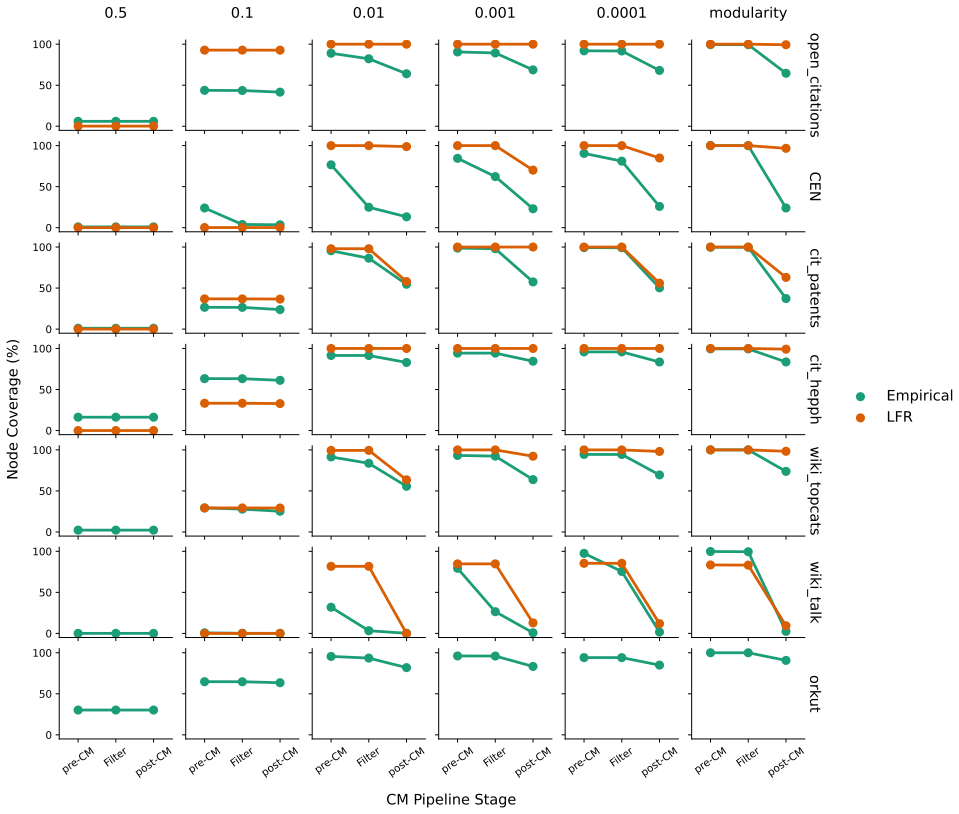


Figure 7: *The effect on node-coverage (percentage of nodes in clusters of size at least 11) produced by the CM processing pipeline on LFR and real-world networks.* Each row corresponds to a real-world network, and each column corresponds to a Leiden clustering (either modularity or CPM-optimization with the specified resolution value). Within each entry, we show node coverage for the Leiden clustering as it goes through the CM pipeline; results on the empirical network are shown in green and results for the LFR network are shown in red. "Pre-CM" indicates the output of the Leiden clustering, "filter" is after removing trees and clusters below size 11, and "post-CM" is after the entire pipeline (which also removes any final clusters below size 11). No results are shown for some combinations (specifically, data are not shown for LFR for any clustering of the Orkut network nor for the wiki_topcats and wiki_talk networks with resolution value $r = 0.5$) due to LFR failing to generate networks for those settings.

its corresponding LFR network. The general trends we observed in previous experiments for OC and CEN also hold for the five other empirical networks examined here both with respect to node coverage in Leiden clustering and how CM-processing impacts node coverage. An interesting trend we see here is that node coverage drops for some CPM clusterings of empirical networks

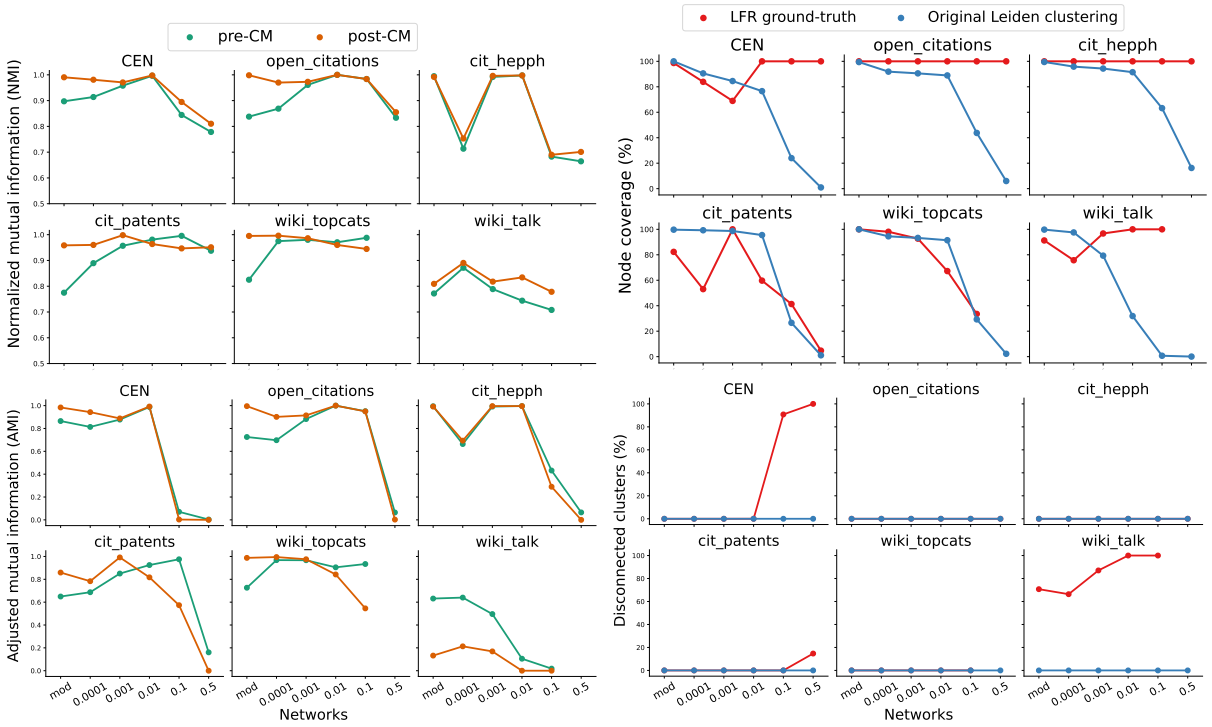


Figure 8: *Impact of CM-processing on accuracy of synthetic networks.* The left panels show accuracy measured in terms of NMI and AMI, with respect to the LFR ground-truth communities. Each condition on the x-axis corresponds to a *different* LFR network, generated based on Leiden-modularity or Leiden-CPM with that specific resolution parameter. In total, there are 34 different LFR networks. The right panels show percentage of the LFR ground-truth communities that are small or disconnected. In most conditions, CM improves the accuracy of the original Leiden clustering, except for some of the conditions when the ground-truth communities have many (at least 60%) disconnected clusters, or the node coverage by clusters of size at least 11 is low (at most 70%).

though not for LFR networks after the filtering stage. Since node coverage in this figure is with respect to clusters of size 11 or larger, any drop in node coverage resulting from filtering is due only to removing trees. The large drop in node coverage for Leiden with CPM-optimization in these cases means that CPM-optimization in some conditions produces a large number of tree clusters. We note this did not occur for resolution value $r = 0.5$, where clusters tend to be small but did occur for other resolution values (Supplementary Tables S3 and S4).

Of specific interest is whether clusterings of the LFR networks respond similarly to CM-

processing as clusterings of their corresponding empirical networks, as this addresses our hypothesis regarding why CM-processing impacts node coverage in empirical networks. While node coverage does drop to the same degree for some LFR networks as for the empirical networks on which they are based, there are many cases where node coverage drops much more for the real-world network than for the corresponding LFR network, and no cases where there is a bigger drop in node coverage for the LFR network than for the real-world network. Thus, the idea that real-world networks not having universal coverage by valid communities cannot be ruled out.

We examined the impact of CM-processing on clustering accuracy; results for Normalized Mutual Information (NMI) and Adjusted Mutual Information (AMI) are shown in Figure 8, and results for the Adjusted Rand Index (ARI) are similar to AMI and are provided in Supplementary Materials, Figure S11 (*22–24*). We see that CM-processing improves NMI accuracy for modularity and also for CPM-optimization when used with small resolution values. CM-processing tends to be neutral for NMI otherwise, and was only detrimental on two network:clustering pairs. The impact on AMI accuracy is more variable. For example, CM-processing reduced AMI accuracy for all wiki_talk network:clustering pairs except for CPM-optimization with $r = 0.1$ where accuracy was very low and the impact was neutral. CM-processing also reduced accuracy for CPM-optimization on some network:clustering pairs for large resolution values.

However, when examining the properties of the ground-truth clusterings of these networks, we see that the cases where CM-processing produced a noteworthy reduction in accuracy for NMI or AMI are those where there are many disconnected ground truth clusters, for example all wiki_talk clusters, or where the node coverage by clusters of size at least 11 is small, for example cit_patents at the three largest resolution values and wiki_topcats with the two largest resolution values. However, there are conditions with low node coverage or a large fraction of

disconnected clusters where CM-processing is neutral or even beneficial.

The occurrence of disconnected ground-truth clusters in the LFR networks is striking and problematic, since a basic expectation of a community is that it is connected, if not well-connected (*10*). Hence, we assert that it is unreasonable to evaluate accuracy with respect to a ground-truth set of communities if the communities are not connected. In other words, it does not make sense to evaluate clustering accuracy for those LFR networks that contain many disconnected ground truth communities, including the entire set of LFR networks constructed for wiki_talk invalid, which is one of the conditions where CM-processing reduces AMI accuracy. The fact that LFR networks had ground truth clusters that were not connected also indicates the failure of LFR software to reproduce features of the input network:clustering pairs, which by construction always have 100% of the clusters connected.

It is easy to see why a low node coverage by clusters of size at least 11 could reduce accuracy for CM-processing, since CM automatically removes all small clusters. However, this property depends on the network:clustering pair, which depends on network features as well as the clustering methodology. In this study, clustering real-world networks produced a large fraction of small clusters when we used CPM-optimization with large resolution values; the conditions where CM-processing reduced AMI accuracy on the LFR networks with low node coverage by clusters of size at least 11 are drawn from those conditions. We also note that users typically select the clustering method that produces cluster sizes that match their interest. Therefore, CM-processing will not be beneficial where there is interest in recovering small communities unless the bound B is replaced by a smaller value.

In interpreting these results, we also note the discrepancy between some empirical statistics of LFR networks and those of the real-world network:clustering pairs that were used to simulate the LFR network. Beyond incomplete matching of features, differences such as the frequency of disconnected clusters and the percentage of clusters that are small make it questionable whether

accuracy on LFR networks is suggestive of accuracy on real-world networks.

Discussion In this study, we considered the question of whether clusters produced by community detection methods are well connected. We use a mild definition to demonstrate that five different clustering paradigms generate, to varying extents, output clusters that are not well connected. An important implication of these results is that portions of a network may not exhibit community structure. Further, at parameter settings that maximize node coverage, weakly connected parts of a network may be forced into communities. Related prior work (*15, 25*) address whether a graph in general is clusterable, which is a related question.

We developed CM to convert, through partitioning, poorly connected clusters into well connected ones. For flexibility, the function in CM that defines "well connected" can be modified by users to be more or less stringent. Similarly, we provide a parameter B, tunable by the user, that specifies the smallest size of a cluster for it to be retained; this too can be modified by the user to be more or less stringent. At present, CM provides support only for Leiden and IKC, the most scalable of the methods we tested. CM is being extended to provide support for Infomap and MCL and concurrently being redesigned for developers to integrate their own clustering methods into it. CM allows the user to explore cluster quality in the input, as it reveals which clusters are poorly connected, and, in some cases, finds substructure within clusters. Thus, CM-processing can be used to evaluate and improve clustering outputs, and interrogate and explore the community structure within a given network.

Several factors affect how significantly CM-processing changes a given clustering. These include the network itself, as some networks seem to be more impacted by CM-processing. We also see that the choice of resolution parameter for Leiden-CPM has an influence on how much the clustering changes, with generally larger impact for small resolution values.

The findings that LFR networks produce different patterns than empirical networks is per-

haps not surprising, but here we consider potential explanations. First, the LFR methodology assumes that the degree distribution and cluster size distributions follow a power law, however, it is not clear that this is universal to citation and real-world networks (*26–29*). Furthermore, we also observed that the degree distribution and cluster size distributions were imperfectly fit by LFR software in our study (Supplementary Materials, Figs. S2 and S3). Hence, the assumptions about node degree and cluster size distributions that govern the LFR model may not result in adequate simulation of real-world networks.

Another assumption in the LFR methodology is that every node is in a community, which is one that would benefit from deeper investigation. Intuitively, we posit that the assumption that every node in a real world community is in a community may only be reasonable if the communities can be small and/or poorly connected.

We recognize that our perspective on well connected clusters may result in narrow descriptions of communities, perhaps akin to cores of core-periphery structures (*4, 30, 31*). Additionally, informative weaker links (*32*) may be lost from communities since CM partitions input clusters that are poorly connected into sets of well connected clusters. However, such weak links are not lost from the network being analyzed.

Having developed CM and ascertained its quantitative effects, a direction for our future work is to evaluate it in the context of specific evaluations that are supported by mixed methods. More generally, we emphasize leaving the definition, use, and interpretation of well connected to users.

Materials and Methods

Defining well connected clusters. In (*10*), Traag et al. proved that every cluster C in a CPM-optimal clustering of a network (with resolution parameter r) would satisfy the following

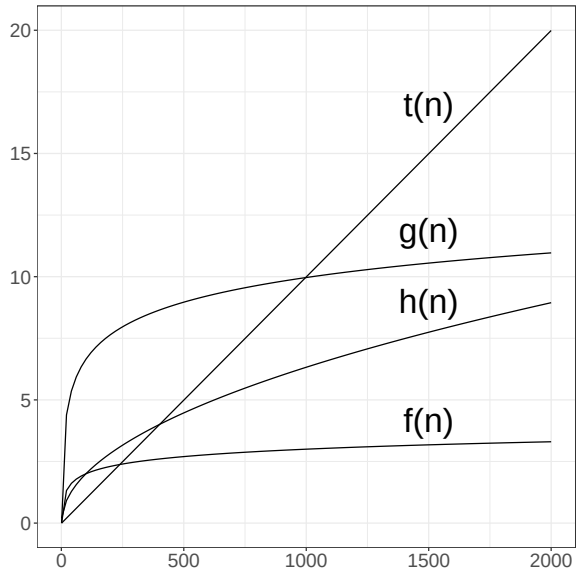


Figure 9: *Comparison of the lower bounds on edge cut sizes for defining well connected clusters.* Here we compare four functions; $t(n) = 0.01(n-1)$ is Traag's function, which provides a lower bound of the cut size for a cluster on n nodes in a CPM-optimal clustering when $r = 0.01$, $f(n) = \log_{10} n$, $g(n) = \log_2 n$, and $h(n) = \frac{\sqrt{n}}{5}$. For $n \geq 1000$, Traag's function provides the largest lower bound on the edge cut size, and so is the strongest guarantee of the functions we compare, while $f(n)$ provides the smallest lower bound for all $n \geq 239$. However, for $n \leq 238$, $f(n) \geq t(n)$, so that $f(n)$ provides a stronger guarantee on these small- to moderate-sized clusters than Traag's bound.

property: any edge cut X of C splitting the nodes into two sets A and B would have at least $r \times |A| \times |B|$ edges. This function is maximized when $|A| = |B|$ and minimized when $|A| = 1$ and $|B| = n - 1$, where C has n nodes. Hence in particular, this establishes that the minimum cut for any cluster with n nodes has size at least $r \times (n - 1)$.

We have already observed that this bound (which we refer to as Traag's bound) is weak when n is not large and r is very small. For example, when $r = 0.01$ and $n = 50$, the bound only establishes that the minimum edge cut is at least 1, which therefore simply asserts that the cluster is connected. Hence, we seek lower bounds on the size of the minimum edge cut that are larger than Traag's bound of $r \times (n - 1)$ for small values of n but grow very slowly, and so

do not exceed Traag's bound for larger n.

In the text, we discussed the bound that requires that for a cluster of n nodes to be well connected, the size of its min cut must be *strictly greater* than $f(n) = \log_{10} n$. We compare $f(n)$ to Traag's lower bound when $r = 0.01$, which we refer to as $t(n)$, as well as two other functions, $g(n) = \log_2 n$ and $h(n) = \frac{\sqrt{n}}{5}$; note that each of $f(n), g(n)$ and $h(n)$ is strictly positive and increasing, and yet grows more slowly than Traag's function $t(n)$.

The comparison between these functions in the range of cluster sizes $1 \leq n \leq 2000$ is shown in Figure 9. As desired, for large enough n, Traag's function $t(n)$ dominates the other functions, thus imposing a much stronger guarantee on the minimum cut size for the cluster. We also note that $f(n) < g(n)$ for all $n > 1$, but that the relationship between $f(n), h(n)$, and $t(n)$ depends on n. For large n, however, $f(n)$ is less than all the other functions, making it the most slowly growing function.

Based on this comparison, we used $f(n)$ to define when a cluster of n nodes is well connected: the min cut size is strictly greater than $\log_{10}(n)$; otherwise we say that the cluster is poorly connected.

Note that if we had picked $g(n)$ instead, we would have considered more clusters poorly connected, since $f(n) < g(n)$ for all n. Similarly, picking $h(n)$ would have generally been more stringent a requirement (at least for all but very small values of n), and so would have ruled more clusters as being poorly connected. Thus, $f(n)$ represents a very modest constraint on the size of the min cut for a cluster, in order for it to qualify as being well connected.

The Connectivity Modifier (CM) Pipeline To remediate poorly-connected clusters, we developed a modular pipeline that we now describe (Fig. 2). By design, this pipeline is guaranteed to return a clustering where each cluster is well connected according to the function $f(n)$ and has size at least B. The values of $f(n)$ and B used in this study are set as default but can be

easily modified by a user. Note that if an input cluster meets these two criteria, it will be present in the output clustering. Furthermore, every cluster in the output will either be one of the input clusters or will be a subset of one of the input clusters.

The CM pipeline requires the user to specify values for three algorithmic parameters:

- B, the minimum allowed size of any "valid" vertex community (default $B = 11$)
- $f(n)$, so that a cluster is well connected only if its min cut size exceeds $f(n)$ (default: $f(n) = \log_{10} n$)
- A clustering method, presently selected from Leiden optimizing CPM, Leiden optimizing modularity, or the Iterative k-core (IKC) method with $k = 10$. Support for additional methods is in active development.

The input to the CM pipeline is a network $\mathcal{G}$ with N nodes and the algorithmic parameters as specified. The pipeline then operates in four stages:

- Stage 1: A clustering is generated from the input network $\mathcal{G}$.
- Stage 2: The clustering is filtered to remove clusters below size B and all trees, which are not well-connected when using $f(n) = \log_{10} n$
- Stage 3: The Connectivity Modifier (CM) is applied to each cluster that remains. First, all nodes of degree at most $f(n)$ are removed (where n is the number of nodes in the cluster), until there are no low degree nodes remaining. Then, for each remaining cluster, a min cut is calculated using VieCut (*33*), and if it is not greater than $f(n)$ in size then the min cut is removed, thus splitting the cluster into two components. These components are then re-clustered using the selected clustering method, and the process repeats until all clusters are well-connected.
- Stage 4: Any resultant clusters below size B are removed.

Additional details for the Connectivity Modifier The Connectivity Modifier was developed using Python 3.10. The original implementation of VieCut (*33*) in C++17 was wrapped into a Python package using PyBind11. We use VieCut to find minimum edge cuts, using the NOI algorithm (*34*). CM is under active development; this study used CM v3.3 of CM, available at `https://github.com/illinois-or-research-analytics/cm_pipeline` (*16*).

Additional details for IKC Due to memory constraints, IKC fails to run with the connectivity modifier on the entirety of the OpenCitations network. To solve this, we modified IKC to run in memory as an imported Python module rather than as a separate executable. Moreover, we split the initial IKC clustering of OpenCitations into separate clusters, and the Connectivity Modifier was then run on the largest clusters independently (see Supplementary Section S2 for full details).

Data A custom-implemented Extract-Transform-Load (ETL) process was designed to process the publicly available Open Citations dataset, downloaded in Aug 2022, and load it into a PostgreSQL table. The resultant network contained 75,025,194 nodes and 1,363,605,603 edges. The CEN is a citation network constructed from the literature on exosome research. From the SNAP repository, we downloaded cit_hepph, an Arxiv High Energy Physics paper citation network; cit_patents, a citation network among US patents; orkut, a social media network; wiki_talk, a network containing users and discussion from the inception of Wikipedia until January 2008; and wiki_topcats, a web graph of Wikimedia hyperlinks. Before clustering, all networks were processed to remove self-loops as well as duplicate and parallel edges.

LFR (synthetic) networks To create simulated networks with ground truth communities, while attempting to emulate the properties of each empirical network and its corresponding Leiden clustering, we used the LFR software from (*20*). The generative model of the LFR

network	nodes	edges	avg_deg	ref
Open Citations	75,025,194	1,363,605,603	36.35	(*17*)
CEN	13,989,436	92,051,051	13.16	(*35*)
cit_hepph	34,546	420,877	24.37	(*36*)
cit_patents	3,774,768	16,518,947	8.75	(*36*)
orkut	3,072,441	117,185,083	76.28	(*37*)
wiki_talk	2,394,385	4,659,565	3.89	(*38*)
wiki_topcats	1,791,489	25,444,207	28.41	(*39*)

Table 1: Summary statistics for networks used in this study. Average degree is the average of the node degrees across the network.

graphs assume that the node degree and the community size distributions are power-law distributions (*40*). The software for generating LFR benchmark graphs (*41*) takes the following eight parameters as input:

- Network properties: Number of nodes N, average and maximum node degrees (k and k_{max} respectively), and negative exponent for degree sequence (τ_1) that is assumed to be a power-law.
- Community properties: Maximum and minimum community sizes (c_{max} and c_{min}), and negative exponent for the community size distribution (τ_2), also modeled as a power-law.
- Mixing parameter μ, that is the ratio between the degree of a node outside its community and its total degree, averaged over all nodes in the network. Lower μ values suggest that a network consists of well-separated communities, as nodes are mostly connected to other nodes inside their communities, rather than outside of it.

Parameter Estimation. To emulate the empirical networks using LFR graphs, we estimated all eight parameters described above for a given pair of network $\mathcal{G}$ and a clustering $\mathcal{C}$. Computing N, k, k_{max}, c_{min} and c_{max} is straightforward using *networkX* (*42, 43*). To estimate μ, we perform a single iteration over all edges of the network, and for each edge, if the nodes on the

two sides of it were in different communities, that edge contributes to the ratio μ of these two nodes. The total μ of the network:clustering pair is the average μ across all the nodes.

To estimate τ_1 and τ_2, we fit a power-law distribution to the node degree sequence and the community size distribution, using the approach from (*44*) that is implemented in the *power-law* Python package (*45*). Note that the power-law property may hold for the tail of the degree or community size sequence and not the whole distributions. Therefore, following (*44*), we estimate x_{min}, the minimum value for which the power-law property holds as well as the exponent α for the tail of the distribution.

Generating LFR networks. After computing these parameters based on the Leiden clusterings of the empirical networks using both modulatiry and CPM with a range of resolution parameters, we simulated LFR networks using the software from (*20*), producing empirical statistics reported in Supplementary Tables S7 and S8.

For networks with more than 10 million nodes, i.e., Open Citations and the CEN, we limited the number of vertices to 3 million, due to scalability limitations of the LFR benchmark graph generator (*46*), while preserving the edge density reflected by average degree, and the mixing parameter. The numbers of nodes of the other LFR graphs exactly match the number of nodes in the corresponding empirical network. In some cases, due to the inherent limitations of the LFR graph generator, we had to modify the ranges of the community sizes, i.e., increase c_{min} and decrease c_{max}, to generate the network. These values are available with the datasets.

As shown in the statistics reported in Supplementary Tables S7 and S8, the average node degree, the mixing parameter, and the exponents for the degree and community size distributions are in all cases very well-preserved. Further details about the pipeline for producing LFR graphs and the statistics of the graphs are provided in Supplementary Materials Section 3. We calculated NMI, AMI, and ARI using the Python Scikit-Learn package (*47*). Other software

used includes Leiden and IKC (*48, 49*).

References

1. M. E. Newman, M. Girvan, Finding and evaluating community structure in networks. *Physical review E* **69**, 026113 (2004).

2. P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, J.-P. Onnela, Community structure in time-dependent, multiscale, and multiplex networks. *Science* **328**, 876–878 (2010).

3. S. Fortunato, M. E. J. Newman, 20 years of network community detection. *Nature Physics* **18**, 848–850 (2022).

4. E. Wedell, M. Park, D. Korobskiy, T. Warnow, G. Chacko, Center-periphery structure in research communities. *Quantitative Science Studies* **3**, 289–314 (2022).

5. K. W. Boyack, R. Klavans, Creation of a highly detailed, dynamic, global model and map of science. *Journal of the Association for Information Science and Technology* **65**, 670–685 (2013).

6. L. Waltman, N. J. van Eck, A new methodology for constructing a publication-level classification system of science. *Journal of the American Society for Information Science and Technology* **63**, 2378–2392 (2012).

7. K. W. Boyack, D. Newman, R. J. Duhon, R. Klavans, M. Patek, J. R. Biberstine, B. Schijvenaars, A. Skupin, N. Ma, K. Börner, Clustering more than two million biomedical publications: Comparing the accuracies of nine text-based similarity approaches. *PLoS ONE* **6**, e18029 (2011).

8. M. Coscia, F. Giannotti, D. Pedreschi, A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining* **4**, 512–546 (2011).

9. F. Bonchi, D. García-Soriano, A. Miyauchi, C. E. Tsourakakis, Finding densest k-connected subgraphs. *Discrete Applied Mathematics* **305**, 34–47 (2021).

10. V. A. Traag, L. Waltman, N. J. Van Eck, From Louvain to Leiden: guaranteeing well-connected communities. *Scientific reports* **9**, 1–12 (2019).

11. V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* **2008**, P10008 (2008).

12. V. A. Traag, P. V. Dooren, Y. Nesterov, Narrow scope for resolution-limit-free community detection. *Physical Review E* **84** (2011).

13. M. Rosvall, C. T. Bergstrom, Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences* **105**, 1118–1123 (2008).

14. S. V. Dongen, Graph clustering via a discrete uncoupling process. *SIAM Journal on Matrix Analysis and Applications* **30**, 121–141 (2008).

15. P. Miasnikof, A. Y. Shestopaloff, A. Raigorodskii, Statistical power, accuracy, reproducibility and robustness of a graph clusterability test. *International Journal of Data Science and Analytics* **15**, 379–390 (2023).

16. V. Ramavarapu, F. Ayres, M. Park, V. K. Pailodi, G. Chacko, T. Warnow, Connectivity modifier (2023). Available online at `https://github.com/illinois-or-research-analytics/cm_pipeline`.

17. S. Peroni, D. Shotton, OpenCitations, an infrastructure organization for open scholarship. *Quantitative Science Studies* **1**, 428–444 (2020).

18. B. Liu, M. Park, Connectivity modifier, `https://github.com/RuneBlaze/connectivity-modifier` (2022).

19. S. Fortunato, M. Barthelemy, Resolution limit in community detection. *Proceedings of the National Academy of Sciences* **104**, 36–41 (2007).

20. A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms. *Physical review E* **78**, 046110 (2008).

21. J. Leskovec, A. Krevl, SNAP Datasets: Stanford large network dataset collection (2014). Http://snap.stanford.edu/data.

22. L. Danon, A. Diaz-Guilera, J. Duch, A. Arenas, Comparing community structure identification. *Journal of statistical mechanics: Theory and experiment* **2005**, P09008 (2005).

23. N. X. Vinh, J. Epps, J. Bailey, *Proceedings of the 26th annual international conference on machine learning* (2009), pp. 1073–1080.

24. L. Hubert, P. Arabie, Comparing partitions. *Journal of classification* **2**, 193–218 (1985).

25. C. Gao, J. Lafferty, Testing for global network structure using small subgraph statistics (2017). ArXiv:1710.00862.

26. F. Radicchi, S. Fortunato, C. Castellano, Universality of citation distributions: Toward an objective measure of scientific impact. *Proceedings of the National Academy of Sciences* **105**, 17268–17272 (2008).

27. M. J. Stringer, M. Sales-Pardo, L. A. N. Amaral, Statistical validation of a global model for the distribution of the ultimate number of citations accrued by papers published in a scientific journal. *Journal of the American Society for Information Science and Technology* **61**, 1377–1385 (2010).

28. I. Artico, I. Smolyarenko, V. Vinciotti, E. C. Wit, How rare are power-law networks really? *Proceedings of the Royal Society A* **476**, 20190742 (2020).

29. M. Brzezinski, Power laws in citation distributions: evidence from Scopus. *Scientometrics* **103**, 213–228 (2015).

30. R. Breiger, *Explorations in Structural Analysis (RLE Social Theory)* (Routledge, 2014).

31. P. Rombach, M. A. Porter, J. H. Fowler, P. J. Mucha, Core-periphery structure in networks (revisited). *SIAM Review* **59**, 619–646 (2017).

32. M. S. Granovetter, The strength of weak ties. *American Journal of Sociology* **78**, 1360–1380 (1973).

33. M. Henzinger, A. Noe, C. Schulz, D. Strash, Practical minimum cut algorithms. *ACM Journal of Experimental Algorithmics* **23** (2018).

34. H. Nagamochi, T. Ono, T. Ibaraki, Implementing an efficient minimum capacity cut algorithm. *Math. Program.* **67**, 325–341 (1994).

35. A. Jakatdar, B. Liu, T. Warnow, G. Chacko, AOC: Assembling overlapping communities. *Quantitative Science Studies* **3**, 1079–1096 (2022).

36. J. Leskovec, J. Kleinberg, C. Faloutsos, *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (ACM, 2005), pp. 177–187.

37. J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* **42**, 181–213 (2013).

38. J. Leskovec, D. Huttenlocher, J. Kleinberg, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (ACM, 2010), pp. 1361–1370. https://doi.org/10.1145/1753326.1753532.

39. H. Yin, A. R. Benson, J. Leskovec, D. F. Gleich, *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2017), pp. 555–564.

40. R. Albert, A.-L. Barabási, Statistical mechanics of complex networks. *Reviews of modern physics* **74**, 47 (2002).

41. S. Fortunato, Resources (2023). https://www.santofortunato.net/resources.

42. A. Hagberg, P. Swart, D. S Chult, Exploring network structure, dynamics, and function using networkx, *Tech. rep.*, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States) (2008).

43. Y. Tabatabaee, Emulating real networks using LFR graphs (2023). https://github.com/ytabatabaee/emulate-real-nets.

44. A. Clauset, C. R. Shalizi, M. E. Newman, Power-law distributions in empirical data. *SIAM review* **51**, 661–703 (2009).

45. J. Alstott, E. Bullmore, D. Plenz, powerlaw: a python package for analysis of heavy-tailed distributions. *PloS one* **9**, e85777 (2014).

46. G. M. Slota, J. W. Berry, S. D. Hammond, S. L. Olivier, C. A. Phillips, S. Rajamanickam, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2019), pp. 1–14.

47. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, Scikit-learn: Machine learning in Python. *the Journal of Machine Learning Research* **12**, 2825–2830 (2011).

48. V. Traag, Leiden algorithm: leidenalg, `https://github.com/vtraag/leidenalg` (2019).

49. E. Wedell, M. Park, Iterative k-core software (2021). `https://github.com/chackoge/ERNIE_Plus/blob/master/Illinois/clustering/eleanor/code/IKC.py`.

Acknowledgements: The authors thank Nathan Bryans, Christine Ballard, and Bryan Barker from Oracle Research for their assistance in setting up and using the Oracle Cloud Infrastructure,

Funding: This work was supported in part by Insper-Illinois Collaboration and by a Research Award to TW from Oracle Research.

Author Contributions GC and TW conceived the research; GC, TW, and YT designed the analyses. VR, FA, MP, BL, YT, VP, DK, and RR developed software to support analyses. MP, YT, GC, VR, VP conducted the analyses. GC, TW, YT, and MP interpreted the data and wrote the manuscript.

Competing Interests The authors declare that they have no competing financial interests.

Data and materials availability: Additional data and materials are available online.

Supplementary Materials for "Well-Connected Communities in Real-World and Synthetic Networks"

Minhyuk Park*[1], Yasamin Tabatabaee*[1], Vikram Ramavarapu*[1], Baqiao Liu[1], Vidya Kamath Pailodi[1], Rajiv Ramachandran[1], Dmitriy Korobskiy[2], Fabio Ayres[3], George Chacko†[1,4], and Tandy Warnow‡[1]

[1]Department of Computer Science, University of Illinois Urbana-Champaign, Urbana, IL 61801, USA
[2]NTT DATA, McLean, VA 22102, USA
[3]Insper Institute, São Paulo, Brazil
[4]Office of Research, Grainger College of Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801, USA

Contents

List of Tables

List of Figures

*Minhyuk Park, Yasamin Tabatabaee, and Vikram Ramavarapu contributed equally
†chackoge@illinois.edu
‡warnow@illinois.edu

S1 Additional Results

Table S1: Overview of all networks and methods analyzed with cluster counts (clus_count), node coverage (*nc*, the percentage of nodes in clusters of size at least 11), and min, median, and max cluster sizes.

network	method	parameter	clus_count	nc	min	median	max
oc	leiden	0.5	20966119	6.0	2	2	192
oc	leiden	0.1	8642175	43.8	2	5	882
oc	leiden	0.01	2134603	88.9	2	15	3729
oc	leiden	0.001	839902	90.5	2	3	21385
oc	leiden	0.0001	561116	91.9	2	2	125017
oc	leiden	modularity	184257	99.4	2	2	9922297
oc	ikc	10	2569	23.6	11	40	6650349
cen	leiden	0.5	433761	1.0	2	2	70
cen	leiden	0.1	517669	24.0	2	11	320
cen	leiden	0.01	280544	76.6	2	28	3236
cen	leiden	0.001	66470	84.5	2	97	13025
cen	leiden	0.0001	11779	90.5	2	377	70138
cen	leiden	modularity	253	100.0	2	28	1312837
cen	ikc	10	128	3.8	14	79	214877
cen	infomap	default	187	100.0	2	103	2314354
cit_hepph	leiden	0.5	7569	16.3	2	3	55
cit_hepph	leiden	0.1	3050	63.2	2	6	169
cit_hepph	leiden	0.01	810	91.5	2	9	744
cit_hepph	leiden	0.001	366	94.3	2	3	3436
cit_hepph	leiden	0.0001	284	95.8	2	2	20000
cit_hepph	leiden	modularity	82	99.5	2	2	4031
cit_hepph	ikc	10	28	20.1	14	59.5	1530
cit_hepph	infomap	default	68	99.6	2	2	13027
cit_hepph	mcl	2.0	3036	69.0	2	4	887
cit_patents	leiden	0.5	1143221	1.1	2	2	116
cit_patents	leiden	0.1	556688	26.6	2	5	330
cit_patents	leiden	0.01	134380	95.5	2	19	804
cit_patents	leiden	0.001	29463	98.7	2	63	3268
cit_patents	leiden	0.0001	9125	99.3	2	4	20000
cit_patents	leiden	modularity	3708	99.7	2	2	198537
cit_patents	ikc	10	582	2.0	15	35	23468
cit_patents	infomap	default	3902	99.7	2	2	245582
orkut	leiden	0.5	617430	30.2	2	2	217
orkut	leiden	0.1	232145	64.8	2	6	1444
orkut	leiden	0.01	47094	95.5	2	21	12016
orkut	leiden	0.001	11269	96.1	2	44	46062
orkut	leiden	0.0001	11348	94.1	2	2	190195
orkut	leiden	modularity	36	100.0	3	22572.5	800146
orkut	ikc	10	758	43.7	11	29	386103
orkut	infomap	default	20	100.0	8	26660	1417010
wiki_talk	leiden	0.5	47808	0.1	2	2	133
wiki_talk	leiden	0.1	48481	0.7	2	4	1141
wiki_talk	leiden	0.01	31983	31.9	2	11	200
wiki_talk	leiden	0.001	16035	79.3	2	48	1205
wiki_talk	leiden	0.0001	7315	97.5	2	167	13635
wiki_talk	leiden	modularity	2739	99.8	2	2	225817
wiki_talk	ikc	NA	NA	NA	NA	NA	NA
wiki_topcats	leiden	0.5	455139	2.2	2	2	89
wiki_topcats	leiden	0.1	240673	29.3	2	5	523
wiki_topcats	leiden	0.01	58343	91.4	2	17	2547
wiki_topcats	leiden	0.001	20257	93.3	2	3	18024
wiki_topcats	leiden	0.0001	13269	94.5	2	2	125725
wiki_topcats	leiden	modularity	27	100.0	155	45467	273395
wiki_topcats	ikc	10	170	6.8	11	28	45613
wiki_talk	infomap	default	22191	98.6	2	17	53773

Table S2: Overview of all networks and methods analyzed with cluster counts (cc) and node coverage (nc) of all clusters with size at least 11 (> 10), of all non-singleton clusters at most size 10 ($> 1, <= 10$), and of all singleton clusters ($= 1$). The number of nodes and node coverage of all non-singleton clusters are at the very end of each row.

network	method	parameter	cc (> 10)	cc ($> 1, <= 10$)	cc ($= 1$)	nc (> 10)	nc ($> 1, <= 10$)	nc ($= 1$)	num nodes (> 1)	nc (> 1)
cen	ikc	10	128	0	13454271	3.8	0	96.2	535165	3.8
wiki_topcats	ikc	10	170	0	1669262	6.8	0	93.2	122227	6.8
cit_hepph	ikc	10	28	0	27611	20.1	0	79.9	6935	20.1
cit_patents	ikc	10	582	0	3699025	2	0	98	75743	2
orkut	ikc	10	758	0	1729355	43.7	0	56.3	1343086	43.7
oc	ikc	10	2569	0	57301796	23.6	0	76.4	17723398	23.6
cit_hepph	mcl	2.0	612	2424	862	69	28.5	2.5	33684	97.5
cen	infomap	default	130	57	0	100	0	0	13989436	100
wiki_talk	infomap	default	14491	7700	35	98.6	1.4	0	2394350	100
wiki_topcats	infomap	default	5676	3161	1	98.9	1.1	0	1791488	100
cit_hepph	infomap	default	8	60	0	99.6	0.4	0	34546	100
cit_patents	infomap	default	297	3605	0	99.7	0.3	0	3774768	100
orkut	infomap	default	19	1	0	100	0	0	3072441	100
cen	leiden	0.0001	11327	452	1323426	90.5	0	9.5	12666010	90.5
cen	leiden	0.001	65915	555	2165097	84.5	0	15.5	11824339	84.5
cen	leiden	0.01	275807	4737	3238339	76.6	0.3	23.1	10751097	76.9
cen	leiden	0.1	273971	243698	8332294	24	16.5	59.6	5657142	40.4
cen	leiden	0.5	8448	425313	12701688	1	8.2	90.8	1287748	9.2
cen	leiden	modularity	187	66	0	100	0	0	13989436	100
wiki_talk	leiden	0.0001	4488	2827	52664	97.5	0.3	2.2	2341721	97.8
wiki_talk	leiden	0.001	9622	6413	479532	79.3	0.6	20	1914853	80
wiki_talk	leiden	0.01	16278	15705	1585195	31.9	1.9	66.2	809190	33.8
wiki_talk	leiden	0.1	1403	47078	2145665	0.7	9.7	89.6	248720	10.4
wiki_talk	leiden	0.5	60	47748	2292833	0.1	4.2	95.8	101552	4.2
wiki_talk	leiden	modularity	166	2573	0	99.8	0.2	0	2394385	100
wiki_topcats	leiden	0.0001	1148	12121	64237	94.5	1.9	3.6	1727252	96.4
wiki_topcats	leiden	0.001	8014	12243	87381	93.3	1.9	4.9	1704108	95.1
wiki_topcats	leiden	0.01	44040	14303	96200	91.4	3.2	5.4	1695289	94.6
wiki_topcats	leiden	0.1	27734	212939	172523	29.3	61.1	9.6	1618966	90.4
wiki_topcats	leiden	0.5	2666	452473	575060	2.2	65.7	32.1	1216429	67.9
wiki_topcats	leiden	modularity	27	0	0	100	0	0	1791489	100
cit_hepph	leiden	0.0001	14	270	630	95.8	2.3	1.8	33916	98.2
cit_hepph	leiden	0.001	63	303	1029	94.3	2.7	3	33517	97
cit_hepph	leiden	0.01	380	430	1370	91.5	4.6	4	33176	96
cit_hepph	leiden	0.1	821	2229	2094	63.2	30.7	6.1	32452	93.9
cit_hepph	leiden	0.5	346	7223	5187	16.3	68.7	15	29359	85
cit_hepph	leiden	modularity	18	64	0	99.5	0.5	0	34546	100
cit_patents	leiden	0.0001	3238	5887	7621	99.3	0.5	0.2	3767147	99.8
cit_patents	leiden	0.001	21596	7867	21815	98.7	0.7	0.6	3752953	99.4
cit_patents	leiden	0.01	114308	20072	64155	95.5	2.8	1.7	3710613	98.3
cit_patents	leiden	0.1	58858	497830	123020	26.6	70.2	3.3	3651748	96.7
cit_patents	leiden	0.5	2653	1140568	1049310	1.1	71.1	27.8	2725458	72.2
cit_patents	leiden	modularity	94	3614	0	99.7	0.3	0	3774768	100
orkut	leiden	0.0001	691	10657	153956	94.1	0.9	5	2918485	95
orkut	leiden	0.001	6279	4990	106634	96.1	0.4	3.5	2965807	96.5
orkut	leiden	0.01	41535	5559	115873	95.5	0.7	3.8	2956568	96.2
orkut	leiden	0.1	59555	172590	147860	64.8	30.4	4.8	2924581	95.2
orkut	leiden	0.5	45170	572260	347743	30.2	58.5	11.3	2724698	88.7
orkut	leiden	modularity	34	2	0	100	0	0	3072441	100
oc	leiden	0.0001	38701	522415	4697754	91.9	1.8	6.3	70327440	93.7
oc	leiden	0.001	231899	608003	5487615	90.5	2.1	7.3	69537579	92.7
oc	leiden	0.01	1361462	773141	5721905	88.9	3.4	7.6	69303289	92.4
oc	leiden	0.1	1311357	7330818	5413061	43.8	49	7.2	69612133	92.8
oc	leiden	0.5	297230	20668889	15583737	6	73.2	20.8	59441457	79.2
oc	leiden	modularity	2184	182073	0	99.4	0.6	0	75025194	100

Table S3: Table of Tree Cluster counts (a). The count of clusters of size at least 11 is provided for trees, stars (a type of tree), and non-tree clusters for all seven networks clustered with Leiden (CPM at various resolutions or modularity). The total number of clusters of size at least 11 is the sum of these three types

	Network	res	type	N
1	cen	0.0001	non_tree	9520
2	cen	0.0001	star	1186
3	cen	0.0001	tree	621
4	cen	0.001	non_tree	43083
5	cen	0.001	star	18064
6	cen	0.001	tree	4768
7	cen	0.01	non_tree	64566
8	cen	0.01	star	205229
9	cen	0.01	tree	6012
10	cen	0.1	non_tree	18576
11	cen	0.1	star	255395
12	cen	0.5	non_tree	8448
13	cen	mod	non_tree	48
14	cen	mod	tree	8
15	cen	mod	star	131
16	cit_hepph	0.0001	non_tree	14
17	cit_hepph	0.001	non_tree	63
18	cit_hepph	0.01	non_tree	379
19	cit_hepph	0.01	tree	1
20	cit_hepph	0.1	non_tree	820
21	cit_hepph	0.1	star	1
22	cit_hepph	0.5	non_tree	346
23	cit_hepph	mod	non_tree	18
24	cit_patents	0.0001	non_tree	3024
25	cit_patents	0.0001	tree	178
26	cit_patents	0.0001	star	36
27	cit_patents	0.001	non_tree	20608
28	cit_patents	0.001	tree	939
29	cit_patents	0.001	star	49
30	cit_patents	0.01	non_tree	91410
31	cit_patents	0.01	star	415
32	cit_patents	0.01	tree	22483
33	cit_patents	0.1	non_tree	58543
34	cit_patents	0.1	star	315
35	cit_patents	0.5	non_tree	2653
36	cit_patents	mod	non_tree	75
37	cit_patents	mod	tree	9
38	cit_patents	mod	star	10
39	oc	0.0001	non_tree	33705
40	oc	0.0001	tree	3519
41	oc	0.0001	star	1477
42	oc	0.001	non_tree	207713
43	oc	0.001	star	1499
44	oc	0.001	tree	22687
45	oc	0.01	non_tree	1033761
46	oc	0.01	star	8591
47	oc	0.01	tree	319110
48	oc	0.1	non_tree	1297088
49	oc	0.1	star	14268
50	oc	0.1	tree	1
51	oc	0.5	non_tree	297230
52	oc	mod	non_tree	912
53	oc	mod	star	508
54	oc	mod	tree	764

Table S4: Table of Tree Cluster counts (b). The count of clusters of size at least 11 is provided for trees, stars (a type of tree), and non-tree clusters for all seven networks clustered with Leiden (CPM at various resolutions or modularity). The total number of clusters of size at least 11 is the sum of these three types.

	gp	res	type	N
1	orkut	0.0001	non_tree	677
2	orkut	0.0001	tree	13
3	orkut	0.0001	star	1
4	orkut	0.001	non_tree	6199
5	orkut	0.001	tree	79
6	orkut	0.001	star	1
7	orkut	0.01	non_tree	37470
8	orkut	0.01	star	13
9	orkut	0.01	tree	4052
10	orkut	0.1	non_tree	59372
11	orkut	0.1	star	183
12	orkut	0.5	non_tree	45170
13	orkut	mod	non_tree	34
14	wiki_talk	0.0001	non_tree	3104
15	wiki_talk	0.0001	star	128
16	wiki_talk	0.0001	tree	1256
17	wiki_talk	0.001	non_tree	3760
18	wiki_talk	0.001	star	1797
19	wiki_talk	0.001	tree	4065
20	wiki_talk	0.01	non_tree	2579
21	wiki_talk	0.01	star	7850
22	wiki_talk	0.01	tree	5849
23	wiki_talk	0.1	non_tree	133
24	wiki_talk	0.1	star	1270
25	wiki_talk	0.5	non_tree	60
26	wiki_talk	mod	non_tree	149
27	wiki_talk	mod	tree	7
28	wiki_talk	mod	star	10
29	wiki_topcats	0.0001	non_tree	1088
30	wiki_topcats	0.0001	tree	47
31	wiki_topcats	0.0001	star	13
32	wiki_topcats	0.001	non_tree	7611
33	wiki_topcats	0.001	tree	311
34	wiki_topcats	0.001	star	92
35	wiki_topcats	0.01	non_tree	36854
36	wiki_topcats	0.01	star	891
37	wiki_topcats	0.01	tree	6295
38	wiki_topcats	0.1	non_tree	25364
39	wiki_topcats	0.1	star	2370
40	wiki_topcats	0.5	non_tree	2666
41	wiki_topcats	mod	non_tree	27

Table S5: Split Cluster Fates. For all seven networks (rows) clustered with Leiden with CPM at various resolutions or modularity (columns), the maximum number subclusters for a cluster is shown; by definition, the minimum number is 2. The highest number of subclusters is seen for modularity clustering and low resolution values of CPM. Results not shown are because there are no split clusters.

	gp	0.5	0.1	0.01	0.001	0.0001	mod
1	oc			2	3	5	149
2	cen				2	2	21
3	cit_patents				3	5	210
4	cit_hepph				2	2	7
5	orkut				2	4	61
6	wiki_talk						8
7	wiki_topcats				2	5	15

Table S6: Extant Cluster counts for IKC clusterings. The fraction of clusters that are extant, i.e., not modified by CM, is expressed as a percentage of total clusters. No results are shown for wiki_talk because IKC did not return any clusters for that network.

	not extant	extant	perc_extant
CEN	18	110	85.9
OC	154	2415	94.0
cit_hepph	2	26	92.9
cit_patents	38	544	93.5
wiki_topcats	14	156	91.8
orkut	91	667	88.0

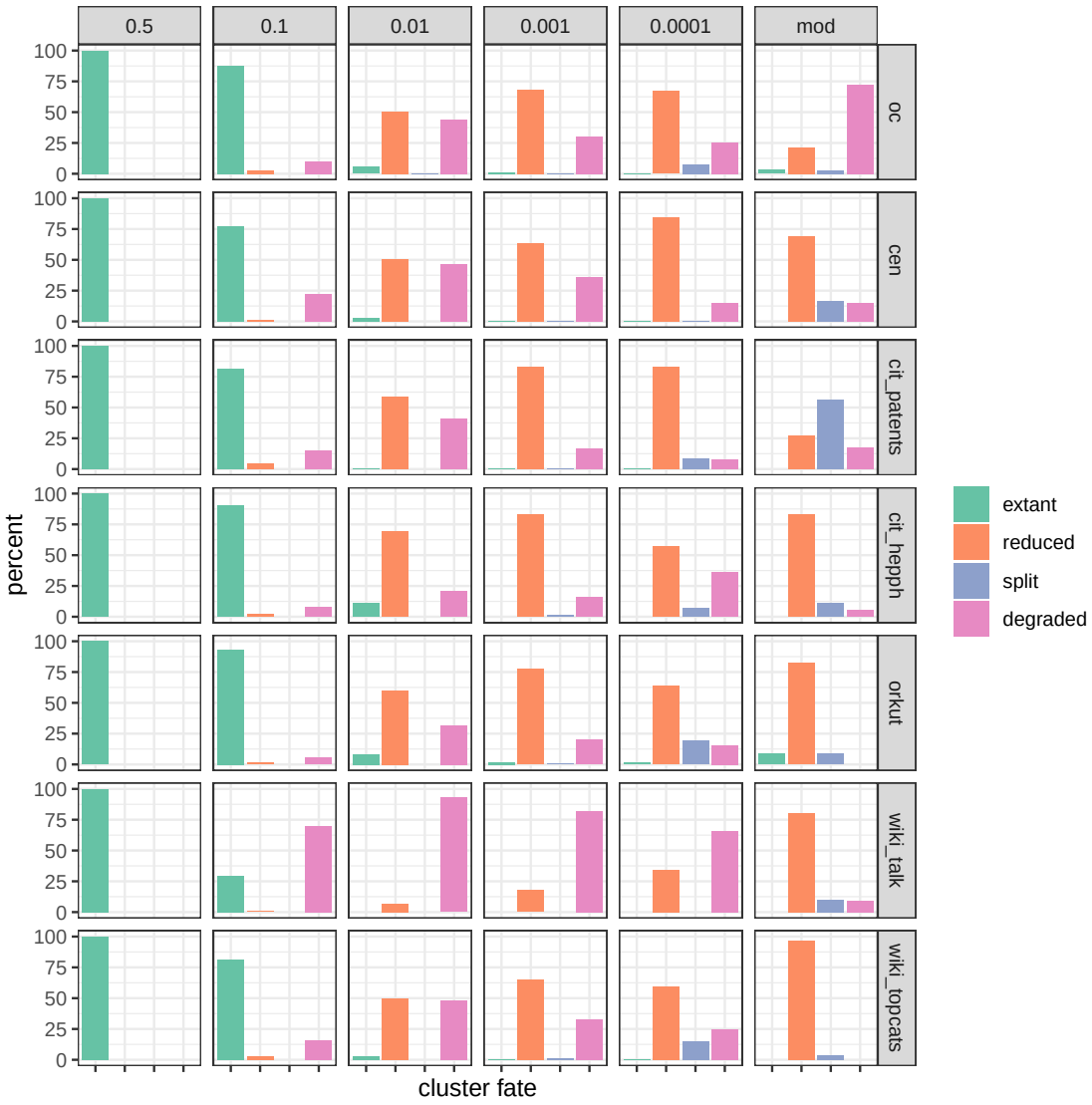


Figure S1: Cluster Fate for Leiden clusters on 7 different networks. The fate of Leiden clusters from seven different networks as extant (not modified by CM), reduced (replaced by a smaller cluster), split (replaced by two or more smaller clusters), or degraded (replaced entirely by singletons) is indicated. Rows correspond to networks and columns correspond to clustering approach. Data are shown for CM treatment of Leiden clusters, under either CPM for 5 different resolution values or modularity. As resolution is decreased, the number of extant clusters decreases; the number of clusters that are reduced in size increases then decreases; the number of clusters that are split generally increases with the exception of modularity, which varies according to network; and the number of clusters that are degraded to singletons increases and then decreases.

S2 Connectivity Modifier Pipeline

Preprocessing an input network

We remove duplicate/parallel edges and self-loops from the networks, using the `cleanup_el.R` script with the following command:

```
Rscript cleanup_el.R <original_network.tsv> <cleaned_network.tsv>
```

This step is now included as Stage 1 in the pipeline.

CM Pipeline

The cleaned input network $\mathcal{G}$ is then processed in a pipeline that has the following stages:

- **Clustering**: Clustering $\mathcal{G}$, using either the Leiden algorithm (in default mode with two iterations) or Iterative K-core Clustering (IKC) method, with the following commands respectively:

  ```
  leidenalg.find_partition(net, leidenalg.CPMVertexPartition, resolution_parameter=r)

  IKC.py -e <cleaned_network.tsv> -k 10 -o <output file>
  ```

  We used the Python version of Leiden (leidenalg v0.8.2) and IKC (v1.0.0). For the Open Citations network, we ran CM on the IKC clustering as follows. Each cluster of size greater than 100,000 nodes was run in a separate CM analysis until completion or failure. The remaining clusters were run in a single CM analysis together. The output clusterings from all of the individual CM runs were then combined to make the final output clustering. Two of the CM runs, originating from two large IKC clusters, could not complete due to memory issues. Any nodes from these two clusters were returned as singleton clusters in the final output clustering.

- **Filtering during pre-processing**: Removing clusters of size at most 10 as well as trees (defined as acyclic connected clusters, or equivalently connected clusters where the number of edges is exactly one less than the number of nodes). We used the following command for this step:

  - `subset_graph_nonnetworkit.R` followed by `make_cm_ready.R`

- **CM**: Applying connectivity modifier with the following commands (assuming the clustering method used is Leiden), where the option `-g` specifies the resolution parameter:

  ```
  $ cm -i <cleaned_network.tsv> -c leiden -e <filtered_leiden_clustering.tsv> -g <r>
  -t 1log10 -o <cm_output.tsv>
  $ cm2universal -g <cleaned_network.tsv> -i <cm_output.tsv> -o <cm_output.tsv>
  ```

- **Filtering during post-processing**: Removing clusters of size at most 10, using the same commands as the first filtering step.

  - `post_cm_filter.R`: the default, a simple script that removes all clusters of size 10 or less.

Important note: The current version of the CM pipeline is an improvement over the earlier version, available at https://github.com/RuneBlaze/connectivity-modifier. Some of the changes fixed errors in the implementation in terms of faithfully performing the algorithmic steps (for example, how disconnected clusters are handled, whether given as input or created during the CM pipeline), and other changes were made to ensure that how we recorded 'extant clusters' was done correctly.

S3 Experiments on LFR Graphs

S3.1 Emulating Real World Networks with LFR Graphs

The code for estimating the parameters of a pair of network and clustering and generating an LFR benchmark graph that represents their properties is available at https://github.com/ytabatabaee/emulate-real-nets and was used with the following commands.

```
python3 estimate_properties.py -n <network.tsv> -c <clustering_memberships.tsv>
```

that produces a json file containing all parameters of the network/clustering pair and then this json file is used to generate an LFR graph as follows

```
python3 gen_lfr.py -n <clustering_memberships.json> -lp <lfr-benchmark-software-path>
-cm <cmin_value>
```

in which the LFR software is used with the following command:

```
./binary_networks/benchmark -N <node-count> -k <avg-degree>
                            -maxk <max-degree> -mu <mixing-parameter>
                            -maxc <max-cluster-size> -minc <min-cluster-size>
                            -t1 <degree-exponent> -t2 <comm-size-exponent>
```

As discussed in the main paper, a somewhat limited range of parameters can be used to successfully generate an LFR graph. Due to restrictions of the LFR methodology, we made the following adjustments in our experiments for emulating our collection of empirical networks:

- LFR software is not scalable to large networks, and therefore we emulated the two largest networks (Open Citations and CEN) with LFR graphs with 3,000,000 nodes, and adjusted the maximum degree and maximum community size accordingly.
- The Wikipedia talk network (wiki_talk) has a very small average degree of 3.89, a maximum degree of 100029, and an estimated power-law degree exponent of 1.90. Based on the assumptions of the LFR methodology, we had to reduce the input maximum degree (`maxk`) to 31, as for the given average degree, any value above this for `maxk` resulted in an error from the LFR software.
- In all our real networks with all resolution parameters, the minimum community size was 1, as singletons were always present in the empirical networks. However, the LFR methodology assumes that all nodes are in a valid community. In most cases, when setting `minc` $= 1$, the LFR software either took very long to run (more than our time limit of 4 hours per network), or could not generate the community size distribution with the given properties at all. Therefore, with the exception of high energy physics citation network for which `minc` $= 1$ successfully generated the LFR, for other networks, we found the *minimum* value of `minc` that generated the LFR graph in less than 4 hours with a brute-force search (exploring all values of `minc` starting from 1, to the point where the LFR was generated in the given time limit).

Figures S2 and S3 show a comparison between the degree distribution and community size distributions of the six empirical networks and their corresponding LFR graphs for different resolution values. Tables S7 and S8 show various parameters of the empirical networks and their corresponding LFR graphs.

S3.2 LFR Accuracy Experiments

For the accuracy experiments in the paper, we compared the accuracy of the pre-CM and post-CM clusterings of the LFR networks with respect to the LFR ground-truth communities. We used the normalized mutual information (NMI), adjusted mutual information (AMI), and adjusted Rand index (ARI) criteria that are commonly used for partition comparison, as implemented in the Scikit-learn library. The nodes in the original network that are removed from the post-CM clustering are added back as singletons (with distinct clusters), and hence the partitions are compared on the whole set of nodes.

Table S7: Properties of the empirical networks and their LFR model graphs for Leiden-CPM clusterings. Parameters: r stands for resolution value used for generating the Leiden clustering, μ stands for the mixing parameter, and τ_1 and τ_2 are the *estimated* power-law exponents for the degree and the community size distributions respectively.

network	r	nodes	edges	average degree	μ	τ_1	τ_2
open_citations	0.0001	75,025,194	1,363,303,678	36.34	0.407	2.974	4.045
open_citations_LFR	0.0001	3,000,000	55,134,095	36.76	0.407	2.978	4.036
open_citations	0.001	75,025,194	1,363,303,678	36.34	0.500	2.974	4.375
open_citations_LFR	0.001	3,000,000	55,067,530	36.71	0.500	2.980	4.372
open_citations	0.01	75,025,194	1,363,303,678	36.34	0.602	2.974	5.205
open_citations_LFR	0.01	3,000,000	54,801,081	36.53	0.602	2.983	5.195
open_citations	0.1	75,025,194	1,363,303,678	36.34	0.711	2.974	6.194
open_citations_LFR	0.1	3,000,000	54,906,125	36.60	0.711	2.980	6.192
open_citations	0.5	75,025,194	1,363,303,678	36.34	0.871	2.974	6.152
open_citations_LFR	0.5	3,000,000	55,104,605	36.74	0.871	2.978	6.135
CEN	0.0001	13,989,436	92,051,051	13.16	0.402	2.618	2.259
CEN_LFR	0.0001	3,000,000	20,817,560	13.88	0.402	2.620	2.269
CEN	0.001	13,989,436	92,051,051	13.16	0.522	2.618	2.368
CEN_LFR	0.001	3,000,000	20,809,023	13.87	0.522	2.620	2.372
CEN	0.01	13,989,436	92,051,051	13.16	0.645	2.618	5.420
CEN_LFR	0.01	3,000,000	20,554,876	13.70	0.646	2.616	5.402
CEN	0.1	13,989,436	92,051,051	13.16	0.879	2.618	6.184
CEN_LFR	0.1	3,000,000	20,743,710	13.83	0.878	2.622	6.152
CEN	0.5	13,989,436	92,051,051	13.16	0.988	2.618	3.270
CEN_LFR	0.5	3,000,000	20,821,520	13.88	0.988	2.620	3.296
cit_patents	0.0001	3,774,768	16,518,947	8.75	0.211	4.017	2.981
cit_patents_LFR	0.0001	3,774,768	15,640,593	8.29	0.211	4.000	2.974
cit_patents	0.001	3,774,768	16,518,947	8.75	0.284	4.017	4.830
cit_patents_LFR	0.001	3,774,768	15,642,211	8.29	0.284	4.024	4.829
cit_patents	0.01	3,774,768	16,518,947	8.75	0.382	4.017	2.565
cit_patents_LFR	0.01	3,774,768	15,640,109	8.29	0.382	4.000	2.573
cit_patents	0.1	3,774,768	16,518,947	8.75	0.511	4.017	4.639
cit_patents_LFR	0.1	3,774,768	15,643,333	8.29	0.511	4.010	4.623
cit_patents	0.5	3,774,768	16,518,947	8.75	0.805	4.017	4.162
cit_patents_LFR	0.5	3,774,768	15,606,781	8.27	0.807	4.004	4.124
cit_hepph	0.0001	34,546	420,877	24.37	0.086	3.631	2.313
cit_hepph_LFR	0.0001	34,546	431,138	24.96	0.086	3.632	2.280
cit_hepph	0.001	34,546	420,877	24.37	0.219	3.631	1.465
cit_hepph_LFR	0.001	34,546	431,138	24.96	0.219	3.632	1.659
cit_hepph	0.01	34,546	420,877	24.37	0.384	3.631	1.825
cit_hepph_LFR	0.01	34,546	430,104	24.90	0.384	3.632	2.089
cit_hepph	0.1	34,546	420,877	24.37	0.570	3.631	2.333
cit_hepph_LFR	0.1	34,546	431,138	24.96	0.570	3.632	6.137
cit_hepph	0.5	34,546	420,877	24.37	0.781	3.631	2.790
cit_hepph_LFR	0.5	34,546	431,138	24.96	0.781	3.632	3.544
wiki_topcats	0.0001	1,791,489	25,444,207	28.41	0.379	2.430	1.645
wiki_topcats_LFR	0.0001	1,791,489	24,504,754	27.36	0.379	2.440	1.684
wiki_topcats	0.001	1,791,489	25,444,207	28.41	0.544	2.430	1.913
wiki_topcats_LFR	0.001	1,791,489	24,504,163	27.36	0.544	2.440	1.927
wiki_topcats	0.01	1,791,489	25,444,207	28.41	0.682	2.430	2.429
wiki_topcats_LFR	0.01	1,791,489	24,491,676	27.34	0.682	2.441	2.432
wiki_topcats	0.1	1,791,489	25,444,207	28.41	0.791	2.430	3.281
wiki_topcats_LFR	0.1	1,791,489	24,346,081	27.18	0.793	2.445	3.281
wiki_topcats	0.5	1,791,489	25,444,207	28.41	0.902	2.430	4.045

network	r	nodes	edges	average degree	μ	τ_1	τ_2
wiki_talk	0.0001	2,394,385	4,659,565	3.89	0.170	1.901	2.824
wiki_talk_LFR	0.0001	2,394,385	3,240,464	2.71	0.162	2.138	2.816
wiki_talk	0.001	2,394,385	4,659,565	3.89	0.346	1.901	1.901
wiki_talk_LFR	0.001	2,394,385	3,299,276	2.76	0.342	2.081	1.943
wiki_talk	0.01	2,394,385	4,659,565	3.89	0.754	1.901	1.917
wiki_talk_LFR	0.01	2,394,385	3,297,120	2.75	0.753	2.062	2.277
wiki_talk	0.1	2,394,385	4,659,565	3.89	0.941	1.901	2.228
wiki_talk_LFR	0.1	2,394,385	3,296,124	2.75	0.940	2.062	2.340
wiki_talk	0.5	2,394,385	4,659,565	3.89	0.984	1.901	3.064

Table S8: Properties of the empirical networks and their LFR model graphs for Leiden-modularity clusterings. Parameters: μ stands for the mixing parameter, and τ_1 and τ_2 are the *estimated* power-law exponents for the degree and the community size distributions respectively.

network	nodes	edges	average degree	μ	τ_1	τ_2
open_citations	75,025,194	1,363,303,678	36.34	0.129	2.974	2.697
open_citations_LFR	3,000,000	55,128,496	36.75	0.129	2.978	2.707
CEN	13,989,436	92,051,051	13.16	0.180	2.618	1.255
CEN_LFR	3,000,000	20,821,202	13.88	0.180	2.620	1.489
cit_patents	3,774,768	16,518,947	8.75	0.114	4.017	2.365
cit_patents_LFR	3,774,768	15,648,081	8.29	0.114	4.000	2.361
cit_hepph	34,546	420,877	24.37	0.155	3.631	1.305
cit_hepph_LFR	34,546	431,138	24.96	0.155	3.632	1.525
wiki_topcats	1,791,489	25,444,207	28.41	0.199	2.430	3.961
wiki_topcats_LFR	1,791,489	23,581,074	26.33	0.200	2.454	3.947
wiki_talk	2,394,385	4,659,565	3.89	0.115	1.901	2.074
wiki_talk_LFR	2,394,385	3,278,574	2.74	0.114	2.082	2.074

Table S9: Percentage of LFR ground-truth clusters that are disconnected. The LFR graphs are generated based on the Leiden-modularity and Leiden-CPM (with five resolution values) clusterings for each of the six empirical networks. "N.A." means the statistic is unavailable because the LFR network for that condition could not be created.

		Leiden-CPM				
	Leiden-mod	0.0001	0.001	0.01	0.1	0.50
open_citations	0	0	0	0	0	0.001
CEN	0	0	0	0.04	90.81	100
cit_hepph	0	0	0	0	0	0
cit_patents	0	0	0	0	0.002	14.69
wiki_talk	70.72	66.36	87.03	100	100	N.A.
wiki_topcats	0	0	0	0	0.013	N.A.

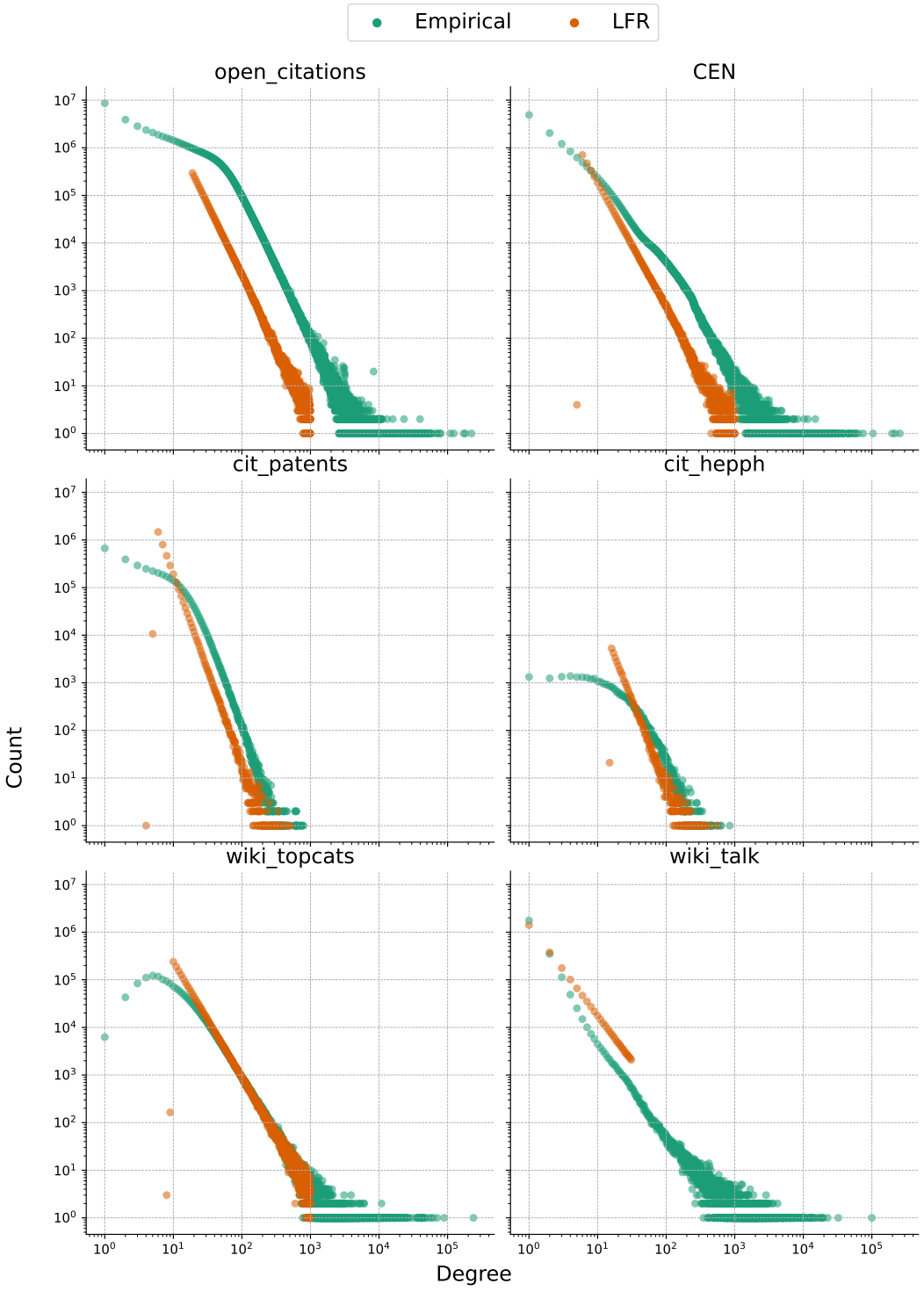


Figure S2: Comparison between the degree distribution of empirical and LFR networks using CPM with $r = 0.01$. The LFR networks are produced to emulate the characteristics of their corresponding real network. For the CEN and the OpenCitations network, the number of nodes of the LFR network is 3,000,000, and for the other networks it exactly matches the number of nodes in its corresponding empirical network. The clustering method for the empirical networks was CPM with resolution parameter 0.01. The axes are shown in log scale.

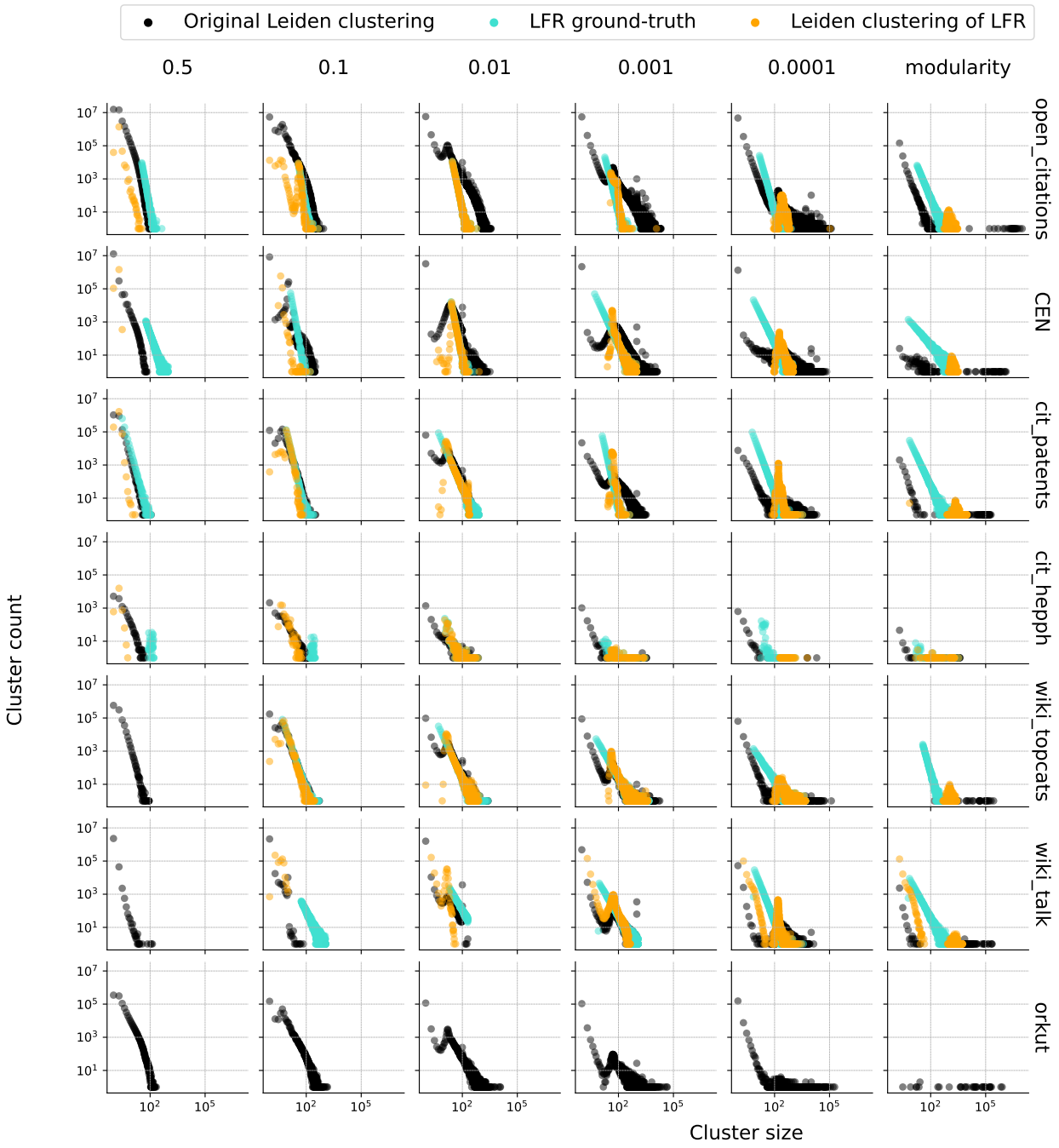


Figure S3: Cluster size distribution for all networks. The LFR ground-truth communities (shown in light green) are generated according to the parameters estimated from the Leiden clustering of the empirical network (shown in black), and then the LFR network is re-clustered using Leiden with the same resolution value (in orange). We were not able to generate an LFR graph for the Orkut network. The axes are shown in log scale.

S3.3 Additional Results for Experiments with Synthetic Networks

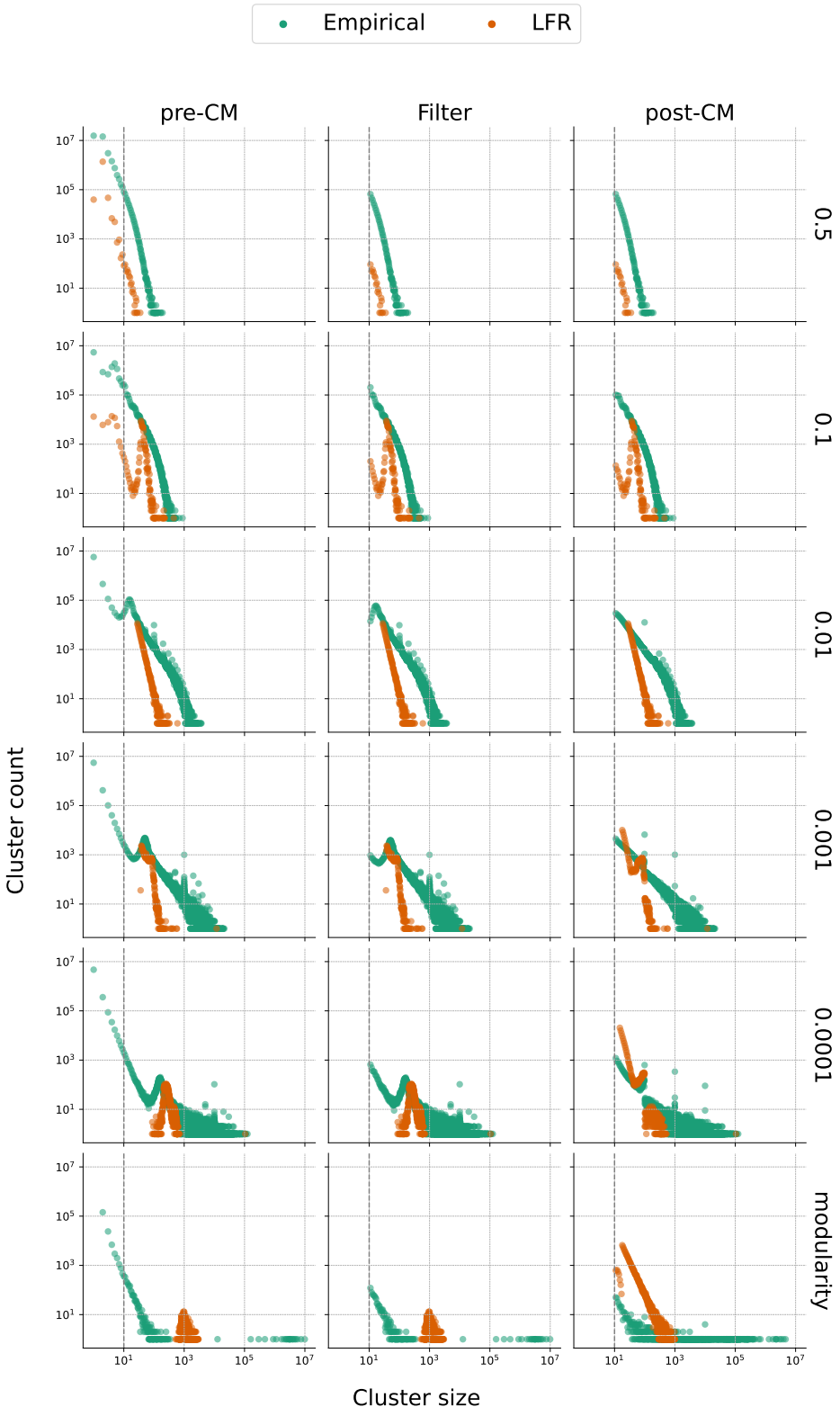


Figure S4: Impact of CM-processing on cluster size distributions of Leiden clusterings of empirical and LFR networks for the open_citations network. The empirical network has 75,025,194 nodes with an average degree of 36.35. Each row represents a different Leiden clustering method, with CPM-optimization for the top 5 rows and modularity-optimization for the bottom row. Each column represents a stage in the CM pipeline. The panels show the cluster size distributions in each step of running CM for the empirical network and its corresponding LFR graph. The axes are shown in log scale.

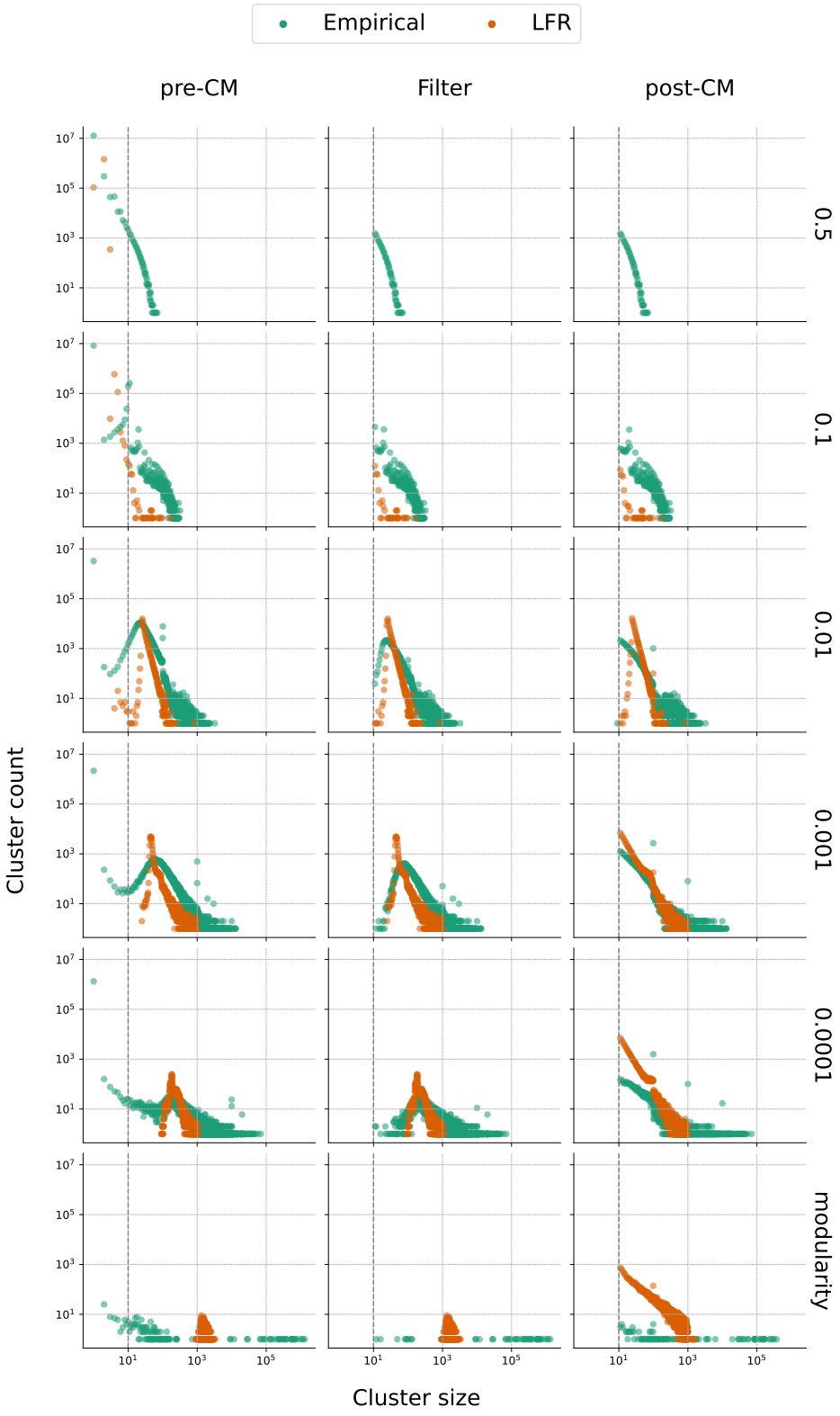


Figure S5: Impact of CM-processing on cluster size distributions of Leiden clusterings of empirical and LFR networks for the CEN network. The empirical network has 13,989,436 nodes with an average degree of 13.16. Each row represents a different Leiden clustering method, with CPM-optimization for the top 5 rows and modularity-optimization for the bottom row. Each column represents a stage in the CM pipeline. The panels show the cluster size distributions in each step of running CM for the empirical network and its corresponding LFR graph. The axes are shown in log scale.

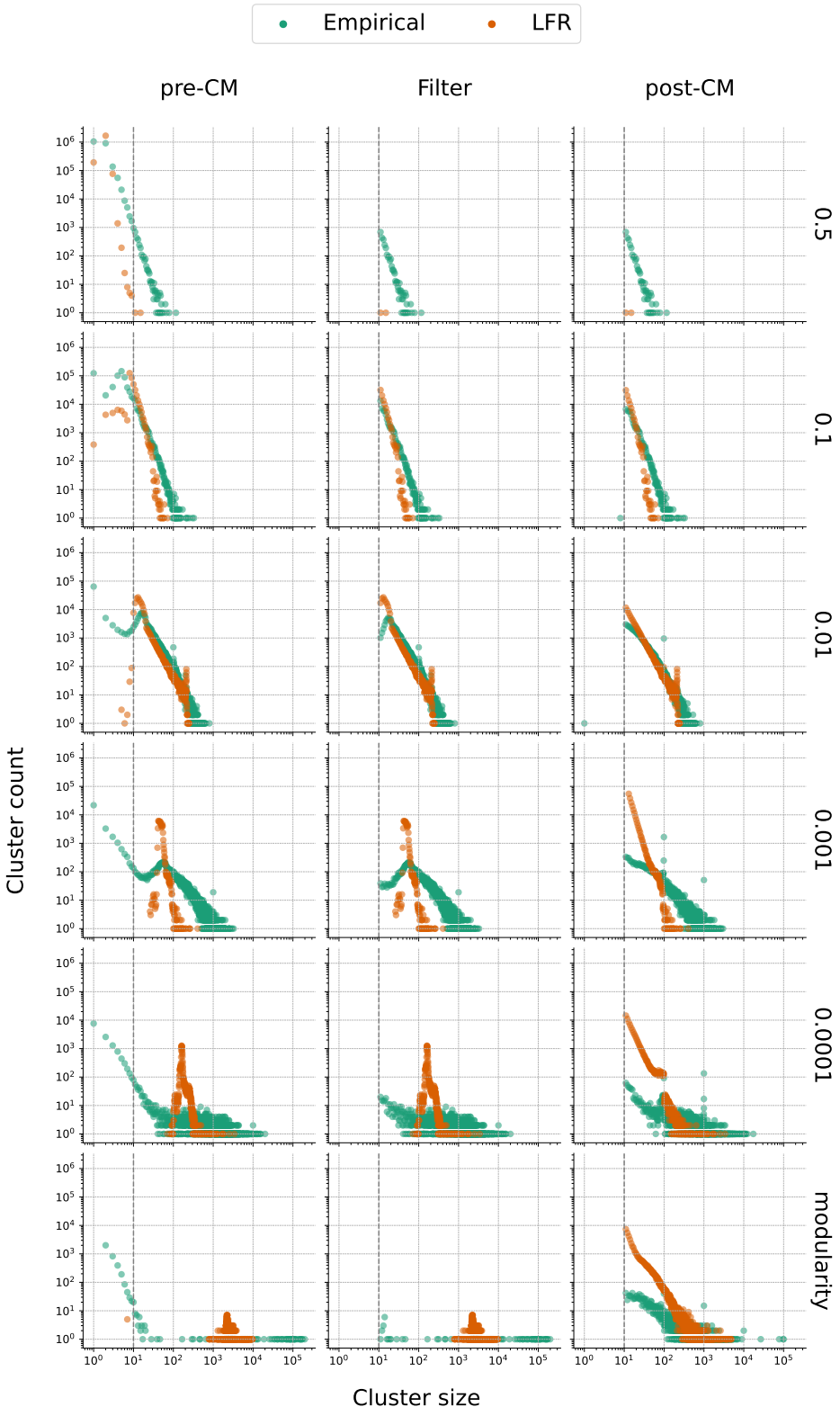


Figure S6: Impact of CM-processing on cluster size distributions of Leiden clusterings of empirical and LFR networks for the cit_patents network. The empirical network has 3,774,768 nodes with an average degree of 8.75. Each row represents a different Leiden clustering method, with CPM-optimization for the top 5 rows and modularity-optimization for the bottom row. Each column represents a stage in the CM pipeline. The panels show the cluster size distributions in each step of running CM for the empirical network and its corresponding LFR graph. The axes are shown in log scale.

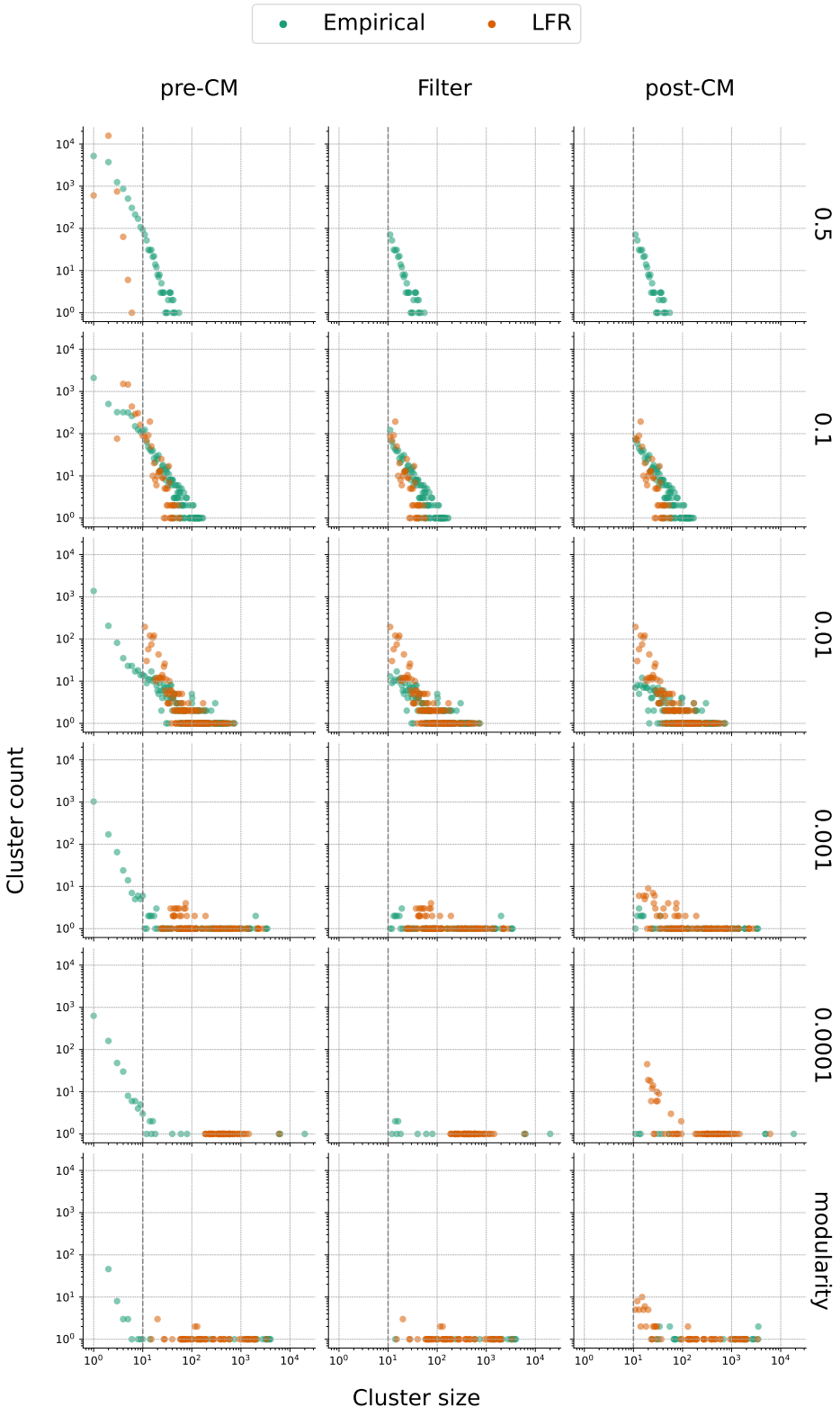


Figure S7: Impact of CM-processing on cluster size distributions of Leiden clusterings of empirical and LFR networks for the cit_hepph network. The empirical network has 34,546 nodes with an average degree of 24.37. Each row represents a different Leiden clustering method, with CPM-optimization for the top 5 rows and modularity-optimization for the bottom row. Each column represents a stage in the CM pipeline. The panels show the cluster size distributions in each step of running CM for the empirical network and its corresponding LFR graph. The axes are shown in log scale.

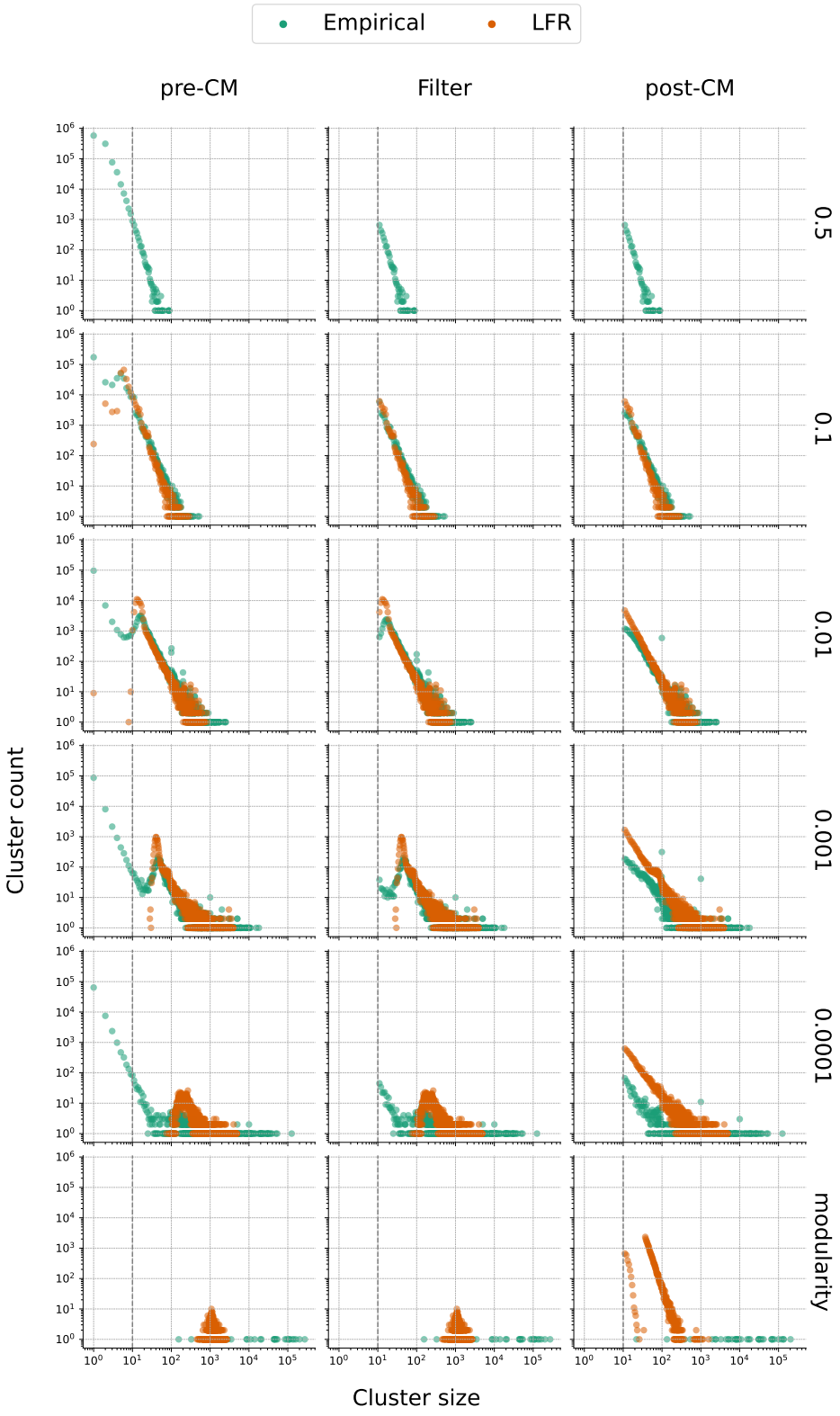


Figure S8: Impact of CM-processing on cluster size distributions of Leiden clusterings of empirical and LFR networks for the wiki_topcats network. The empirical network has 1,791,489 nodes with an average degree of 28.41. Each row represents a different Leiden clustering method, with CPM-optimization for the top 5 rows and modularity-optimization for the bottom row. Each column represents a stage in the CM pipeline. The panels show the cluster size distributions in each step of running CM for the empirical network and its corresponding LFR graph. For this network, we were not able to generate a corresponding LFR graph for $r = 0.5$. The axes are shown in log scale.

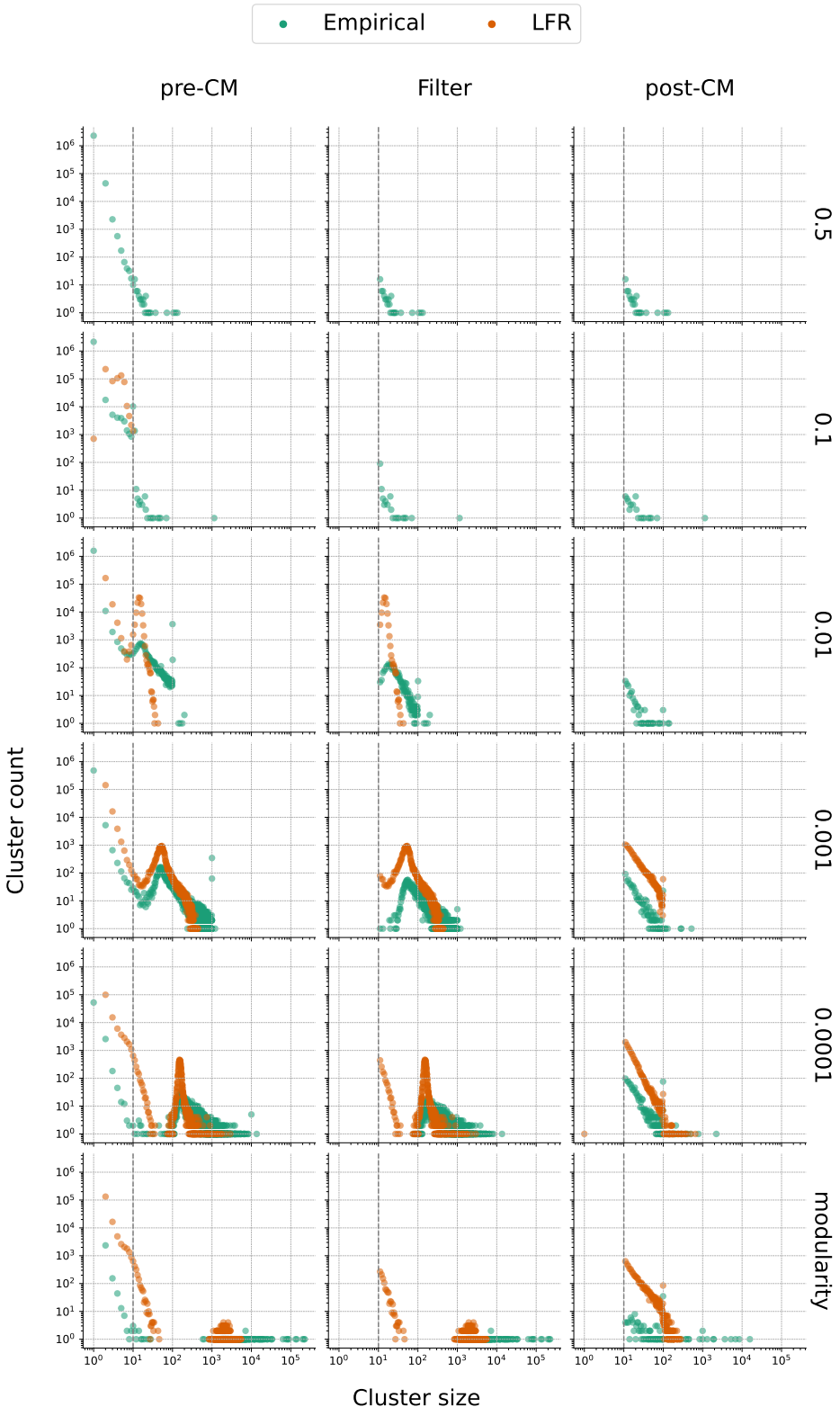


Figure S9: Impact of CM-processing on cluster size distributions of Leiden clusterings of empirical and LFR networks for the wiki_talk network. The empirical network has 2,394,385 nodes with an average degree of 3.89. Each row represents a different Leiden clustering method, with CPM-optimization for the top 5 rows and modularity-optimization for the bottom row. Each column represents a stage in the CM pipeline. The panels show the cluster size distributions in each step of running CM for the empirical network and its corresponding LFR graph. For this network, we were not able to generate a corresponding LFR graph for $r = 0.5$. The axes are shown in log scale.

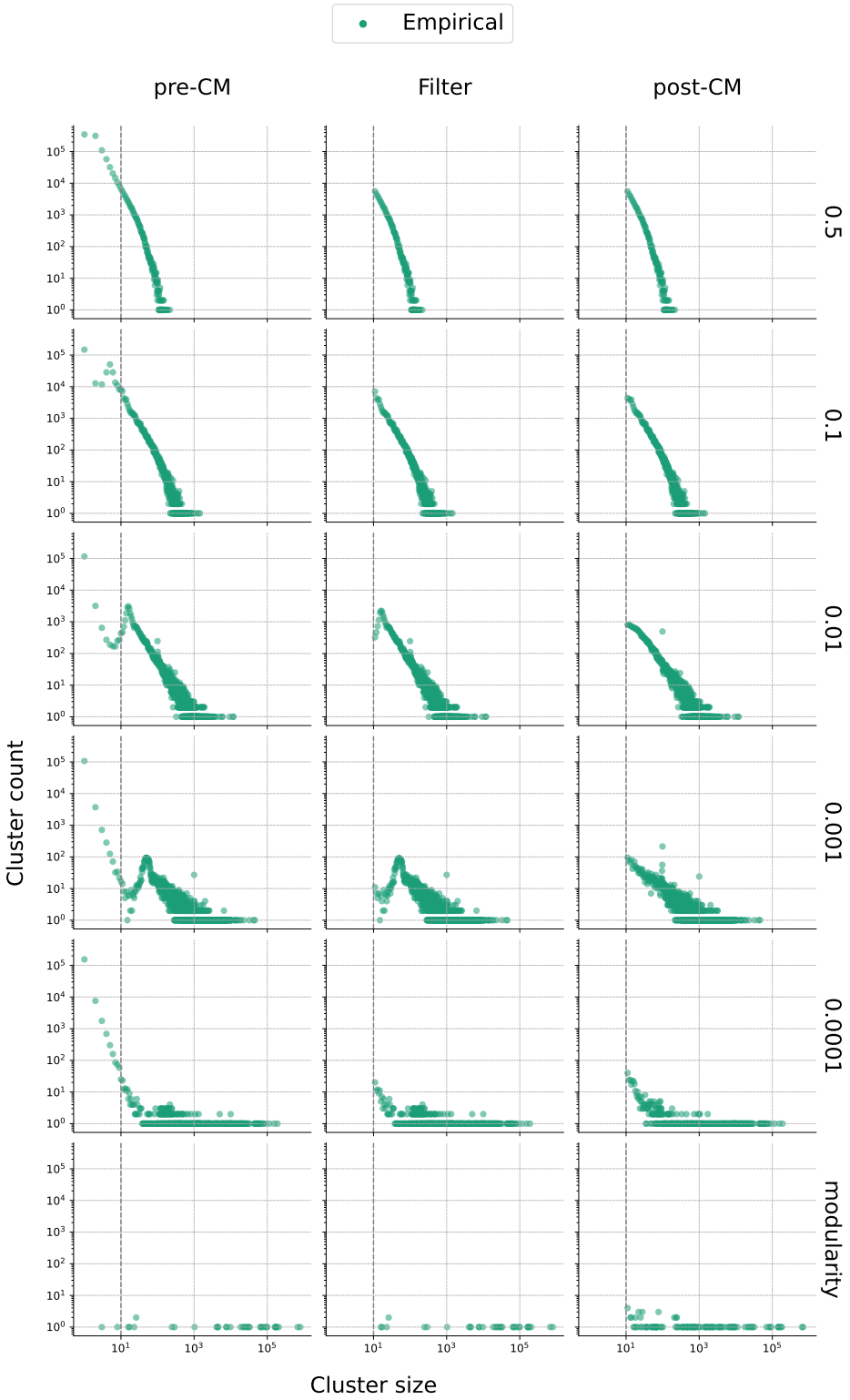


Figure S10: Impact of CM-processing on cluster size distributions of Leiden clusterings of the empirical Orkut network. The empirical network has 3,072,441 nodes with an average degree of 76.28. Each row represents a different Leiden clustering method, with CPM-optimization for the top 5 rows and modularity-optimization for the bottom row. Each column represents a stage in the CM pipeline. The panels show the cluster size distributions in each step of running CM for the empirical network. For this network, we were not able to generate a corresponding LFR graph in any condition. The axes are shown in log scale.

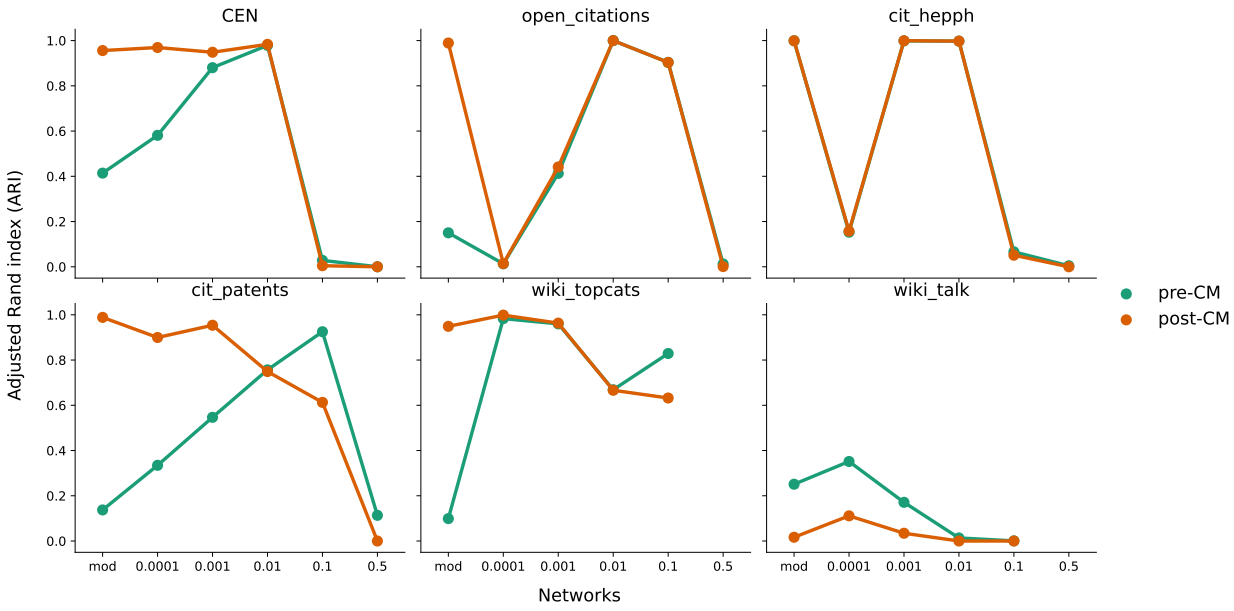


Figure S11: Comparison between ARI (Adjusted Rand Index) accuracy of pre-CM and post-CM clusterings of synthetic networks. The panels show accuracy measured in terms of adjusted Rand index (ARI) with respect to the LFR ground-truth communities. Each condition on the x-axis corresponds to a *different* LFR network, generated based on Leiden-modularity or Leiden-CPM with that specific resolution parameter. In total, there are 34 different LFR networks.