PLOS COMPLEX SYSTEMS

RESEARCH ARTICLE

Well-connectedness and community detection

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*], **Tandy Warnow**[1*]

1 Siebel School of Computing and Data Science, University of Illinois Urbana-Champaign, Urbana, Illinois, United States of America, **2** NTT DATA, McLean, Virginia, United States of America, **3** Insper Institute, São Paulo, Brazil

☯ These authors contributed equally to this work.
* chackoge@illinois.edu (GC); warnow@illinois.edu (TW)

Citation: Park M, Tabatabaee Y, Ramavarapu V, Liu B, Pailodi VK, Ramachandran R, et al. (2024) Well-connectedness and community detection. PLOS Complex Syst 1(3): e0000009. https://doi.org/10.1371/journal.pcsy.0000009

Editor: Réka Albert, Pennsylvania State University, UNITED STATES OF AMERICA

Received: January 30, 2024

Accepted: July 22, 2024

Published: November 5, 2024

Data Availability Statement: Data used in this study are available in the Illinois Data Bank, at https://doi.org/10.13012/B2IDB-6271968_V1 [66].

Funding: GC and TW received a grant from Insper-Illinois Collaboration (https://cs.illinois.edu/research/insper). GC and TW received a grant from the US National Science Foundation, number 2402559. The funders 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

Community detection methods help reveal the meso-scale structure of complex networks. Integral to detecting communities is the expectation that communities in a network are edge-dense and "well-connected". Surprisingly, we find that five different community detection methods–the Leiden algorithm optimizing the Constant Potts Model, the Leiden algorithm optimizing modularity, Infomap, Markov Cluster (MCL), and Iterative k-core (IKC)–identify communities that fail even a mild requirement for well-connectedness. To address this issue, we have developed the Connectivity Modifier (CM), which iteratively removes small edge cuts and re-clusters until communities are well-connected according to a user-specified criterion. We tested CM on real-world networks ranging in size from approximately 35,000 to 75,000,000 nodes. Post-processing of the output of community detection methods by CM resulted in a reduction in node coverage. Results on synthetic networks show that the CM algorithm generally maintains or improves accuracy in recovering true communities. This study underscores the importance of network clusterability–the fraction of a network that exhibits community structure–and the need for more models of community structure where networks contain nodes that are not assigned to communities. In summary, we address well-connectedness as an important aspect of clustering and present a scalable open-source tool for well-connected clusters.

Author summary

Community detection—a term interchangeably used with clustering—is used in network analysis. An expectation is that communities or clusters should be dense and well-connected. However, density is separable from well-connectedness, as clusters may be dense without being well-connected. Our study demonstrates that several clustering algorithms generate clusters that are not well-connected according to a mild standard we impose. To address this issue, we developed the Connectivity Modifier (CM), a tool to allow users to

specify a threshold for well-connectedness and enforce it in the output of multiple community detection methods.

Introduction

Graph clustering, also known as community detection, is integral to network analysis and has many applications [1, 2]. In graph clustering, the vertices of an input network are partitioned into disjoint sets with the object of having each such set (community or cluster) exhibit greater internal cohesion [3–5].

Approaches to defining communities in graphs are often based on finding subsets of nodes that are more densely connected to each other in comparison to the background density of the network [6, 7]. A related expectation is that a community should be well-connected; to rephrase, its minimum edge cut, a set of edges whose deletion splits the community into two parts, should not be small [8, 9]. While the two concepts are related, it is possible for a cluster to be dense yet not well-connected. An illustrative example is a community comprising two cliques connected by a single edge, inside a relatively sparse network [10]. Thus, density is a desirable property of communities, but does not ensure well-connectedness by itself. In other words, *density and well-connectedness are expected but separable properties of communities*.

Preceding work also indicates support for using well-connectedness as a cluster quality criterion. Of several cluster quality criteria, cut-conductance [11, 12], measures how well separated a cluster is from its complement, and set-conductance measures how internally well-connected a cluster is [9, 13]. Accordingly, high quality clusters should demonstrate low cut-conductance values and high set-conductance values. Cluster quality is also examined using mixing times, and high quality clusters are expected to have fast mixing times [14, 15]. Thus, set-conductance is directly related to how well-connected a cluster is, while mixing times are indirectly related.

It is known that disconnected clusters are sometimes produced by clustering methods. For example, the Louvain algorithm for modularity optimization [16] was noted to produce some poorly connected to disconnected clusters and this problem was addressed in the Leiden [8] algorithm, which does guarantee connected clusters. However, little seems to be reported about the degree to which clusters produced by popular clustering methods are well-connected.

To address well-connectedness from an empirical perspective, we evaluate five different clustering methods on seven real-world networks that range in size from $\sim$35 thousand to $\sim$75 million nodes. We explore the Leiden algorithm [8], optimizing under either modularity or the Constant Potts Model (CPM), Infomap [17], Markov clustering (MCL) [18], and the Iterative k-core (IKC) method [19].

Our goal was to identify, using a very mild threshold, the extent to which a selection of clustering methods produce clusters that are not well-connected. Therefore, we picked a very slow growing function, $\log_{10} n$, so that if a cluster with n nodes had an edge cut of at most $\log_{10} n$ it would not be considered well-connected. For example, when $n = 500$, we would require that the min cut contain at least three edges. Because this threshold is very modest, failure to meet it is evidence of insufficient internal cohesion. However, meeting the threshold may not be particularly strong evidence of good cohesion. We find that each method produces clusters that are not well-connected according to our mild criterion, with the extent varying according to the method and network, and some methods even produce disconnected clusters.

As a remediating tool, we developed the Connectivity Modifier (CM) [20–23] to ensure well-connected clusters. The current version [23] provides support for Leiden, IKC, MCL, and Infomap and is being extended to integrate other clustering methods.

We have previously described CM [22], and provided evidence that Leiden (optimizing either modularity or CPM), IKC, Infomap, and MCL all produced clusters that were poorly connected, according to this mild requirement for the size of the minimum cut; further, we explored the impact of CM on Leiden and IKC on both real-world and synthetic networks. In this study we present an evaluation of the impact of CM on community finding by the Infomap and MCL methods, and provide a more detailed investigation into the impact of CM on clustering outputs from IKC and Leiden optimizing modularity or CPM. Finally, we share observations using synthetic networks.

Results and discussion

Results shown here are based on seven real-world networks (Materials and methods), ranging in size from 34,546 nodes to 75,025,194 nodes. We are especially interested in clustering methods that can be used on large citation networks [8, 19], and our choice of networks reflects this interest. We generated 34 synthetic networks based on these real-world networks using the LFR software [24] with parameters derived from Leiden clusterings of these networks. For comparison, we also generated 20 nPSO [25] networks with 10,000 vertices and varying average degree and temperature (which controls for the clustering coefficient) as well as 50 Erdős-Rényi graphs with 10,000 vertices and different values for p (the probability of an edge between two nodes). Sections B and C, Tables A–G, and Fig J in S1 Appendix contain additional results for the real-world networks and Sections D–G in S1 Appendix contain additional information on the synthetic networks.

Initial observations

Examining minimum cut sizes. In an initial exploration of a 75,025,194-node citation network (Open Citations, Materials and Methods), we computed the min cut of all clusters generated using the Leiden algorithm optimizing either CPM or modularity (Fig 1). For CPM, we used five different resolution parameter values (r, drawn from 0.5, 0.1, 0.01, 0.001, 0.0001). Designating very small clusters as not being of practical interest, unless otherwise specified, we report node coverage throughout this manuscript as the percentage of nodes in clusters of size at least 11.

For Leiden-CPM clusterings (Fig 1), we see that as the resolution value is decreased, (i) node coverage increases from 6% to 99%, (ii) the frequency of small mincuts increases, and (iii) cluster sizes increase. With respect to these measurements, clustering under modularity is most similar to clustering under CPM at the lowest resolution value used. Strikingly, optimizing under modularity had the highest node coverage (99.4%) but 98.7% of its 2,184 clusters had a minimum edge cut size of 1, that is, they could be split by removing a single edge. Additionally, where node coverage is high, clusters tend to be larger and fewer. We also observe that as the resolution value decreases from 0.5 down to 0.0001, the maximum size of any cluster increases for CPM-optimization.

These data illustrate a tradeoff that users make with the Leiden algorithm optimizing CPM between small clusters, lower node coverage, and a reduced tendency for clusters to have small min cuts (achieved with larger resolution values for Leiden-CPM) versus larger clusters, higher node coverage, and higher frequency of clusters with small min cuts (CPM-optimization with small resolution values or modularity-optimization).

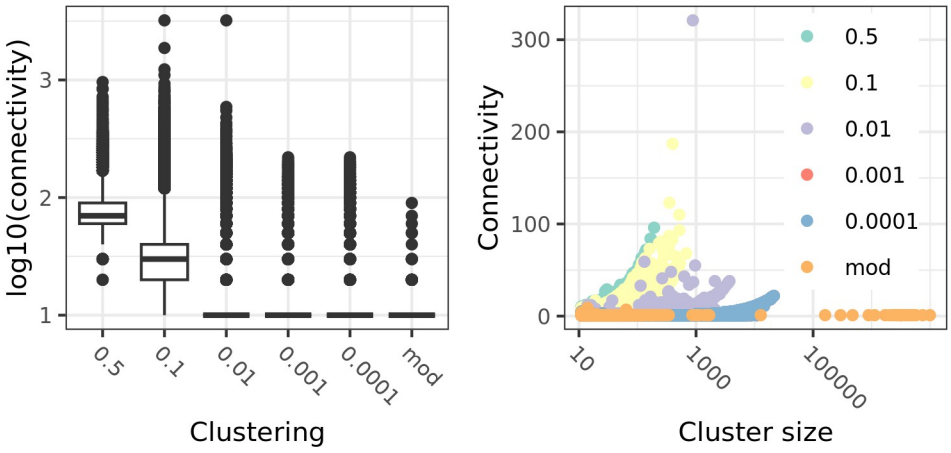

Fig 1. Minimum edge cut sizes and node coverage of Leiden clusters (using CPM and modularity) on the Open Citations network. Each subfigure contains results from clustering using Leiden under CPM optimization with five different resolution values and modularity. *Left Panel*: Min cut sizes found across all clusters for a given clustering. Node coverage, the percentage of nodes in the input network found in clusters of size at least 11 is (from left to right) 6.0%, 43.8%, 88.9%, 90.5%, 91.9%, and 99.4%. *Right Panel*: Min cut sizes as a function of the cluster size. Note that except for CPM-clustering with very large resolution values ($r = 0.1$ or $r = 0.5$), most clusters have edge cuts of size 1, and that CPM-clustering with the high resolution values and modularity clustering have low node coverage (below 50%). The size of the minimum edge cut increases with cluster size for CPM-optimization, but this is not the case for modularity-optimization.

https://doi.org/10.1371/journal.pcsy.0000009.g001

In addition to the Open Citations network, we clustered six more networks, ranging in size from 34,546 nodes to 13,989,436 nodes (Materials and methods), using Leiden optimizing either modularity or the Constant Potts Model, IKC, Infomap, and MCL. Only Leiden and IKC ran to completion on all seven networks. Infomap ran on all but the Open Citations Network. MCL completed only on the smallest network (cit_hepph) we analyzed. IKC, the most conservative of the methods, did not return any clusters from the sparse wiki_talk network.

Interestingly, while neither Leiden nor IKC generated clusters that were disconnected, 7.2% of MCL's clusters of size at least 11 were disconnected on the cit_hepph network and the percentage of disconnected clusters of size at least 11 for Infomap ranged from 0% to 75% (Table H in S1 Appendix). These observations describe the proportion of clusters that are not even connected, with the extent dependent on clustering method and network.

Defining well-connected clusters. In a second exploratory experiment, we explored ways of defining "well-connected" or its negation, "poorly-connected". While the existence of a small edge cut for a cluster intuitively signals poorly-connected, there are different ways of formalizing how small the cut must be for the cluster to be considered poorly connected. For the purposes of this study, we offer a definition that we use to evaluate clusters (Fig 2). Briefly, we consider functions $B(n)$, with the interpretation being that if a cluster with n nodes has an edge cut of size $B(n)$ or smaller, then the cluster will be considered poorly connected. We want this bound to be modest, so that we do not label clusters as poorly connected too readily. Thus, we want $B(n)$ to grow very slowly so that it serves as a mild bound. We also want $B(n) \geq 1$ for all n that are large enough for the cluster to be considered a potential community.

Traag et al. [8] proved that if X is an edge cut for a cluster C in a CPM-optimal clustering (with resolution parameter r), whose removal splits C into two sets A and B, then $|X| \geq r \times |A| \times |B|$. Thus, the function $a(r, A, B) = r \times |A| \times |B|$ is a lower bound on the size of the minimum cut for any cluster in a CPM-optimal clustering. This function is maximized when $|A| = |B|$ and minimized when $|A| = 1$ and $|B| = n - 1$, where C has n nodes. Therefore, the minimum

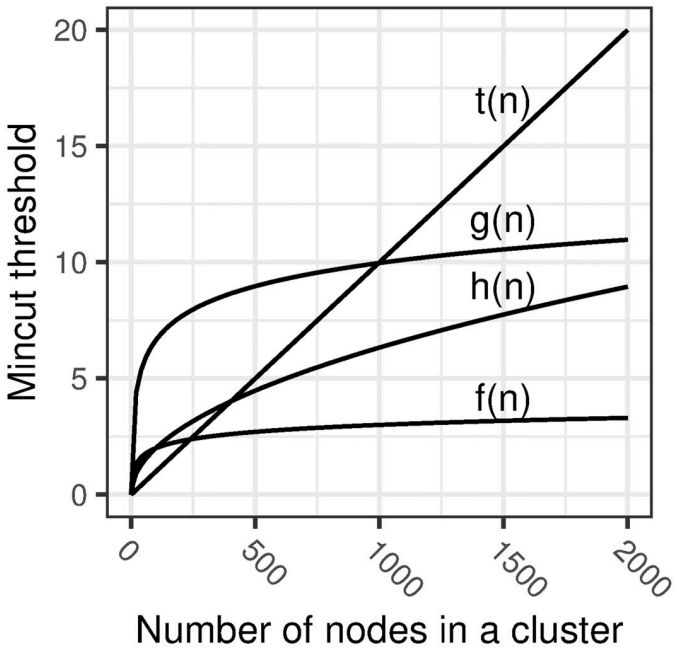

Fig 2. Comparison of the lower bounds on edge cut sizes for defining well-connected clusters. (x-axis: n, y-axis: function value.) Here we compare four functions; $t(n) = 0.01(n-1)$, $f(n) = \log_{10} n$, $g(n) = \log_2 n$, and $h(n) = \frac{\sqrt{n}}{5}$. By [8], $t(n)$ provides a guaranteed lower bound on the edge cut size that splits one node from the remaining nodes for a cluster with n nodes in an optimal CPM clustering, when $r = 0.01$. Also, $t(n)$ is the largest of the functions when $n > 1000$ and $f(n)$ is the smallest of the functions when $n > 239$. Thus, in general $t(n)$ is the strongest guarantee of the functions we compare when n is large, but the other functions provide somewhat stronger guarantees for the smaller values of n. In particular, 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 $t(n)$.

https://doi.org/10.1371/journal.pcsy.0000009.g002

cut for any cluster with n nodes has size at least $r \times (n-1)$. For example, when $r = 0.01$ and $n = 50$, then writing $t(n) = 0.01(n-1)$ yields the lower bound on the size of the minimum cut in a CPM-optimal clustering with resolution parameter $r = 0.01$. Note that $0 < t(50) < 1$, and so this bound only establishes that the minimum edge cut is at least 1, which therefore simply asserts that the cluster is connected. In other words, for resolution values such as $r = 0.01$, the lower bound given in [8] is not particularly significant, and the bound is even weaker when the resolution value is smaller.

We seek lower bounds on the size of the minimum edge cut that are larger than $t(n) = 0.01 \times (n-1)$ (the bound when $r = 0.01$) for small values of n but grow very slowly, and so do not exceed the $t(n)$ bound for large values of n.

We consider three functions: $f(n) = \log_{10} n$, $g(n) = \log_2 n$, and $h(n) = \frac{\sqrt{n}}{5}$; note that each of $f(n)$, $g(n)$ and $h(n)$ is strictly positive and increasing. The comparison between these functions in the range of cluster sizes $1 \leq n \leq 2000$ is shown in Fig 2. As desired, for large enough n, $t(n)$ 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 use $f(n) = \log_{10}(n)$ to define when a cluster of n nodes is well-connected: if the min cut size for the cluster is strictly greater than $\log_{10}(n)$ then we consider it well-connected, and otherwise we say that the cluster is poorly connected. Note that if we had

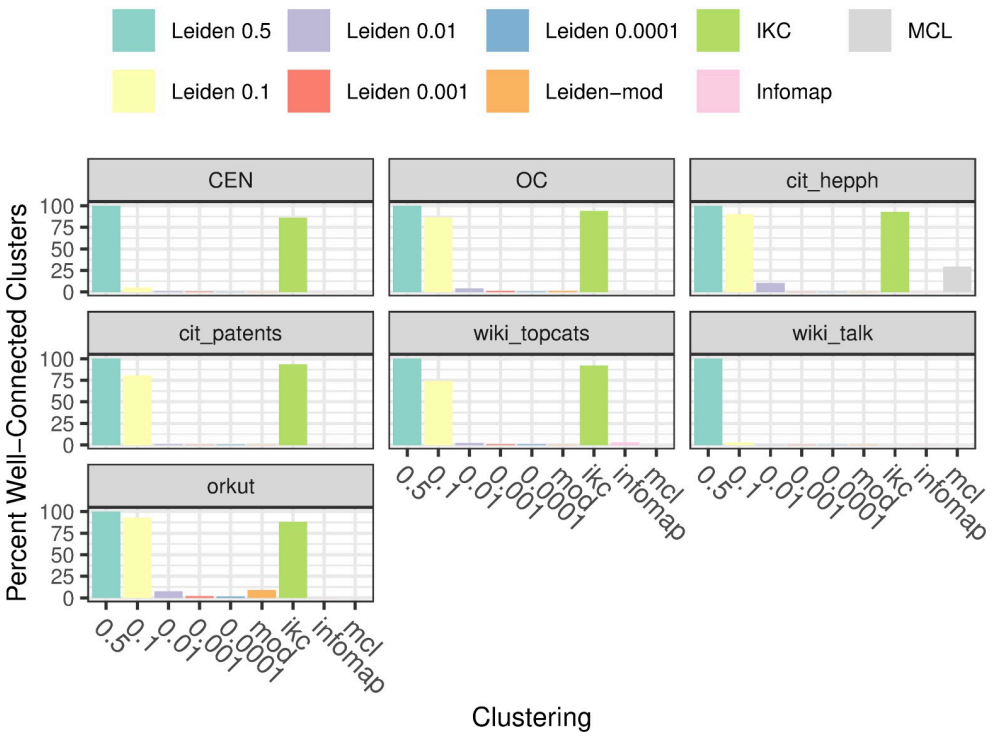

Fig 3. Percentage of clusters that are well-connected in seven real-world networks ranging from 35K to 75M nodes. Results are shown for Leiden optimizing CPM at different resolution values, Leiden optimizing modularity, IKC, Infomap, and MCL. IKC did not return any clusters from the wiki_talk network. Infomap completed on all but Open Citations. MCL completed only on cit_hepph. Leiden and IKC ran to completion on all seven networks. Percentage is shown for clusters of size at least 11.

https://doi.org/10.1371/journal.pcsy.0000009.g003

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, $f(n) = \log_{10} n$ *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.*

Frequency of well-connected clusters. Our third exploratory experiment evaluated the frequency with which each of the clustering methods produced well-connected clusters on the seven networks. As previously noted, Leiden and IKC completed on all networks, Infomap failed on the largest network, and MCL returned output only from the smallest network (cit_hepph) we analyzed.

For those methods that completed on a given network, we calculated the percentage of the clusters of size at least 11 that were well-connected, according to the threshold $f(n) = \log_{10} n$. Although all clustering methods generated some clusters that were not well-connected, the following trends can be observed (Fig 3. When using Leiden-CPM, the choice of resolution value had a large impact on the frequency of well-connected clusters, with only the two largest resolution factor values, 0.5 and 0.1, producing mostly well-connected clusters. IKC had a high frequency of well-connected clusters, almost as high as Leiden-CPM using $r = 0.5$, and higher than Leiden-CPM with $r = 0.1$. The remaining clustering methods had very few well-connected clusters.

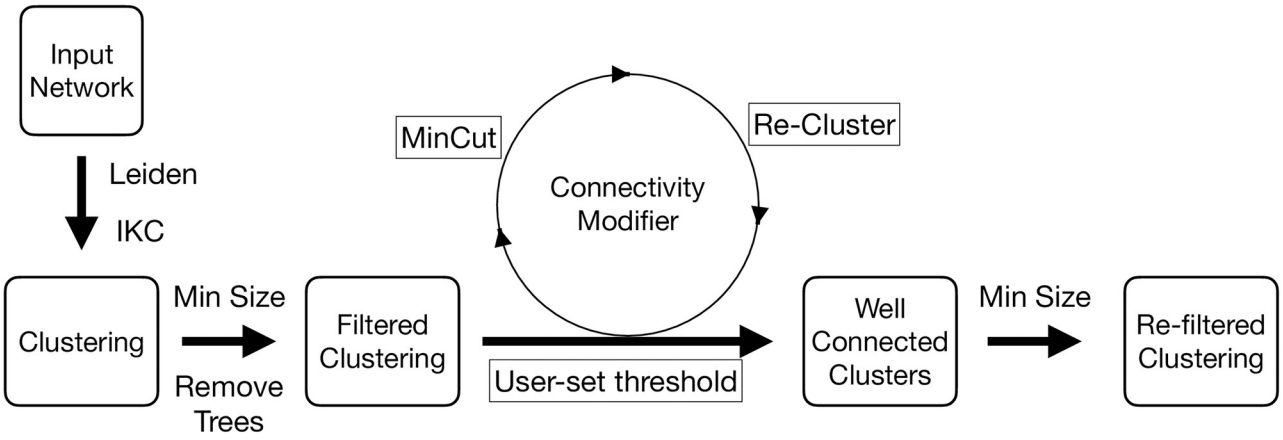

Fig 4. Connectivity Modifier Pipeline Schematic. The four-stage pipeline depends on user-specified algorithmic parameters: B (default 11), the minimum allowed size of a cluster, and $f(n)$ (default $\log_{10}(n)$), a bound on the minimum edge cut size for a cluster with n nodes, and clustering method. *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 Connectivity Modifier (CM) is applied to each cluster, removing edge cuts of sizes at most $f(n)$, reclustering, and recursing on clusters. *Stage 4*: clusters are post-processed by removing those of size less than B.

https://doi.org/10.1371/journal.pcsy.0000009.g004

The connectivity modifier pipeline

To ensure that clusters are well-connected, we designed the Connectivity Modifier (CM) [20, 21] pipeline, which post-processes outputs of clustering methods in order to ensure that all returned clusters are well-connected and not too small, both according to specified thresholds provided by the user. CM can currently be used with four different clustering methods: Leiden (optimizing modularity or CPM), IKC, Infomap, and MCL. A parallelized version of CM has since been developed [23].

Here we describe the CM pipeline (Fig 4 and Section A in S1 Appendix). The algorithm depends on three parameters that the user may specify (or use the default settings): (i) the clustering method, (ii) $f(n)$ (the lower bound on the size of a min cut), and (iii) B, the minimum allowed size of a cluster. In our study we explored the pipeline with the default settings as follows: $f(n) = \log_{10}(n)$ and $B = 11$. In order to speed up the analysis, the pipeline includes a pre-processing (filtering) step, which discards clusters that are trees (and so not well-connected) or of size less than B. Each cluster is then processed by CM, independently. First, the cluster is checked using VieCut [26] to see if it contains an edge cut of size at most $f(n)$; if so, the edge cut is removed, thus producing two subsets of nodes, which are each then re-clustered using the specified clustering method. This process repeats until the current iteration produces no change. At the end, all clusters are well-connected, but some may be below the allowed threshold: those are removed in a final post-processing step.

Effect of the CM pipeline on clustered real-world networks

Having demonstrated that all the clustering methods we study return poorly connected clusters under some conditions, we now examine the impact of using the CM pipeline. Because our interest is mainly in methods that can scale to large networks (which MCL did not) and that do not tend to produce disconnected clusters (which Infomap does), our main focus is on the impact of CM on Leiden and IKC. However, we also provide an examination of the impact of CM on MCL and Infomap, although our discussion for these two methods is more limited.

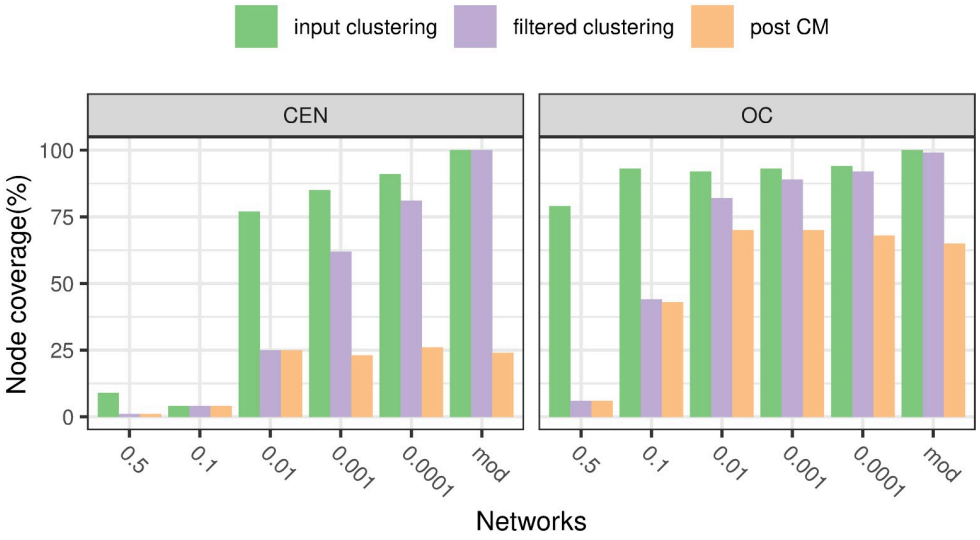

Fig 5. Reduction in node coverage after CM treatment of Leiden clusters. The left panel shows results for the CEN and the right panel shows Open Citations results. In each subfigure, green bars denote node coverage in clusters of size at least 2 in the input clustering, purple bars denote node coverage after trees and clusters of size at most 10 are removed, and orange bars denote node coverage in the final output by CM. The Open Citations and CEN networks were clustered using the Leiden algorithm optimizing either modularity or CPM, with five different resolution values.

https://doi.org/10.1371/journal.pcsy.0000009.g005

Impact of the CM pipeline on Leiden optimizing modularity or CPM. We examined two aspects of the impact of CM on Leiden clustering, optimizing both modularity and CPM using five different resolution parameter values. The first aspect is how node coverage drops as a result of the initial reduction (removing small clusters and trees) and the second is the drop after the full pipeline is traversed (post-CM). Our initial experiment (Fig 5) examined results on the two largest networks, allowing the resolution parameter to vary for Leiden-CPM. These results show initial node coverage according to whether modularity or CPM is optimized, and in the case of CPM, the resolution parameter. For both these large networks, node coverage drops after CM-processing, with maximum post-CM node coverage about 25% for the CEN and 69% for Open Citations.

Using Leiden for CPM-optimization with the largest resolution value ($r = 0.5$) produced the smallest node coverage, both before CM-processing and after, while using the smallest resolution value ($r = 0.0001$) achieved the largest node coverage before CM-processing. The three smallest resolution values ($r = 0.01$, 0.001, and 0.0001) all produced effectively the largest node coverage after CM-processing on these two networks. Optimizing modularity produced slightly higher pre-CM node coverage than all CPM-based clusterings, but slightly lower post-CM node coverage. The impact of pre-processing (i.e., Stage 2) is large for the two larger resolution values, but decreases with resolution value, while the third stage, where the Connectivity Modifier is applied to poorly connected clusters, has only a small impact on node coverage when the resolution parameter value is large, but the impact increases as the resolution parameter value decreases.

We then examined node coverage both pre-CM and post-CM for Leiden optimizing modularity or optimizing CPM with $r = 0.01$ on the remaining networks (Table 1); results for other settings of the resolution parameter for Leiden-CPM are shown in Table I in S1 Appendix, with a detailed stage-by-stage analysis in Table J in S1 Appendix. While pre-CM Leiden-CPM has relatively low node coverage on the wiki_talk network, it has high node coverage (at least

Table 1. Node coverage (NC) before and after CM-processing.

method	network	pre-CM NC $n \geq 2$	pre-CM NC $n \geq 11$	post-CM NC $n \geq 11$
Leiden-Mod	CEN	1.000	1.000	0.240
Leiden-Mod	OC	1.000	0.994	0.645
Leiden-Mod	cit_hepph	1.000	0.995	0.836
Leiden-Mod	cit_patents	1.000	0.997	0.373
Leiden-Mod	wiki_talk	1.000	0.998	0.025
Leiden-Mod	wiki_topcats	1.000	1.000	0.738
Leiden-Mod	orkut	1.000	1.000	0.907
Leiden-CPM(0.01)	CEN	0.769	0.766	0.132
Leiden-CPM(0.01)	OC	0.924	0.889	0.640
Leiden-CPM(0.01)	cit_hepph	0.960	0.915	0.830
Leiden-CPM(0.01)	cit_patents	0.983	0.955	0.547
Leiden-CPM(0.01)	wiki_talk	0.338	0.319	0.002
Leiden-CPM(0.01)	wiki_topcats	0.946	0.914	0.556
Leiden-CPM(0.01)	orkut	0.962	0.955	0.819
IKC	CEN	0.038	0.038	0.038
IKC	OC	0.236	0.236	0.201
IKC	cit_hepph	0.201	0.201	0.201
IKC	cit_patents	0.020	0.020	0.020
IKC	wiki_topcats	0.068	0.068	0.051
IKC	orkut	0.437	0.437	0.437
Infomap	CEN	0.952	0.950	0.245
Infomap	cit_hepph	1.000	0.996	0.824
Infomap	cit_patents	1.000	0.997	0.391
Infomap	wiki_talk	1.000	0.986	0.006
Infomap	wiki_topcats	1.000	0.989	0.762
Infomap	orkut	1.000	1.000	0.896
MCL	cit_hepph	0.975	0.690	0.617

Leiden-CPM(0.01): Leiden optimizing CPM using resolution parameter $r = 0.01$.

https://doi.org/10.1371/journal.pcsy.0000009.t001

76%, but most above 90%) on the other networks, even when restricted to clusters of size at least 11. Post-CM, however, all node coverages drop, and when restricted to clusters of size at least 11 the node coverage ranges from 0.2% to 83.0%. Although the highest final node coverage is achieved on the smallest network, the smallest node coverage is on wiki_talk, which is far from the largest network.

Detailed results for the impact of each stage of CM on Leiden clusterings of real-world networks are shown in Table J in S1 Appendix. Optimizing modularity or CPM with very small resolution values show very high node coverage, followed by at most a small reduction in node coverage as a result of Stage 2, and then a substantial drop in node coverage after Stage 3. Results for Leiden optimizing CPM for large resolution values (i.e., $r = 0.1$ or $r = 0.5$) show a different outcome: a significant drop in node coverage from Stage 2 (due to clusters being too small) and then little to no drop in node coverage from Stage 3. Thus, the impact of CM on node coverage depends on whether CPM or modularity is being optimized, and if CPM, then on the resolution value. Furthermore, the Leiden clusterings that have high initial node coverage reduce in coverage the most.

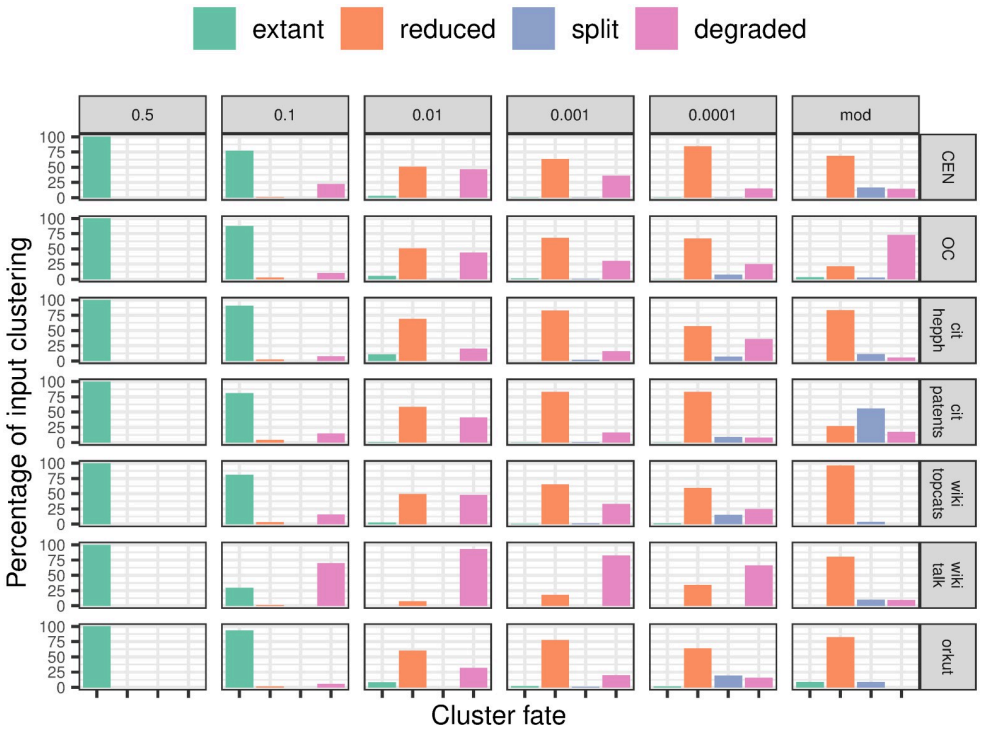

Fig 6. Cluster Fate for Leiden Clusters. The fate of Leiden clusters of size at least 11 from seven different networks as extant (not modified by CM), reduced (replaced by a smaller cluster), split (replaced by two or more smaller clusters), or degraded (replaced entirely by singletons) is indicated. 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.

https://doi.org/10.1371/journal.pcsy.0000009.g006

To provide a detailed insight into the impact of CM on clusterings produced using Leiden optimizing CPM or modularity, we examined each cluster to see whether it was modified at all by CM, and if it was, whether it was reduced in size but remained a single cluster, or whether it was split into two or more non-singleton clusters, or replaced entirely by singleton clusters (which is what happens when it is replaced by a collection of clusters that are all below the minimum size requirement, B). We refer to clusters that are not modified as *extant*, clusters that are made smaller but remain a single cluster as *reduced*, clusters that are split into at least two clusters as *split*, and the remaining clusters that are replaced by a collection of singletons as *degraded*.

The interpretation of these classifications is as follows. An extant cluster is one that was not modified by CM, and is regarded as valid. The reduced clusters are those that remain a single, but smaller, cluster after CM treatment, indicating that the set contained–as a proper subset–a single valid community. Those clusters that are split contain two or more valid clusters, suggesting a "resolution limit" effect reported for modularity in [27]. Finally, those clusters that are degraded, and so replaced by either a collection of singletons or clusters of size at most 10 are considered to not contain any valid communities.

As seen in Fig 6, when using Leiden optimizing CPM with resolution values $r \geq 0.1$, most produced clusters are extant. Hence, CPM optimization with resolution values at least 0.1 is

not particularly affected by CM, indicating that many of the clusters are well-connected and above the minimum threshold for size as otherwise, the clusters would be changed by CM. However, as seen in the previous figure, CPM optimization using these large resolution values also has small node coverage, indicating that this trend is likely due to the conservative nature of CPM clustering with large resolution values.

All cluster types described above are evident when running Leiden optimizing CPM with the smaller resolution values or optimizing modularity (Fig 6). In all networks, most of the clusters are either degraded or reduced. Interestingly, however, in all networks, modularity optimization produces some clusters that are split, and in all but two networks, CPM-optimization with the smallest resolution value also produces some clusters that are split (this is also true for some intermediate resolution values on some networks). Thus, this study suggests that both Leiden optimizing CPM using these smaller resolution values and Leiden optimizing modularity will, for some networks, produce clusters that contain two or more valid clusters, which appears similar to the resolution limit problem [27].

Impact of CM on IKC. Trends observed for IKC show the following trends. First, node coverage pre-CM is generally low for all networks, and the drop in node coverage as a result of CM-processing is very small (Table 1). These trends are consistent with IKC producing mostly well-connected clusters, as described earlier, and noting that IKC by design does not produce clusters of size less than 11, due to its default setting for the value of $k = 10$ so that each cluster is an l-core for some $l \geq k$. The trends for IKC resemble those for Leiden-CPM using $r = 0.5$.

Cluster fate for IKC clusters of size at least 11 show the following trends. Nearly all are extant (i.e., are well-connected), but there are also clusters that are split, meaning they are cut into two or more parts, and no clusters are reduced or degraded (Table 2). These trends are different from those observed for Leiden clusterings described in the previous section and likely reflect the conservative nature of recursive k-core extraction that is the basis of IKC.

Impact of CM on Infomap and MCL. Under the conditions of our study, Infomap completed on all but Open Citations while MCL only completed on one network (cit_hepph).

Table 2. Impact of CM: Cluster Fates for IKC, Infomap, and MCL on the real-world networks.

Method	Network	extant	reduced	split	degraded
IKC	CEN	85.9	0.0	14.1	0.0
IKC	OC	94.1	0.0	5.9	0.0
IKC	cit_hepph	92.9	0.0	7.1	0.0
IKC	cit_patents	93.5	0.0	6.4	0.0
IKC	wiki_topcats	91.8	0.0	8.2	0.0
IKC	orkut	88.0	0.0	12.0	0.0
MCL	cit_hepph	29.5	59.1	0.0	11.5
Infomap	CEN	28.9	43.3	2.7	25.1
Infomap	cit_hepph	88.2	7.4	2.9	1.5
Infomap	cit_patents	92.0	4.2	3.8	0.0
Infomap	wiki_topcats	33.5	35.7	0.5	30.2
Infomap	wiki_talk	32.6	1.9	0.0	65.5
Infomap	orkut	5.0	65.0	15.0	15.0

The fraction of clusters that are 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 shown as a percentage of the total number of clusters of size at least 11. Results not shown for a given pair of network and method indicate the method did not run or produce any clusters of size at least 11.

https://doi.org/10.1371/journal.pcsy.0000009.t002

Infomap has high initial node coverage (pre-CM) for all networks except cit_patents, and while the impact of CM-processing on node coverage depends on the network, for all networks, there is at least a moderate drop in node coverage (Table 1). For Infomap, the difference in node coverage based on clusters of size at least 2 or clusters of size at least 11 is small, so that the reduction in node coverage comes from finding and processing clusters with small edge cuts. This is consistent with the results shown in Fig 3(b), where most of the clusters produced by Infomap were poorly connected on four of the six networks on which it ran.

MCL has high initial node coverage on cit_hepph, and the impact of running CM is moderate: before CM-processing, 97.5% of the nodes are in clusters of size at least two with 69% are in clusters of size at least 11, but after CM-processing, the node coverage drops to 61.7% (Table 1). Thus, while the major impact comes from the initial removal of the clusters of size at most 10 or tree clusters, there is also a reduction in node coverage as a result of finding clusters that are poorly connected.

For Infomap, cluster fates vary substantially across the different networks, with no particular trends holding (Table 2). Cluster fates for MCL show only 29.5% of the clusters of size at least 11 are extant, so that roughly 70% are modified by CM-processing (Table 2). Of these, the vast majority are reduced and the remaining are degraded.

Trends and observations. The response to CM-processing varies between methods and across networks. While results from all methods show reduced node coverage as a result of CM-processing, some methods are impacted to a greater extent.

A comparison between Leiden optimizing CPM with different resolution values and optimizing modularity shows the following trends. The impact on Leiden optimizing CPM depends on the resolution value: for larger resolution values, it tends to produce extant clusters, but for smaller resolution values it produces few extant clusters. Leiden optimizing modularity has similar responses to CM-processing as Leiden optimizing CPM with the smallest resolution value. We note also that the pre-CM and post-CM node coverage is highest for Leiden optimizing CPM using the smallest resolution value, and lowest for CPM-optimization with the two largest resolution values. An explanation for both trends is provided by the guarantee for the minimum cut size for any CPM-optimal cluster: the lower bound increases with the resolution value r, so that when $r = 0.5$, it provides a very meaningful guarantee. Thus, the Leiden CPM clusterings that are the most robust to CM-processing use larger resolution values, such as $r = 0.5$.

IKC clustering is also fairly robust to CM-processing, with generally a small reduction in node coverage, making it similar in that sense to Leiden-CPM with $r = 0.5$. It is striking to note that both of these methods have low initial node coverage as defined in our study, reflecting that the two methods are very *conservative* in producing clusters. In contrast, most of the clustering methods we studied (e.g., Leiden-CPM with small resolution values) had high initial node coverage, reaching close to 100% in most cases, and these were the cases that were most impacted by CM-processing. Interpreting results from MCL and Infomap is more difficult given that the former did not scale and the latter produced disconnected clusters.

These findings suggest that for the real-world networks tested, the clustering methods used may have been overly aggressive in optimizing node coverage. Second, that only a subset of the nodes assigned to clusters may actually belong according to them, if our definition of validity is enforced. An implication is that real-world networks may not be fully covered by what we consider "valid" communities, and clustering outputs may be overly optimistic.

Clustering on synthetic LFR networks

In our analyses of real-world networks, we observed substantial changes in clusterings after Stage 2 when using Leiden-CPM with large resolution values and after Stage 3 when using Leiden-CPM with small resolution values, or using Leiden-mod. Here we ask: when CM-processing changes the clustering, does it improve or hurt accuracy?

To evaluate the impact of CM-processing on accuracy, we generated synthetic networks (Tables K–T in S1 Appendix). For this experiment, we computed statistics for the Leiden clusterings of the seven real-world networks we previously explored, and used them as input to the LFR software [24] (Materials and methods). The LFR software failed to produce networks for any of the clusterings of the Orkut network or for two other networks (wiki_talk and wiki_topcats) using Leiden-CPM with $r = 0.5$. This left us with a collection of 34 LFR networks with ground truth communities. We then clustered each of these 34 LFR networks using the corresponding Leiden clustering parameters.

Properties of the LFR networks. We begin with an examination of the LFR network properties, as shown in Fig 7. The upper subfigure (top two rows) indicates that the LFR networks do not match the profile of node coverage exhibited in the original Leiden clustering. Most importantly, the lower subfigure shows that the majority of the ground truth clusters for the LFR wiki_talk networks and the two LFR CEN networks based on Leiden-CPM for the larger resolution values are disconnected (see also Table M in S1 Appendix).

By design, Leiden clusters are always connected, so that having LFR ground truth clusters be disconnected represents a substantial violation of the cluster properties for these real-world networks. To alleviate the concern that some LFR ground truth clusters are disconnected, we restricted our attention to those LFR networks where at most 15% of their ground truth clusters are disconnected. This meant that we removed seven LFR networks: all wiki_talk networks and two CEN networks; the remaining 27 networks were retained. Results on the LFR networks that had a large fraction of disconnected clusters are available in Table O in S1 Appendix.

Impact of CM-processing on clustering accuracy. We then examined the impact of CM-processing on clustering accuracy on these LFR networks with at most 15% disconnected clusters using Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and Adjusted Mutual Information (AMI), which were calculated using the Python Scikit-Learn package [28]. For all three criteria, CM usually improves or maintains accuracy, especially if applied to the networks based on Leiden-modularity or Leiden-CPM with small resolution values (Fig 8). Three conditions where CM hurts accuracy are cit_patents with CPM-optimization using $r \geq 0.1$, wiki_topcats for CPM-optimization using $r = 0.1$, and cit_hepph with CPM-optimization using $r = 0.1$. However, for all three cases, the effect of CM is almost entirely due to removing clusters below size 11 during Stage 2 (Table N in S1 Appendix). Therefore, while CM can reduce accuracy when it removes small clusters, the step where it finds and modifies poorly connected clusters is either neutral or beneficial to clustering accuracy.

Evaluating impact of the clustering coefficient

We examined the question of whether CM's impact on accuracy may be influenced by the network's average local clustering coefficient. As seen in Tables K and L, Figs A and B in S1 Appendix, the average local clustering coefficients of the LFR networks vary substantially, allowing us to examine how the impact of CM varies with the local clustering coefficient. Furthermore, the impact of CM tends to be positive (improving accuracy) when the average clustering coefficient is high, and is otherwise mostly neutral; the few cases where the impact is detrimental, so that clustering accuracy is decreased, are for those LFR network/clustering

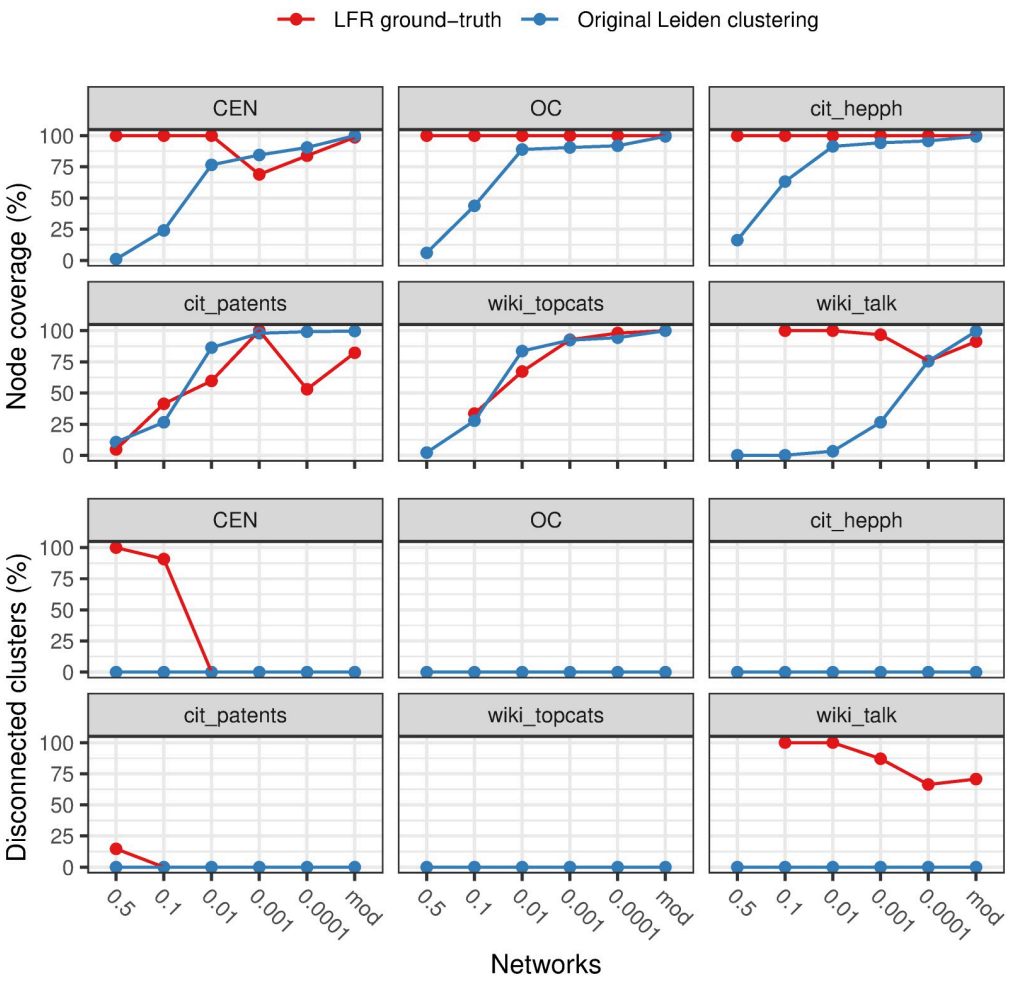

Fig 7. Properties of the LFR networks. The top subfigure shows the node coverage (percent of each network that is covered by clusters of size at least 11) and the bottom shows the percentage of the clusters of size at least 11 that are disconnected. Each LFR network is based on a real-world network, with cluster parameters, shown on the x-axis, based on Leiden optimizing modularity or CPM, with varying resolution parameters. Results not shown for LFR networks are due to the LFR software not returning networks for the provided input parameters.

https://doi.org/10.1371/journal.pcsy.0000009.g007

pairs with very low average local clustering coefficients, typically less than 0.1. It is noteworthy therefore to realize that the range of average local clustering coefficients for the real-world networks were almost all well above that value, suggesting again that the specific LFR networks where CM reduces accuracy are in the set where LFR produces networks that are different in important ways from the real-world network.

Since the clustering coefficient can be controlled for more directly with the nPSO software, we performed additional analyses using nPSO (Section F and Table T in S1 Appendix) that we summarize here. Under the nPSO3 model, we generated networks with 10,000 nodes and 100 communities, varying the average degree from 2 to 32 and temperature (which controls the clustering coefficient) from 0 to 0.9. Ten of the twenty nPSO networks we created had more than 15% of their ground truth clusters disconnected (the same threshold was used for LFR networks), and so were not included in the study. On the remaining networks, we found that Leiden and Leiden+CM clusterings had nearly identical accuracy. The only conditions under which there was a change of more than 1% occurred for the Leiden-CPM clusterings with

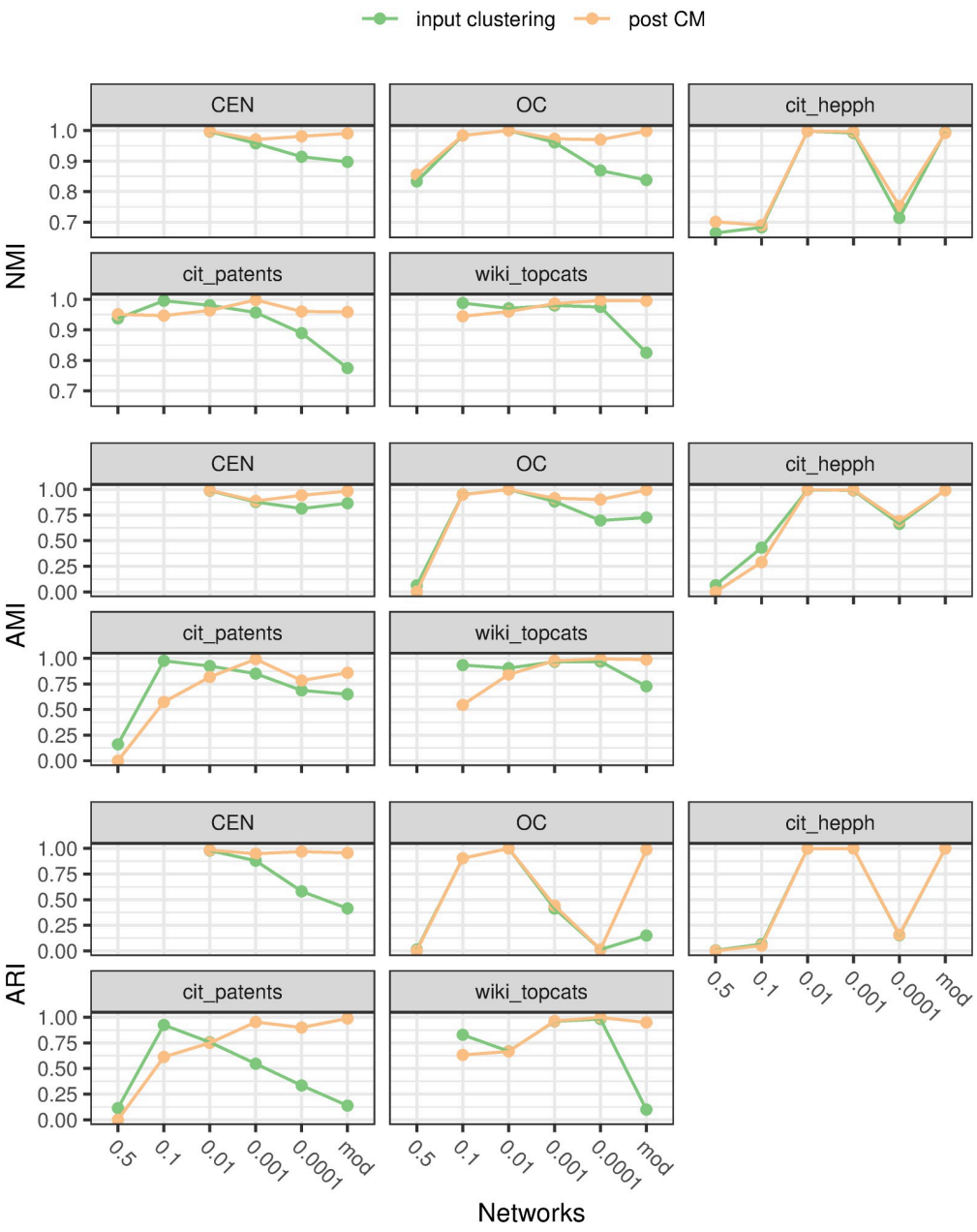

Fig 8. Impact of CM-processing on clustering accuracy on 27 LFR networks. The panels show accuracy measured in terms of NMI (top), AMI (middle) and ARI (bottom), with respect to the LFR ground-truth communities, restricted to the networks that did not have a large fraction of their clusters disconnected (27 of the original 34, see Fig 7). Each condition on the x-axis corresponds to a *different* LFR network, generated to reproduce the empirical properties of a Leiden-modularity or Leiden-CPM clustering with a given resolution parameter.

https://doi.org/10.1371/journal.pcsy.0000009.g008

$r = 0.1$ or $r = 0.5$, and only when the second stage of CM (which removes small clusters and tree clusters) produced a large decrease in node coverage due to having a large number of small clusters. For these clusterings and networks, clustering accuracy decreased. For all other cases, the impact of the CM pipeline was neutral.

The explanation is that the remaining stages of the CM pipeline, and in particular the third stage where poorly connected clusters are modified, resulted in very little change to the

clustering. This is distinctly different from what we observed for both LFR networks and real-world networks, where the third stage had an impact on the node coverage for Leiden-modularity and Leiden-CPM with small resolution values. Therefore, the trends on nPSO graphs are consistent with results on LFR networks.

We now consider how the clustering coefficient, controlled by the temperature parameter, influences whether CM helps or hurts accuracy. After restricting to the networks with a low incidence of disconnected ground truth clusters, the only case CM was other than neutral was for Leiden-CPM with $r = 0.5$ or $r = 0.1$, and this did not depend on the temperature or the average degree. Hence, the clustering coefficient of the nPSO networks has no impact in this study with respect to whether CM is beneficial or detrimental.

Results on Erdős-Rényi graphs

We performed an experiment on 50 Erdős-Rényi graphs, each with 10,000 nodes and different values of p, the probability of an edge between two nodes. These results, described in detail in Section E and Tables P–S in S1 Appendix, show that the impact of following a Leiden-CPM or Leiden-modularity clustering with CM depended on the value of p, with a large impact for the smaller values of p, i.e., eliminating all the nodes or greatly reducing their number, and a more modest impact for the largest values of p.

Thus, this experiment shows that for very large values of p, the default threshold we used for well-connectedness, $\log_{10} n$, was very permissive in that it allowed most or even all the clusters to remain untouched for the denser Erdős-Rényi graphs. Raising the threshold to a larger value, however, successfully removed the clusters. This experiment shows that the default threshold we picked is very generous, and that a higher threshold, which would modify even more clusters and reduce node coverage further, is needed if the purpose is to ensure that all the returned clusters are valid.

Evaluating how well LFR and nPSO networks resemble the real-world networks and clusterings. In this study, we used both LFR and nPSO networks to evaluate accuracy for different clustering methods, as well as to explore the impact of using CM-processing on this accuracy. The discussion above has already identified some aspects of the LFR networks that may not match their associated real-world networks very well. We now more fully examine this question, and also explore how well nPSO resembles the real-world networks we studied.

LFR networks. As provided in Tables K and L in S1 Appendix, statistics such as average node degree, the mixing parameter, and exponents for the power-law distributions for vertex degrees and community sizes in the LFR networks are very close to the statistics for the associated real-world networks. Yet, a more careful examination of the node degree distributions and community size distributions Figs C–I, K in S1 Appendix show that they are very different in their tails. We also see that the average local clustering coefficients in the LFR networks are very different from the real-world clustering coefficients, sometimes larger and sometimes smaller, and their values are poorly correlated (rank correlation coefficient of -0.23). Finally, as we discussed above, some of the LFR networks we generated using the LFR software had disconnected clusters. Given that all the communities we used to define the LFR networks are computed by Leiden and hence are guaranteed to be connected [8], this is another significant way in which LFR can produce ground truth clusters that do not match the features of the clustering of the associated real-world network.

Explaining why LFR produces networks with these different properties than their corresponding real-world networks is in some cases easy. Specifically, the difference in the degree distributions and cluster size distributions is explained by the realization that the real-world network and clusterings do not perfectly fit the power law distribution. It is known that the

power-law may not fit many real-world networks well [29–32], and so this is not surprising nor unusual. For the cases where the LFR network communities were disconnected, it is likely that the parameters given to the LFR methodology, which uses the configuration model, suggested very sparse clusters, and so the model produced disconnected clusters. This is a natural outcome of using the configuration model without any constraints for connected clusters. Why there are striking differences between the average clustering coefficients is less obvious, and merits further investigation.

Finally, we ask: to what extent do we see drops in node coverage for each of the CM pipeline stages, and do these resemble the trends observed for their corresponding real-world networks? Here we see reasonably similar trends. Specifically, when using Leiden-CPM with large resolution values (0.1 and 0.5), there is a drop in node coverage resulting from removing small clusters in Stage 2 and little to no further drop in Stage 3; this is similar to what we see in real-world networks. However, when using Leiden-modularity or Leiden-CPM with small resolution values, there is little or no drop in Stage 2, and a variable drop in node coverage (sometimes no drop, sometimes a large drop) when modifying poorly connected clusters in Stage 3; this is different from real-world networks, where we nearly always saw a substantial drop in Stage 3. Thus, Leiden-CPM clusterings using small resolution values show slightly different results on LFR networks than real-world networks.

nPSO networks. Unlike LFR, the input parameters for nPSO do not aim to produce networks that are particularly similar to a given real-world network. Therefore, we do not consider it appropriate to evaluate nPSO with respect to degree distribution or community size distribution. Nevertheless, we note the following. In order for us to obtain networks that had very few disconnected clusters, we needed to set the average degree to at least 8, and even then when the temperature was large (0.9) there were a large number of disconnected clusters. For all these networks, after clustering using Leiden-modularity or Leiden-CPM with small resolution values, the use of CM produced essentially no change in node coverage. This is very different from what we saw on the real-world networks, and even on the LFR networks.

Recall that the purpose of using synthetic networks was to understand the impact of CM on clustering accuracy, when using Leiden-CPM and Leiden-modularity. Therefore, it is necessary to explore conditions where CM changes the clustering, which is during Stage 2 (where it removes small clusters and tree clusters) and Stage 3 (where it finds and modifies poorly connected clusters). The more important of these is the impact of Stage 3, as ameliorating poorly connected clusters is the main point of this work, and also the stage that had the biggest impact on Leiden-mod and Leiden-CPM clusterings (when using resolution values that are not very large) on the real-world networks. Given this, the fact that Stage 3 did not produce a change in node coverage nor clustering accuracy on clusterings of nPSO networks suggests that nPSO is not helpful to our understanding of the impact of CM.

Summary

Our study, using a very modest requirement for well-connectedness, showed that CM-processing generally reduced node coverage, but the extent to which CM-processing changed a given clustering varies according to the input network and the choice of clustering algorithm with its algorithmic parameters, such as the resolution parameter for CPM-optimization.

We observed that those methods that achieved the highest node coverage, such as Leiden optimizing modularity or optimizing CPM with very small resolution values, e.g. $r = 0.0001$, typically had large drops in node coverage after CM-processing. In contrast, methods such as IKC or Leiden optimizing CPM with a very large resolution value (e.g., $r = 0.5$) had relatively low node coverage, and CM-processing had a minor impact on their node coverage. These

results together suggest that clustering methods that aim to maximize node coverage may be "over-clustering", and suggests the possibility that not all of the network may exhibit community structure.

This study also examined the impact of CM on clustering accuracy, using nPSO and LFR synthetic networks. After restriction to the networks that had a low incidence of disconnected ground truth communities, we observed that the CM pipeline sometimes reduced accuracy of Leiden clustering, optimizing either modularity or CPM, but only when there were many small clusters that were removed in the CM pipeline. We also observed that CM tended to improve accuracy when the third stage, which found and modified poorly connected clusters, produced a substantial change in the clustering reflected by a drop in node coverage; furthermore, the improvement could be large when the drop in node coverage was large. Under other conditions, the use of CM tended to be neutral to beneficial. These trends suggest that CM will improve accuracy for clusterings that tend to produce many poorly connected clusters, which is the case for many clustering methods, such as when using Leiden optimizing modularity or CPM with small resolution parameter values. While accuracy on synthetic networks is not necessarily indicative of accuracy on real-world networks, these results are encouraging.

Conclusions

Density and well-connectedness are two separable properties of clusters. Studying real-world networks ranging in size from tens of thousands to tens of millions of nodes with five clustering methods, we report that each of these methods produces clusters that do not meet a mild standard for well-connectedness, and some methods even produced disconnected clusters. Although somewhat limited in scope due to the small number of networks we explored, our study presents evidence that well-connectedness, although expected for valid communities, is not reliably produced for the clustering methods we explored.

Motivated by these observations, we designed the Connectivity Modifier (CM), a method that takes as input a network and an initial clustering and modifies the clustering iteratively—finding and removing edge cuts in clusters that are below the user-specified threshold for well-connectedness, and then clustering the subsets produced until all clusters are well-connected. A final stage removes all the clusters below a minimum size. Importantly, CM allows the user to set the thresholds for well-connectedness and minimum cluster size, which can depend on the context and purpose of the investigation, the network, the clustering algorithm used, and the parameter settings.

Our study, using a very mild standard for well-connectedness, revealed substantial differences between clustering methods, and suggested the possibility that some popular methods may "over-cluster". Using a simulation study on synthetic networks, we also noted that, for many conditions, CM improved clustering accuracy. These results suggest the need to address cluster quality with respect to how well-connected clusters are, and also suggest ways of improving clustering approaches, either through post-processing as in CM or through integration of a requirement for well-connectedness in clustering algorithms.

This study was based only on seven real-world networks and five clustering methods. While we selected clustering methods because they could scale to large networks, other clustering methods such as those based on Stochastic Block Models [33] certainly merit study in this regard. Clustering methods that may not scale to the network sizes such as LGI-MCL [34] and methods developed using Graph Neural Networks and deep learning could also be explored [35, 36]. Further exploration of trends on smaller networks would be helpful in clarifying whether the trends we observed are mostly seen in large networks. Further, the questions we address about well-connectedness can also be applied to community search methods that

return the best cluster for a given node or set of nodes [37, 38] and methods that return overlapping clusters [39–41].

We used a very modest threshold for well-connectedness. A cluster of size n was considered poorly connected if it had an edge cut of size at most $log_{10}n$. This setting was selected in order to be so low that failing to meet this requirement should be concerning. Larger settings of the threshold would therefore have removed edge cuts from more clusters and potentially reduced node coverage, and this was not our objective.

However, some research questions would seem to benefit from higher thresholds. For example, one potential use of methods like CM would be to determine, for a given input network, whether it has any valid community structure. This question has been posed in [42] and is related to the question of determining if a given cluster is statistically significant [43, 44]. Recalling how CM performed in the experiment on Erdős-Rényi graphs, it is clear that our default setting for the threshold was too low for this purpose. In fact, CM returned clusters on the dense Erdős-Rényi graphs, thus demonstrating that our default for the threshold is likely too permissive for this question of determining if a cluster is valid.

A third goal is to define and identify the clusterable portions of a network, and to some extent the output set of clusters produced by pipelines that include CM can be used for such a purpose. Clearly, setting a threshold too low for the network will not remove enough, while setting it too high will remove too much.

Given the value of weaker links [45], setting too high a threshold might reduce what can be learned from the network, making a lower threshold perhaps more suitable. In all these contexts, how the threshold is selected is of great importance, and will need to be set according to the network and the question at hand. More generally, this discussion highlights the importance of considering the goals and perspective of the researcher who wants to use community detection to answer a question [46].

This study also suggests a potential benefit of refining optimality criteria for clustering, so that well-connectedness is part of the optimization. For example, methods for partitioning the graph into "expanders" might produce communities with high set-conductance and low cut-conductance [47–49]. Therefore, a potentially interesting direction would be to use these theoretical ideas to develop new methods for practical use.

This study examined synthetic networks produced using LFR and nPSO, but other simulators have been developed that could be explored, including ABCD [50] and its extension ABCD+o [51] that allows for outliers, and Stochastic Block Models [33]. We note that a prior study showed that Stochastic Block Models (SBMs) did well in reproducing features of a large set of real-world networks [52], and so SBMs are obvious candidates. Finally, synthetic networks and clusterings could be modified through a post-processing step such as [53] to ensure that the clusters are connected, suggesting another potential direction for future work.

Materials and methods

Real-world networks

The real-world networks we used in this study are listed in Table 3. The publicly available Open Citations dataset was downloaded in Aug 2022, parsed into a PostgreSQL database, curated, and annotated with integer identifiers for each DOI [59]. The CEN is a citation network constructed from the literature on exosome research. From the SNAP repository, we downloaded cit_hepph, a high energy physics citation network; cit_patents, a citation network of 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

Table 3. Real-world networks used in this study.

network	# nodes	# edges	avg. deg	reference
Open Citations	75,025,194	1,363,303,678	36.34	[54]
CEN	13,989,436	92,051,051	13.16	[41]
cit_hepph	34,546	420,877	24.37	[55]
cit_patents	3,774,768	16,518,947	8.75	[55]
orkut	3,072,441	117,185,083	76.28	[56]
wiki_talk	2,394,385	4,659,565	3.89	[57]
wiki_topcats	1,791,489	25,444,207	28.41	[58]

https://doi.org/10.1371/journal.pcsy.0000009.t003

Wikimedia hyperlinks. All networks were treated as undirected and then processed to remove self-loops and duplicate edges before clustering.

LFR networks

Using the LFR software [24, 60] we generated synthetic networks with known community structure in order to evaluate accuracy, we need networks with ground truth communities. Details of this process are provided in Section B.2 in S1 Appendix; here we provide an overview of the protocol we followed.

The LFR software requires the following parameters:

- Network properties: The simulator requires information about the degree distribution of the network (assumed to be a power law), provided by the average k and maximum k_{max} vertex degrees, and the negative exponent for degree sequence (τ_1). It also requires the number N of nodes.
- Community properties: The simulator requires information about the cluster size distribution, provided by the maximum and minimum community sizes (c_{max} and c_{min}), and the negative exponent for the community size distribution (τ_2).
- Mixing parameter μ: The simulator also requires this value, which is the ratio between the number of neighbors of a vertex that are not in the same community, to the total number of neighbors of the vertex, averaged over all vertices.

We used properties of real-world networks to produce LFR networks. Network properties are directly available but community properties and mixing parameter values depend on a given clustering. Therefore, for each of the real-world networks, we also considered Leiden clusterings, produced either using modularity optimization or CPM-optimization with varying resolution parameters.

For each combination of real-world network and Leiden clustering, we estimated the community properties. We used *networkX* [61, 62] to compute the number of nodes and average and maximum node degrees. For the negative exponent of the two power law distributions, we used the approach from [63] that is implemented in the *powerlaw* Python package [64]. Since the power-law property may not hold for the whole distribution, following [63], we estimated x_{min}, the minimum value for which the power-law property holds as well as the exponent α for the tail of the distribution. We wrote a custom script to estimate the mixing parameter, μ.

We then gave the required parameters (network properties, community properties, and mixing parameter) to the LFR software [24]. We observed scalability limitations of the software [65] when trying to simulate very large networks, corresponding to the Open Citations network and CEN. Therefore, for any network with more than 10 million nodes, we simulated

networks with only 3 million nodes, using modified network and community property parameters (i.e., preserving average degree and mixing parameter).

We also found that for the LFR network simulator to converge, we had to modify the ranges of the community sizes, i.e., increase c_{min} and decrease c_{max}. As shown in the statistics reported in Tables K and L in S1 Appendix, this protocol produced LFR networks that had very good fit to the average node degree, the mixing parameter, and the exponents for the degree and community size distributions, of the associated real-world networks. Except for the LFR networks simulated for the CEN and Open Citations network, the LFR networks had the same number of nodes as the corresponding real-world network.

The connectivity modifier pipeline

To remediate poorly-connected clusters, we developed a modular pipeline that we now describe (Fig 4). By design, this pipeline is guaranteed to return a clustering where each cluster meets user-specified thresholds for well-connectedness and minimum size, as we now describe.

The CM pipeline (open source code available at [23]) 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 considered well-connected if and only if its min cut size exceeds $f(n)$ (default: $f(n) = \log_{10} n$); we require $f(n) \geq 1$ for all n, and that $f(n)$ be an increasing function.
- A clustering method; the current implementation allows for Leiden optimizing CPM, Leiden optimizing modularity, the Iterative k-core (IKC) method with $k = 10$, MCL, and Infomap. 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 tree clusters (since trees can be split into two parts by deleting a single edge, and are therefore not well-connected when they have size at least 10, according to our framework).
- 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 [26], 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.

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.

Supporting information

S1 Appendix. Supplementary materials document. This PDF document contains additional details about the LFR generation protocol, and additional results provided in 20

supplementary tables and 11 supplementary figures.
(PDF)

Author Contributions

Conceptualization: George Chacko, Tandy Warnow.

Data curation: George Chacko.

Formal analysis: Yasamin Tabatabaee, Vikram Ramavarapu, George Chacko.

Funding acquisition: George Chacko, Tandy Warnow.

Investigation: Minhyuk Park, Vikram Ramavarapu, George Chacko, Tandy Warnow.

Methodology: Minhyuk Park, Yasamin Tabatabaee, George Chacko, Tandy Warnow.

Project administration: George Chacko, Tandy Warnow.

Resources: George Chacko.

Software: Minhyuk Park, Yasamin Tabatabaee, Vikram Ramavarapu, Baqiao Liu, Vidya Kamath Pailodi, Rajiv Ramachandran, Dmitriy Korobskiy, Fabio Ayres.

Supervision: George Chacko, Tandy Warnow.

Validation: Minhyuk Park, George Chacko.

Writing – original draft: Minhyuk Park, Yasamin Tabatabaee, George Chacko, Tandy Warnow.

Writing – review & editing: Minhyuk Park, Yasamin Tabatabaee, George Chacko, Tandy Warnow.

References

1. Karataş A, Şahin S. Application areas of community detection: A review. In: 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT). IEEE; 2018. p. 65–70.
2. Dey AK, Tian Y, Gel YR. Community detection in complex networks: From statistical foundations to data science applications. WIREs Computational Statistics. 2021; 14(2). https://doi.org/10.1002/wics.1566
3. Newman ME, Girvan M. Finding and evaluating community structure in networks. Physical Review E. 2004; 69(2):026113. https://doi.org/10.1103/PhysRevE.69.026113 PMID: 14995526
4. Mucha PJ, Richardson T, Macon K, Porter MA, Onnela JP. Community Structure in Time-Dependent, Multiscale, and Multiplex Networks. Science. 2010; 328(5980):876–878. https://doi.org/10.1126/science.1184819 PMID: 20466926
5. Fortunato S, Newman MEJ. 20 years of network community detection. Nature Physics. 2022; 18 (8):848–850. https://doi.org/10.1038/s41567-022-01716-7
6. Javed MA, Younis MS, Latif S, Qadir J, Baig A. Community detection in networks: A multidisciplinary review. J Netw Comput Appl. 2018; 108:87–111. https://doi.org/10.1016/j.jnca.2018.02.011
7. Coscia M, Giannotti F, Pedreschi D. A classification for community discovery methods in complex networks. Stat Anal Data Min. 2011; 4(5):512–546. https://doi.org/10.1002/sam.10133
8. Traag VA, Waltman L, Van Eck NJ. From Louvain to Leiden: guaranteeing well-connected communities. Scientific Reports. 2019; 9(1):1–12. https://doi.org/10.1038/s41598-019-41695-z PMID: 30914743
9. Zhu ZA, Lattanzi S, Mirrokni V. A local algorithm for finding well-connected clusters. In: International Conference on Machine Learning. PMLR; 2013. p. 396–404.
10. Bonchi F, García-Soriano D, Miyauchi A, Tsourakakis CE. Finding densest k-connected subgraphs. Discrete Applied Mathematics. 2021; 305:34–47. https://doi.org/10.1016/j.dam.2021.08.032
11. Shun J, Roosta-Khorasani F, Fountoulakis K, Mahoney MW. Parallel local graph clustering. Proceedings of the VLDB Endowment. 2016; 9(12):1041–1052. https://doi.org/10.14778/2994509.2994522

12. Emmons S, Kobourov S, Gallant M, Börner K. Analysis of network clustering algorithms and cluster quality metrics at scale. PloS one. 2016; 11(7):e0159161. https://doi.org/10.1371/journal.pone.0159161 PMID: 27391786

13. Kannan R, Vempala S, Vetta A. On clusterings: Good, bad and spectral. Journal of the ACM (JACM). 2004; 51(3):497–515. https://doi.org/10.1145/990308.990313

14. Torghabeh RP, Santhanam NP. Modeling community detection using slow mixing random walks. In: 2015 IEEE International Conference on Big Data (Big Data). IEEE; 2015. p. 2205–2211.

15. Avrachenkov K, El Chamie M, Neglia G. Graph clustering based on mixing time of random walks. In: 2014 IEEE International Conference on Communications (ICC). IEEE; 2014. p. 4089–4094.

16. Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment. 2008; 2008(10):P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008

17. Rosvall M, Bergstrom CT. Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences. 2008; 105(4):1118–1123. https://doi.org/10.1073/pnas.0706851105 PMID: 18216267

18. Dongen SV. Graph Clustering Via a Discrete Uncoupling Process. SIAM Journal on Matrix Analysis and Applications. 2008; 30(1):121–141. https://doi.org/10.1137/040608635

19. Wedell E, Park M, Korobskiy D, Warnow T, Chacko G. Center-periphery structure in research communities. Quantitative Science Studies. 2022; 3(1):289–314. https://doi.org/10.1162/qss_a_00184

20. Liu B, Park M. Connectivity Modifier; 2022. https://github.com/RuneBlaze/connectivity-modifier.

21. Ramavarapu V, Ayres F, Park M, Pailodi VK, Chacko G, Warnow T. Connectivity Modifier; 2023. https://github.com/illinois-or-research-analytics/cm_pipeline.

22. Park M, Tabatabaee Y, Ramavarapu V, Liu B, Pailodi VK, Ramachandran R, et al. Identifying well-connected communities in real-world and synthetic networks. In: Proceedings of the 2023 International Conference on Complex Networks and Their Applications. Springer; 2023. p. 3–14.

23. Ramavarapu V, Ayres FJ, Park M, Pailodi VK, Lamy JAC, Warnow T, et al. CM++—A Meta-method for Well-Connected Community Detection. Journal of Open Source Software. 2024; 9(93):6073. https://doi.org/10.21105/joss.06073

24. Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs for testing community detection algorithms. Physical Review E. 2008; 78(4):046110. https://doi.org/10.1103/PhysRevE.78.046110 PMID: 18999496

25. Muscoloni A, Cannistraci CV. A nonuniform popularity-similarity optimization (nPSO) model to efficiently generate realistic complex networks with communities. New Journal of Physics. 2018; 20(5):052002. https://doi.org/10.1088/1367-2630/aac06f

26. Henzinger M, Noe A, Schulz C, Strash D. Practical Minimum Cut Algorithms. ACM Journal of Experimental Algorithmics. 2018; 23:1–22. https://doi.org/10.1145/3274662

27. Fortunato S, Barthélemy M. Resolution limit in community detection. Proceedings of the National Academy of Sciences. 2007; 104(1):36–41. https://doi.org/10.1073/pnas.0605965104 PMID: 17190818

28. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res. 2011; 12:2825–2830.

29. Radicchi F, Fortunato S, Castellano C. Universality of citation distributions: Toward an objective measure of scientific impact. Proceedings of the National Academy of Sciences. 2008; 105(45):17268–17272. https://doi.org/10.1073/pnas.0806977105 PMID: 18978030

30. Stringer MJ, Sales-Pardo M, Amaral LAN. 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. 2010; 61(7):1377–1385. https://doi.org/10.1002/asi.21335 PMID: 21858251

31. Artico I, Smolyarenko I, Vinciotti V, Wit EC. How rare are power-law networks really? Proceedings of the Royal Society A. 2020; 476(2241):20190742. https://doi.org/10.1098/rspa.2019.0742

32. Brzezinski M. Power laws in citation distributions: evidence from Scopus. Scientometrics. 2015; 103:213–228. https://doi.org/10.1007/s11192-014-1524-z PMID: 25821280

33. Peixoto TP. Efficient Monte Carlo and greedy heuristic for the inference of stochastic block models. Physical Review E. 2014; 89(1):012804. https://doi.org/10.1103/PhysRevE.89.012804 PMID: 24580278

34. Durán C, Muscoloni A, Cannistraci CV. Geometrical inspired pre-weighting enhances Markov clustering community detection in complex networks. Applied Network Science. 2021; 6(1):29. https://doi.org/10.1007/s41109-021-00370-x

35. Gao J, Chen J, Li Z, Zhang J. ICS-GNN: lightweight interactive community search via graph neural network. Proceedings of the VLDB Endowment. 2021; 14(6):1006–1018. https://doi.org/10.14778/3447689.3447704

36. Su X, Xue S, Liu F, Wu J, Yang J, Zhou C, et al. A comprehensive survey on community detection with deep learning. IEEE Transactions on Neural Networks and Learning Systems. 2022; 35:4682–4702. https://doi.org/10.1109/TNNLS.2021.3137396

37. Van Laarhoven T, Marchiori E. Local network community detection with continuous optimization of conductance and weighted kernel k-means. Journal of Machine Learning Research. 2016; 17(147):1–28.

38. Fang Y, Huang X, Qin L, Zhang Y, Zhang W, Cheng R, et al. A survey of community search over big graphs. The VLDB Journal. 2020; 29:353–392. https://doi.org/10.1007/s00778-019-00556-x

39. Banerjee A, Krumpelman C, Ghosh J, Basu S, Mooney RJ. Model-based overlapping clustering. In: Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining; 2005. p. 532–537.

40. Baadel S, Thabtah F, Lu J. Overlapping clustering: A review. In: 2016 SAI Computing Conference (SAI). IEEE; 2016. p. 233–237.

41. Jakatdar A, Liu B, Warnow T, Chacko G. AOC: Assembling overlapping communities. Quantitative Science Studies. 2022; 3(4):1079–1096. https://doi.org/10.1162/qss_a_00227

42. Miasnikof P, Shestopaloff AY, Raigorodskii A. Statistical power, accuracy, reproducibility and robustness of a graph clusterability test. International Journal of Data Science and Analytics. 2023; 15(4):379–390. https://doi.org/10.1007/s41060-023-00389-6

43. Lancichinetti A, Radicchi F, Ramasco JJ. Statistical significance of communities in networks. Physical Review E. 2010; 81(4):046110. https://doi.org/10.1103/PhysRevE.78.046110

44. Lancichinetti A, Radicchi F, Ramasco JJ, Fortunato S. Finding statistically significant communities in networks. PloS one. 2011; 6(4):e18961. https://doi.org/10.1371/journal.pone.0018961 PMID: 21559480

45. Granovetter MS. The strength of weak ties. American Journal of Sociology. 1973; 78(6):1360–1380. https://doi.org/10.1086/225469

46. von Luxburg U, Williamson RC, Guyon I. Clustering: Science or Art? In: Guyon I, Dror G, Lemaire V, Taylor G, Silver D, editors. Proceedings of ICML Workshop on Unsupervised and Transfer Learning. vol. 27 of Proceedings of Machine Learning Research. Bellevue, Washington, USA: PMLR; 2012. p. 65–79. Available from: https://proceedings.mlr.press/v27/luxburg12a.html.

47. Arora S, Rao S, Vazirani U. Expander flows, geometric embeddings and graph partitioning. Journal of the ACM (JACM). 2009; 56(2):1–37. https://doi.org/10.1145/2837020

48. Gharan SO, Trevisan L. Partitioning into expanders. In: Proceedings of the twenty-fifth annual ACM-SIAM Symposium on Discrete Algorithms. SIAM; 2014. p. 1256–1266.

49. Saranurak T, Wang D. Expander decomposition and pruning: Faster, stronger, and simpler. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms. SIAM; 2019. p. 2616–2635.

50. Kamiński B, Prałat P, Théberge F. Artificial Benchmark for Community Detection (ABCD)—Fast random graph model with community structure. Network Science. 2021; 9(2):153–178. https://doi.org/10.1017/nws.2020.45

51. Kamiński B, Prałat P, Théberge F. Outliers in the ABCD Random Graph Model with Community Structure (ABCD+o). In: Complex Networks and Their Applications XI. Springer International Publishing; 2023. p. 163–174.

52. Vaca-Ramírez F, Peixoto TP. Systematic assessment of the quality of fit of the stochastic block model for empirical networks. Physical Review E. 2022; 105(5):054311. https://doi.org/10.1103/PhysRevE.105.054311 PMID: 35706168

53. Viger F, Latapy M. Efficient and simple generation of random simple connected graphs with prescribed degree sequence. Journal of Complex Networks. 2016; 4(1):15–37. https://doi.org/10.1093/comnet/cnv013

54. Peroni S, Shotton D. OpenCitations, an infrastructure organization for open scholarship. Quantitative Science Studies. 2020; 1(1):428–444. https://doi.org/10.1162/qss_a_00023

55. Leskovec J, Kleinberg J, Faloutsos C. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In: Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining. ACM; 2005. p. 177–187.

56. Yang J, Leskovec J. Defining and evaluating network communities based on ground-truth. Knowledge and Information Systems. 2013; 42(1):181–213. https://doi.org/10.1007/s10115-013-0693-z

57. Leskovec J, Huttenlocher D, Kleinberg J. Signed networks in social media. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM; 2010. p. 1361–1370.

58. Yin H, Benson AR, Leskovec J, Gleich DF. Local Higher-Order Graph Clustering. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM; 2017. p. 555–564.

59. Korobskiy D, Chacko G. Curated Open Citations Dataset; 2023. Available from: https://databank.illinois.edu/datasets/IDB-6389862.

60. Fortunato S. Resources; 2023. https://www.santofortunato.net/resources.

61. Hagberg A, Swart P, S Chult D. Exploring network structure, dynamics, and function using NetworkX; 2008. Technical Report No. LA-UR-08-05495; LA-UR-08-5495, Los Alamos National Laboratory, available at https://www.osti.gov/biblio/960616.

62. Tabatabaee Y. Emulating real networks using LFR graphs; 2023. https://github.com/ytabatabaee/emulate-real-nets.

63. Clauset A, Shalizi CR, Newman ME. Power-law distributions in empirical data. SIAM Review. 2009; 51 (4):661–703. https://doi.org/10.1137/070710111

64. Alstott J, Bullmore E, Plenz D. powerlaw: a Python package for analysis of heavy-tailed distributions. PloS one. 2014; 9(1):e85777. https://doi.org/10.1371/journal.pone.0085777 PMID: 24489671

65. Slota GM, Berry JW, Hammond SD, Olivier SL, Phillips CA, Rajamanickam S. Scalable generation of graphs for benchmarking HPC community-detection algorithms. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis; 2019. p. 1–14.

66. Park M, Tabatabaee Y, Warnow T, Chacko G. Data for Well-Connectedness and Community Detection; 2024. Available from: https://doi.org/10.13012/B2IDB-6271968_V1.