Synthetic Networks That Preserve Edge Connectivity

Lahari Anne, The-Anh Vu-Le, Minhyuk Park, Tandy Warnow, and George Chacko(✉)

Siebel School of Computing and Data Science, Grainger College of Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801, USA
{warnow,chackoge}@cs.illinois.edu

Abstract. Since true communities within real-world networks are rarely known, synthetic networks with planted ground truths are an alternative for evaluating community detection methods. Of several available synthetic network generators, Stochastic Block Models (SBMs) produce networks with ground truth clusters that well approximate input parameters from real-world networks and clusterings. However, SBMs can produce disconnected ground truth clusters, even when provided parameters from clusterings where all clusters are connected. Here we describe the REalistic Cluster Connectivity Simulator (RECCS), a technique that creates and then modifies an SBM synthetic network to improve the fit to a given clustered real-world network. Using real-world networks up to 13.9 million nodes in size, we show that RECCS results in synthetic networks that have a better fit to cluster edge connectivity than their starting SBMs, while providing roughly the same quality fit for other network and clustering parameters as unmodified SBMs.

Keywords: synthetic networks · community detection

1 Introduction

Community detection methods resolve networks at the meso-scale by identifying clusters of nodes that exhibit desired properties, such as density, edge connectivity, and separability from the remainder of the network [3–5,7,13,18]. An understanding of when a method produces clusters of good quality is of some importance given the diverse applications of community detection (clustering) methods.

Since ground truth communities are not reliably known in real-world networks, evaluation using synthetically generated networks with planted ground truth communities provides a useful alternative [14]. Several synthetic network generators are in use, including Stochastic Block Models (SBM) [15,16], the

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-82435-7_14.

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2024, SCI 1189, pp. 166–177, 2025.
https://doi.org/10.1007/978-3-031-82435-7_14

LFR (Lancichinetti-Fortunato-Radicchi) generator [10], ABCD and ABCD+o (Artificial Benchmark for Community Detection) [8,9], and nPSO (nonuniform popularity similarity optimization) [12].

Synthetic network generators should produce networks with ground truth clusterings with properties similar to those of real-world networks [12,19]. It has been shown that SBMs produced using graph-tool [15] have a good fit to many real-world *network properties*, such as degree sequence, local and global clustering coefficients, and diameter [19]. However, whether the ground-truth *clusterings* in the synthetic networks resembled the clusterings of the real-world networks and the fit between outlier nodes in the clustered real-world networks and the outlier nodes in the synthetic network was not examined. However, in [20], the importance of having ground truth clusters that are at least connected is recognized, along with an approach to modify a synthetic network with ground truth clustering to ensure connectedness.

Here, we present RECCS, the REalistic Cluster Connectivity Simulator, which addresses the goal of approximating the edge connectivity of clusters in a given clustered real-world network, while not worsening the fit with respect to other network and clustering properties. The input to RECCS is a real-world network and a set of parameters obtained from clustering it. RECCS first computes and then modifies an SBM for the clustered subnetwork in order to improve the fit to the edge connectivity in the real-world clustering. In the second step the remaining "outlier" nodes are added. We show that, compared to SBM alone, this two-step approach produces synthetic networks that have excellent fit for the edge connectivity of the real-world clustered network, while maintaining the fit for other empirical statistics of the clustered real-world network.

2 Materials and Methods

2.1 New Synthetic Network Generation Pipelines

High-Level Description. The input is a real-world network G and its clustering. Note that the clustering may include clusters containing only a single node (singleton clusters). We refer to nodes in singleton clusters as being "unclustered" or "outliers", and all other nodes are considered"clustered". The subnetwork of G induced by the non-singleton clusters is called the "clustered subnetwork" and is denoted by G_c. From this network and clustering, we extract the parameters required for degree-corrected SBM network generation, which includes the assignment of nodes to clusters, number of edges within each cluster and between each pair of clusters, and the node degree sequence. We also calculate the edge connectivity of each of the clusters, which we now define. For a given non-singleton cluster C, an edge cut is a set of edges such that deleting those edges, but not the endpoints, disconnects the cluster C. The size of the smallest edge cut for C is its edge connectivity, and is denoted $k(C)$.

We then produce a synthetic network N using the degree-corrected SBM network generation methods in graph-tool to model the clustered subnetwork G_c. If G has any unclustered nodes, then the clustered subnetwork G_c will not be

the entire network. Next, we make the SBM network a *simple graph* by removing self-loops and replacing each set of parallel edges with a single edge.

We call this modified network N_c, noting that it is a reduced version of the original SBM network N, and may have fewer edges than in the real-world network G_c; this provides us the opportunity to strategically add edges to ensure the required edge connectivity within the clusters while maintaining the integrity of other network properties. This is Step 1 of the network generation procedure.

We also add outlier nodes back into the network, and determine the edges that are incident with these outlier nodes; this is Step 2 of the network generation process, which produces a network only containing edges involving at least one outlier node. Finally, the two synthetic networks are *merged* into one synthetic network. For additional details, see the Supplementary Materials [1].

Step 1: Improving Edge Connectivity. We refer to the set of parameters we calculate from the real-world network G_c and its clustering $\mathcal{C}$ as $Param(G_c, \mathcal{C})$. The problem we seek to solve can be described as follows:

- Input: Simple graph N_c with clustering $\mathcal{C}$ and $Param(G_c, \mathcal{C})$, where $\mathcal{C}$ does not have singleton clusters
- Output: Network N with the same clustering $\mathcal{C}$ formed by adding edges to N_c, with the objective of having a good fit to $Param(G_c, \mathcal{C})$.

The set $Param(G_c, \mathcal{C})$ includes the edge connectivity values $k(C)$ for every cluster in $\mathcal{C}$. Since SBMs in general provide a good fit to many network parameters but not to edge connectivity in clusters, we seek to improve the fit to the $k(C)$ values without hurting the fit to the other parameters.

In our experiments, we explored techniques to solve this problem that operate in two phases. In the first phase, we add edges to ensure that every cluster has edge connectivity at least $k(C)$, where $\{k(C) : C \in \mathcal{C}\}$ is part of part of the input parameter set. In the second phase we add additional edges to improve the fit with respect to the degree sequence. The following two-phase approach comprises Step 1 of the "RECCS" workflow, shown in Fig. 1.

Step 2: Adding in Outliers. Given the network G and its clustering $\mathcal{C}$, we now focus on creating a synthetic version of the subnetwork G^* of G containing the same nodes but only those edges where at least one endpoint is an outlier node, i.e., the "outlier edges", which can connect two outlier nodes or one outlier node and one clustered node. Note that G^* includes clustered nodes as well as outlier nodes, and has the same set of clusters as in $\mathcal{C}$; however, there are no edges within any non-singleton cluster, and no edges between any two non-singleton clusters.

We propose three strategies for this problem, ranging from Strategy 1, which has the least randomness, to Strategy 3, which has the most randomness. Each takes the input parameters computed from G^* and $\mathcal{C}$ and then returns a synthetic network N^* consisting of outlier edges.

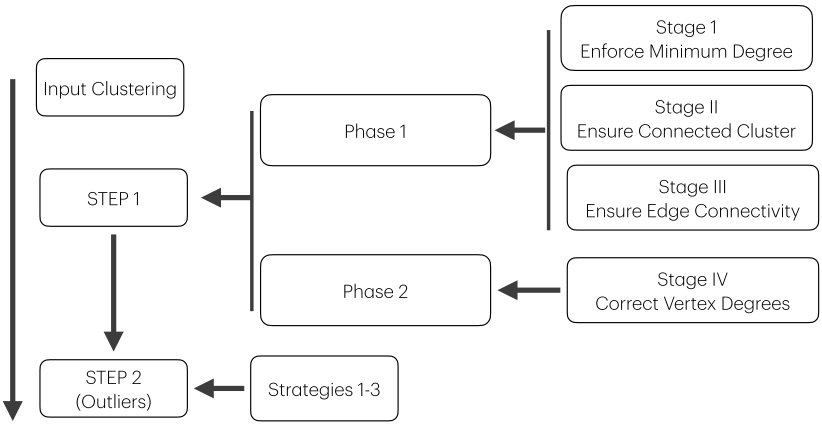

Fig. 1. RECCS **Workflow.** The input to RECCS is a collection of parameters computed on a real-world network N and an estimated clustering. Step 1 of RECCS produces a synthetic network corresponding to the clustered subnetwork of N, and Step 2 adds in the "outlier" nodes. Phase 1 of Step 1 computes an initial synthetic network using the graph-tool SBM software, and then adds edges within the clusters and between clusters to achieve a good fit to the input parameters, focusing on achieving the cluster connectivity (Stages 1, 2, and 3); Phase 2 then modifies the network to improve the fit with respect to the degree sequence. See Sect. 3 for additional details.

Strategy 1: This approach has each outlier in its own cluster, and then passes all the parameters computed for G^* to degree-corrected SBM to generate a network.

Note that this approach reproduces exactly the edges between outlier nodes as well as the number of edges between each outlier node and non-singleton cluster. However, there is still randomness in the assignment of edges between outlier nodes and clustered nodes.

Strategy 2: All the outliers are placed in a single cluster. The set of parameters restricted to the outlier cluster (i.e., the within-cluster degree sequence and number of edges) is used to generate the edges for the outlier cluster.

The remaining edges, between outliers and clustered nodes, are added at random for each outlier node and non-singleton cluster from $\mathcal{C}$ in turn. This strategy is more random than Strategy 1 for how it places edges between outlier nodes, but handles edges between outliers and clustered nodes identically as Strategy 1.

Strategy 3: All the outliers are placed in a single cluster, and the parameters from G^* are used to generate a degree-corrected SBM.

Postprocessing: We postprocess N^* as follows: (1) if any self-loops or parallel edges have been created, the excess edges are removed so that the network becomes a simple graph, and (2) if the outlier nodes had been placed in a single cluster during the generation process (Strategies 2 and 3), then this artificial

treatment is ignored, and the outlier nodes are each considered to be singleton clusters.

Merging: Finally, the random networks generated in Step 1 and Step 2 are merged together; this is straightforward, since the node labelings for each random network are drawn from G.

2.2 Datasets

The datasets used comprise clustered networks, with some specifically used for the algorithm design (training) phase, and others used for the evaluation (testing) phase.

Real-world networks: We used 110 real-world networks that ranged from 1,000 to 13.9 million nodes that were taken from Netzschleuder [17] and the SNAP [11] repositories; see Supplementary Materials [1] for the list of datasets.

Clustering Methods: We clustered all the real-world networks by using a standard clustering method followed by the Connectivity Modifier (CM) [13]. For clustering, we used the Leiden algorithm [18] optimizing either the Constant Potts Model (CPM) with a range of resolution parameters or modularity, and the Iterative k-core method [21].

The Connectivity Modifier operates as follows. First, it removes clusters below size 11. Then, for all clusters that are considered to be poorly connected (because the size of its minimum edge cut is at most $\log_{10}(n)$ where n is the number of nodes in the cluster), CM removes the edge cut (thus breaking the cluster into two pieces), reclusters each piece, and repeats, until all clusters are well-connected. Finally, the clusters below the minimum size (11 by default) are removed. This last step was omitted in this study.

Training Data: We clustered all 110 networks using Leiden optimizing the Constant Potts Model (CPM) with resolution parameter $r = 0.001$, followed CM as described above.

Testing Data: We selected six large real-world networks from the set of 110 networks—Cit_hepph, Cit_patents, the Curated Exosome Network (CEN), Orkut, Wiki_talk, and Wiki_topcats—and clustered these networks using Leiden optimizing CPM at two different resolutions ($r = 0.01$ and $r = 0.1$), Leiden optimizing modularity, and Iterative k-Core (IKC) with $k = 10$, each followed by CM. Because the clusterings are different, these testing data are disjoint from the training data. Additional results for these 110 networks using a different clustering pipeline are included in the Supplementary Materials [1].

2.3 Evaluation Criteria

We evaluate the similarity between synthetic and real-world networks using various network properties. The key network properties include the edge connectivity of the clusters (minimum cuts), network diameter, mixing parameter, degree sequence, ratio of disconnected clusters, and clustering coefficients (both global and local). We use (1) a simple difference, calculated as $(s - s')$, where s is

the statistic value of the real-world network and s' is the statistic value of the synthetic network, for scalar properties bounded between 0 and 1, (2) relative difference calculated as $(s - s')/s$ for unbounded scalar statistics, and (3) Root Mean Square Error (RMSE), where RMSE $= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(s_i - s'_i)^2}$, for comparing sequences. For outlier modeling strategies, we examine the outlier degree sequence and the number of edges involving outlier nodes. Finally, we report the normalized edit distance between the synthetic network and the real-world network, given the bijection between the node sets.

2.4 Experiments

Experiment 1 - Preliminary analysis of SBM: We compute synthetic networks using SBM given parameters on the training dataset. The parameters that are given include the node-to-cluster assignment, number of edges within each cluster and between each pair of clusters, and the degree of every node in the network. The main focus is evaluating the frequency of disconnected ground truth clusters.

Experiment 2 - Algorithm Design and Development: We explore the algorithm design for the two steps of the synthetic network generation strategy, where the first step modifies the starting network with respect to edge connectivity within clusters and the second step focuses on outlier modeling. We use training data for this experiment, and compare our algorithms to SBM generation in graph-tool, post-processed to remove excess edges.

Experiment 3 - Evaluation on Test Data: We evaluate the pipelines we pick in Experiment 2 on testing datasets, in comparison to the generation of SBMs in graph-tool, post-processed to remove excess edges.

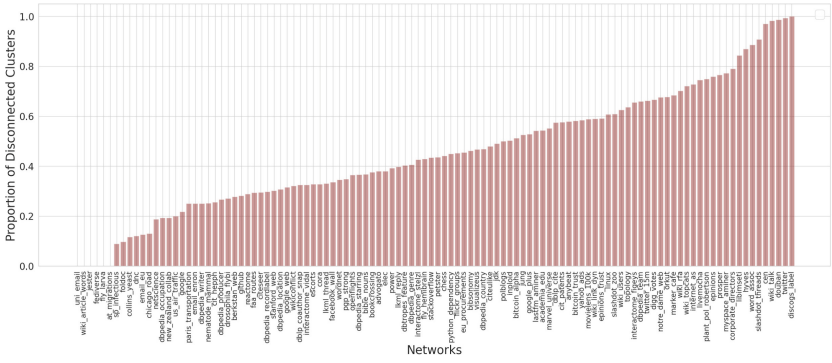

Fig. 2. Proportion of disconnected clusters in SBM generated networks. The x-axis shows 110 SBM networks generated using parameters from real world networks clustered with the Leiden+CM (Connectivity Modifier) pipeline (training data). The SBM method failed to reproduce the guaranteed connectivity of Leiden+CM clusters.

3 Results and Discussion

3.1 Experiment 1 Results: Preliminary Evaluation of SBM

Figure 2 shows the proportion of disconnected clusters in the 110 synthetic networks generated by SBM with inputs from training data as detailed in Sect. 2. Note that approximately half the networks have more than 40% of their clusters disconnected. Since the input clusterings were based on Leiden+CM and these are guaranteed to be connected, the input clustering parameters given to the SBM generation method in graph-tool are achievable with connected clusters, showing that SBM fails to recover this basic feature of the input clustering. This trend motivates our study.

3.2 Experiment 2 Results: Design of RECCS

The algorithmic structure of RECCS, provided in Fig. 1, is the result of a sequence of experiments on training data that we now describe. RECCS begins by computing a synthetic network N_c based on the set $Param(G_c, \mathcal{C})$ of parameter values computed from the clustered subnetwork G_c of a given real-world network G and its clustering $\mathcal{C}$, as described in Sect. 2.1. Recall that N_c is produced by removing excess edges from the SBM network, but it has the same ground truth clustering. RECCS operates by adding edges to N_c in two phases. Phase 1 adds edges to ensure that every cluster C has at least the target edge connectivity $k(C)$ (given in the input), and Phase 2 adds edges to improve the fit of the resultant degree sequence.

In our design of RECCS we initially explored techniques that omitted the second phase. These approaches had excellent fit for the edge connectivity but did not do as well for degree sequence. We then modified the design of first phase to also consider the degree of the nodes when adding edges to the network. However, these modifications did not fully address the edge deficit. We then introduced the second phase, which adds more edges to improve the degree sequence fit, but noticed that it slightly worsened the mixing parameter fit. To counter this effect, we developed a second version of the second phase. This resulted in two versions of RECCS that differ only in Phase 2, which is when edges are added to improve the fit of the degree sequence. RECCS operates in two phases, as follows.

- **Phase 1 (Ensure Edge Connectivity):** Edges are added to each cluster C to ensure edge connectivity at least $k(C)$, as follows:
  - **Stage 1: Enforce minimum degree:** Here we add edges to ensure that every node has at least $k(C)$ neighbors in the cluster. Therefore, if a node v has $d < k(C)$ neighbors in the cluster, we add $k(C) - d$ edges between v and other nodes in the cluster to which it is not adjacent.
  - **Stage 2: Ensure connected clusters.** If a cluster C is disconnected, we add $k(C)$ edges at random between its largest component and each of the other components.

  - **Stage 3: Ensure edge connectivity.** This stage is an iterative method that ensures that the cluster C has edge connectivity at least $k(C)$. Specifically, we use VieCut [6] to calculate the size of a minimum edge cut; if this size is at least $k(C)$, then the cluster meets the desired minimum connectivity. Otherwise, the edge cut defines a partition of the cluster into two parts, and we add the required number of edges between the two parts. We repeat the process until the mincut of the cluster is at least $k(C)$.
- **Phase 2 (Correct vertex degrees):** We add edges to increase the degree of nodes with available degree (i.e., nodes whose current degree is below their target values) using two different techniques:
  - $v1$: for each node with available degree, we add edges to other nodes with available degree, following Algorithm 1 in the Supplementary Materials [1].
  - $v2$: edges are strategically added to nodes with available degrees, taking into account the number of inter-cluster and intra-cluster edges in the input subgraph G_c. This approach, detailed in Algorithm 2 in the Supplementary Materials [1], restricts the addition of edges to not exceed the expected number of inter-cluster edges.

Adding Edges. Recall that we are given a target degree for every node from G_c, but the synthetic network N may not achieve this degree for some nodes. Those nodes whose current degree is less than the target degree are said to be "nodes with available degree". At each stage of Phase 1 of the algorithm, when adding an edge, we first randomly select nodes within the cluster with available degrees and update their available degree status accordingly. If no suitable nodes with available degree are found, we then randomly choose other nodes within the cluster, even if they have no available degree, to add the edge. We do not add parallel edges and self-loops.

3.3 Experiment 3 Results: Evaluation on Test Data

Here we show results of the two-step pipelines on the test data. There are six versions of the two-step pipeline, each formed by using RECCSv1 or RECCSv2 for the first step, and then followed by adding in the outlier nodes in three different ways in the second step. A comparison between all six pipelines on the full set of test networks and clusterings is shown in the Supplementary Materials [1], and reveals that Strategy 1 for Step 2 has the best accuracy. Due to space limitations, we present results here just for the two pipelines using outlier Strategy 1.

We begin with the results when clustering using Leiden-CPM with $r = 0.01$ followed by CM, in each case using Strategy 1 for outlier modeling. Both RECCSv1 and RECCSv2 improve the fit for the minimum edge cut size compared to SBM (Fig. 3, top panel). We also see an improvement in fit for both versions for degree sequence compared to SBM (with a larger improvement for

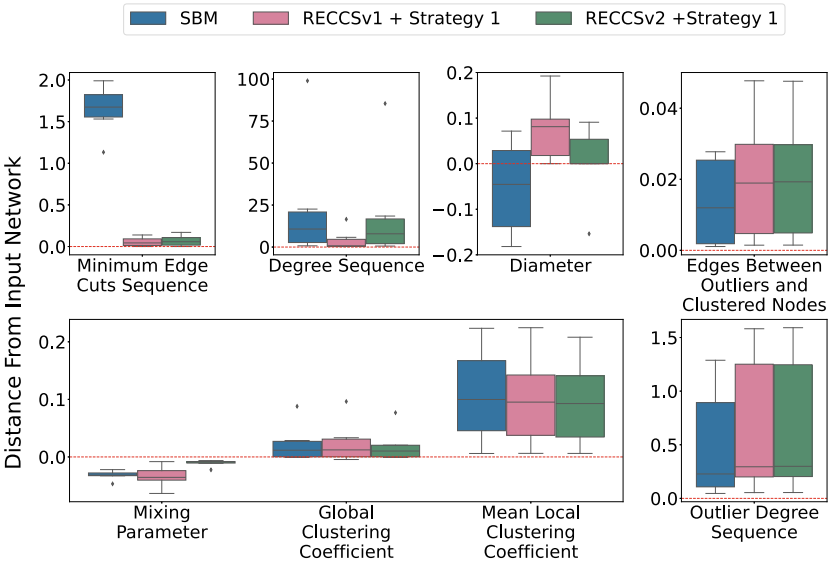

Fig. 3. Comparing SBM to the RECCS pipelines on the test networks using Leiden-CPM(0.01)+CM. We compare SBM networks to networks produced using the two pipelines, RECCSv1+Strategy 1 and RECCSv2+Strategy 1, for different network and clustering statistics. The y-axis shows different distance metrics for various network properties. Error is reported using RMSE for degree sequence, outlier degree sequence, and minimum edge cuts sequence; scalar difference for clustering coefficients and mixing parameter; and relative difference for the diameter, number of edges between outliers, and between outliers and clustered nodes. The test networks contain six real-world networks, each clustered using Leiden-CPM with $r = 0.01$ followed by CM.

RECCSv1). For diameter, RECCSv2 improves on the fit compared to SBM, but RECCSv1 is slightly worse. For the other criteria, the new pipelines have approximately the same accuracy as SBM.

Thus, the two pipelines—RECCSv1+Strategy 1 and RECCSv2+Strategy 1—both improve on SBM for edge cut sizes, with one clearly better suited for degree sequence and the other clearly better suited for diameter, and are nearly indistinguishable for the other properties. Results for the other test datasets (Fig. 4) show the same trends.

Finally, we compared the two pipelines and unmodified SBMs with respect to the *normalized edit distance* between the real-world network and the synthetic network they produce. For this distance, we use the number of edges that need to be added or removed from the real-world network, to produce the synthetic network, then normalized by the number of edges in the real-world network. Across the six networks in this test dataset, SBM and RECCSv2 are very close, while RECCSv2 produced synthetic networks that had a smaller normalized edit distance to the real-world network than RECCSv1 (Fig. 5). Furthermore,

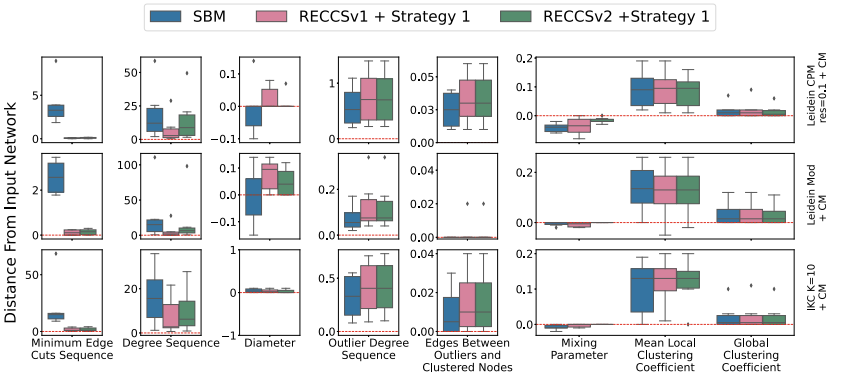

Fig. 4. Accuracy of SBM and Two RECCS pipelines on Test Data, using Three Additional Clusterings. The three additional clusterings are Leiden-CPM with $r = 0.1$ (top row), Leiden-modularity (middle row), and the Iterative k-core (IKC) method (bottom row). The y-axis shows different distance metrics for various network properties. Error is reported using RMSE for degree sequence, outlier degree sequence, and minimum edge cuts sequence; scalar difference is shown for clustering coefficients and mixing parameter; relative difference is shown for the diameter, number of edges between outliers, and between outliers and clustered nodes.

since the maximum normalized edit distance is 2.0, this shows that in all but one network, all three synthetic networks have a large enough distance to the real-world network to not simply replicate the real-world network. Thus, all three strategies—unmodified SBM and the two ways of post-processing the SBM—produce networks that are different from the real-world network and have good

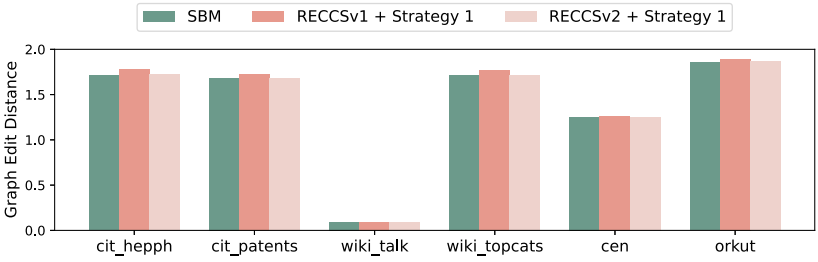

Fig. 5. Comparing SBM, RECCSv1, and RECCSv2 with respect to the normalized edit distance between synthetic and real world networks. The normalized edit distance between the edge sets of the true network G and the synthetic network N, i.e., $\frac{|E(G) \triangle E(N)|}{|E(G)|}$, where $\triangle$ denotes the symmetric difference, and so the maximum possible value is 2.0. Each real-world network is clustered using Leiden-CPM, with $r = 0.01$. Here, RECCSv2+Strategy 1 produces synthetic networks that are closer to the real-world network that RECCSv1+Strategy 1, and about as close as SBM networks.

fit for the network parameters we explored, while the two RECCS-pipelines also have a good fit for the cluster edge connectivity values but SBM does not.

4 Conclusion

Motivated by the need for synthetic networks that reproduce features of clustered real-world networks, we introduced the REalistic Cluster Connectivity Simulator (RECCS), which allows for variants of its basic two-phase approach and includes two strategies for adding in outlier nodes. We showed, using a diverse set of clustered real-world networks, that the RECCS pipelines that use outlier Strategy 1 produce synthetic networks that match or improve on the fit to empirical statistics of the clustered real-world networks compared to SBMs. Furthermore, the two versions of RECCS that we explore have different strengths, allowing for a range of synthetic networks to be developed.

In future work, we will explore a range of clustering methods on these synthetic networks in order to better characterize the conditions under which each method provides accuracy advantages over the other methods.

Funding Information. This work was supported in part by the Illinois-Insper Partnership.

Software. The software for the RECCS pipeline, including the different outlier strategies we explore, are available from GitHub [2].

References

1. Anne, L., Le-Vu, T.A., Park, M., Warnow, T., Chacko, G.: Supplementary materials (2024). https://doi.org/10.5281/zenodo.13367965
2. Anne, L., Warnow, T., Chacko, G.: Github page for RECCS (2024). https://github.com/illinois-or-research-analytics/lanne2_networks
3. Coscia, M., Giannotti, F., Pedreschi, D.: A classification for community discovery methods in complex networks. Stat. Anal. Data Min. ASA Data Sci. J. **4**(5), 512–546 (2011). https://doi.org/10.1002/sam.10133
4. El-Moussaoui, M., Agouti, T., Tikiniouine, A., Adnani, M.E.: A comprehensive literature review on community detection: approaches and applications. Procedia Comput. Sci. **151**, 295–302 (2019). https://doi.org/10.1016/j.procs.2019.04.042
5. Fortunato, S., Newman, M.E.J.: 20 years of network community detection. Nat. Phys. **18**(8), 848–850 (2022). https://doi.org/10.1038/s41567-022-01716-7
6. Henzinger, M., Noe, A., Schulz, C., Strash, D.: Practical minimum cut algorithms. J. Exp. Algorithmics (JEA) **23**, 1–22 (2018)
7. Javed, M.A., Younis, M.S., Latif, S., Qadir, J., Baig, A.: Community detection in networks: a multidisciplinary review. J. Netw. Comput. Appl. **108**, 87–111 (2018). https://doi.org/10.1016/j.jnca.2018.02.011
8. Kamiński, B., Prałat, P., Théberge, F.: Artificial benchmark for community detection (ABCD)-Fast random graph model with community structure. Netw. Sci. **9**(2), 153–178 (2021)

9. Kamiński, B., Prałat, P., Théberge, F.: Artificial benchmark for community detection with outliers (ABCD+ o). Appl. Netw. Sci. **8**(1), 25 (2023)
10. Lancichinetti, A., Fortunato, S.: Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. Phys. Rev. E **80**(1) (2009). https://doi.org/10.1103/physreve.80.016118
11. Leskovec, J., Sosič, R.: SNAP a general-purpose network analysis and graph-mining library. ACM Trans. Intell. Syst. Technol. (TIST) **8**(1), 1–20 (2016)
12. Muscoloni, A., Cannistraci, C.V.: A nonuniform popularity-similarity optimization (nPSO) model to efficiently generate realistic complex networks with communities. New J. Phys. **20**(5), 052002 (2018). https://doi.org/10.1088/1367-2630/aac06f
13. Park, M., et al.: Identifying well-connected communities in real-world and synthetic networks, pp. 3–14. Springer Nature Switzerland (2024). https://doi.org/10.1007/978-3-031-53499-7_1
14. Peel, L., Larremore, D.B., Clauset, A.: The ground truth about metadata and community detection in networks. Sci. Adv. **3**(5) (2017). https://doi.org/10.1126/sciadv.1602548
15. Peixoto, T.P.: The graph-tool python library. Figshare (2014). https://doi.org/10.6084/m9.figshare.1164194
16. Peixoto, T.P.: Bayesian stochastic blockmodeling. In: Doreian, P., Batagelj, V., Ferligoj, A. (eds.) Advances in Network Clustering and Blockmodeling, pp. 289–332. Wiley Online Library (2019)
17. Peixoto, T.P.: The Netzschleuder network catalogue and repository (2020). https://doi.org/10.5281/zenodo.7839980. Zenodo, 5201
18. Traag, V.A., Waltman, L., Van Eck, N.J.: From Louvain to Leiden: guaranteeing well-connected communities. Sci. Rep. **9**(1), 1–12 (2019)
19. Vaca-Ramírez, F., Peixoto, T.P.: Systematic assessment of the quality of fit of the stochastic block model for empirical networks. Phys. Rev. E **105**(5), 054311 (2022)
20. Viger, F., Latapy, M.: Efficient and simple generation of random simple connected graphs with prescribed degree sequence. J. Complex Netw. **4**(1), 15–37 (2016)
21. Wedell, E., Park, M., Korobskiy, D., Warnow, T., Chacko, G.: Center-periphery structure in research communities. Quant. Sci. Stud. **3**(1), 289–314 (2022)