Journal of Computational Science 78 (2024) 102283

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Analyzing modularity maximization in approximation, heuristic, and graph neural network algorithms for community detection

Samin Aref [a,*], Mahdi Mostajabdaveh [b]

[a] Department of Mechanical and Industrial Engineering, University of Toronto, M5S3G8, Canada
[b] Department of Mathematical and Industrial Engineering, Polytechnique Montreal, H3T1J4, Canada

ARTICLE INFO

Keywords:
Network science
Modularity maximization
Integer programming
Approximation
Graph neural network
Graph optimization

ABSTRACT

Community detection, which involves partitioning nodes within a network, has widespread applications across computational sciences. Modularity-based algorithms identify communities by attempting to maximize the modularity function across network node partitions. Our study assesses the performance of various modularity-based algorithms in obtaining optimal partitions. Our analysis utilizes 104 networks, including both real-world instances from diverse contexts and modular graphs from two families of synthetic benchmarks. We analyze ten inexact modularity-based algorithms against the exact integer programming baseline that globally optimizes modularity. Our comparative analysis includes eight heuristics, two variants of a graph neural network algorithm, and nine variations of the Bayan approximation algorithm.

Our findings reveal that the average modularity-based heuristic yields optimal partitions in only 43.9% of the 104 networks analyzed. Graph neural networks and approximate Bayan, on average, achieve optimality on 68.7% and 82.3% of the networks respectively. Additionally, our analysis of three partition similarity metrics exposes substantial dissimilarities between high-modularity sub-optimal partitions and any optimal partition of the networks. We observe that near-optimal partitions are often disproportionately dissimilar to any optimal partition. Taken together, our analysis points to a crucial limitation of the commonly used modularity-based methods: they rarely produce an optimal partition or a partition resembling an optimal partition even on networks with modular structures. If modularity is to be used for detecting communities, we recommend approximate optimization algorithms for a methodologically sound usage of modularity within its applicability limits. This article is an extended version of an ICCS 2023 conference paper (Aref et al., 2023).

1. Introduction

Community detection (CD), the data-driven process of inductively partitioning nodes within a network [1], is a core problem in computational sciences, particularly, in physics, computer science, biology, and computational social science [2]. Among common approaches for CD are the algorithms which are designed to maximize an objective function, modularity [3], across all possible ways that the nodes of the input network can be partitioned into communities. Modularity measures the fraction of edges within communities minus the expected fraction if the edges were distributed randomly; with the random distribution of the edges being a null model that preserves the node degrees. Despite their name and design philosophy, current modularity maximization algorithms, which are used by no less than tens of thousands of peer-reviewed studies [4], are not guaranteed to maximize modularity [5–7].

Modularity is among the first objective functions proposed for optimization-based community detection [3,8]. Several limitations [8–11] of modularity including the resolution limit [12–14] have led researchers to develop alternative CD methods using stochastic block modeling [15–18], information theoretic approaches [19,20], and alternative objective functions [21–24]. Modularity-based heuristics are the most commonly used methods for CD [2,25]. Besides modularity-based heuristics not guaranteeing the proximity to optimality, we do not know [6,26] the extent to which they succeed in returning maximum-modularity (optimal) partitions or similar partitions. Recently developed alternatives to these heuristics are neural network-based algorithms [27] and maximum modularity approximation algorithms [28] which use different approaches for maximizing modularity. Unlike modularity-based heuristics, approximation algorithms provide guarantees on the proximity to optimality.

Despite the availability of many modularity-based algorithms, the analysis of their performance in returning optimal partitions has not

* Corresponding author.
E-mail address: aref@mie.utoronto.ca (S. Aref).

https://doi.org/10.1016/j.jocs.2024.102283
Received 13 October 2023; Received in revised form 5 January 2024; Accepted 28 March 2024
Available online 8 April 2024

received sufficient attention [6,26]. In a previous study, Good et al. used one algorithm and three metabolic networks to show that high-modularity sub-optimal partitions may substantially differ with each other [26]. Our study is a continuation of a similar research path but it focuses on three fundamental and under-explored questions using an expanded scope of 104 networks, ten inexact algorithms, and an exact baseline method.

The contributions of this study are therefore three-fold. (1) It investigates the extent to which modularity maximization algorithms return optimal partitions. (2) It quantifies the cost of sub-optimal partitions in terms of their dissimilarity to the closest optimal partition using three similarity metrics. (3) It evaluates the performance of ten modularity-based algorithms on a structurally diverse set of real and synthetic networks. Our comparative analysis reveals the differences between these algorithms. We investigate several methods from different perspectives and our results show that some methods have certain advantages over the others. This study aims to help practitioners select suitable modularity-based algorithm to use given the specifics of their use case. We do not claim or assume that maximum-modularity partitions represent best partitions; neither do we claim that modularity is a silver bullet for community detection. Throughout the paper, we use the terms network and graph interchangeably. After reviewing the ten algorithms and describing the methods and materials, we present the results in five subsections followed by a discussion of the methodological ramifications and future directions.

2. Reviewing ten modularity-based algorithms

We evaluate ten modularity-based algorithms known as Clauset-Newman-Moore (CNM) [29], Louvain [30], Reichardt Bornholdt with the configuration model as the null model (LN) [31,32], Combo [25], Belief [33], Paris [34], Leiden [35], EdMot-Louvain [36], recurrent graph neural network (GNN) [27], and Bayan [28]. We briefly describe how these ten algorithms use modularity to discover communities.

CNM: The CNM algorithm initializes each node as a community by itself. It then follows a greedy scheme of merging two communities that contribute the maximum positive value to modularity [29].

Louvain: The Louvain algorithm involves two sets of iterative steps: (1) locally moving nodes for increasing modularity and (2) aggregating the communities from the first step [30]. Despite Louvain being the most commonly used modularity-based algorithm [4], it may sometimes lead to disconnected components in the same community [35].

Leiden: The Leiden algorithm attempts to resolve a defect of the Louvain algorithm in returning badly connected communities. It is suggested to guarantee well-connected communities in which all subsets of all communities are locally optimally assigned [35].

LN: The LN algorithm uses the same heuristic rules as the Leiden algorithm, but it supports weighted and directed graphs [32].

Combo: The Combo algorithm is a general optimization-based CD method which supports modularity maximization among other tasks. It involves two sets of iterative steps: (1) finding the best merger, split, or recombination of communities to maximize modularity and (2) performing a series of Kernighan–Lin bisections [37] on the communities as long as they increase modularity [25].

Belief: The Belief algorithm seeks the consensus of different high-modularity partitions through a message-passing algorithm [33] motivated by the premise that maximizing modularity can lead to many poorly correlated competing partitions.

Paris: The Paris algorithm is suggested to be a modularity maximization scheme with a sliding resolution [34]; that is, an algorithm capable of capturing the multi-scale community structure of real networks without a resolution parameter. It generates a hierarchical community structure based on a simple distance between communities using a nearest-neighbor chain [34].

EdMot: The EdMot-Louvain algorithm (EdMot for short) is developed to overcome the hypergraph fragmentation issue observed in previous motif-based CD methods [36]. It first creates the graph of higher-order motifs (small dense subgraph patterns) and then partitions it using the Louvain method to heuristically maximize modularity using higher-order motifs [36].

GNN: The GNN algorithm uses a recurrent graph neural network for maximizing modularity [27]. It relies on a continuous optimization technique that considers current node's *attachment*: continuous variable representing the cluster membership of a given node in a given community. In this algorithm, the attachments of nodes are combined with attachments of their neighbors. It starts with a random initial matrix of all attachments which is then updated iteratively to increase the modularity function using a recurrent graph neural network architecture [27]. We have used two variations of the GNN algorithm: GNN-100 (suggested to be the fastest version) and GNN-25K (suggested to be a slow but very precise version) [27].

Approximate Bayan: Unlike the algorithms discussed earlier, the approximate Bayan algorithm (Bayan for short) is an approximation algorithm for modularity maximization. Bayan is theoretically grounded by an Integer Programming (IP) formulation of the modularity maximization problem [38]. For approximating an optimal solution to the IP problem, Bayan uses a branch-and-cut scheme [28] while accounting for the gap between the upper bound and lower bound of the optimization problem. When the two bounds reach the desired approximation threshold (set by the user), the Bayan algorithm returns the partition with the highest modularity found alongside the maximum potential modularity gap (in percentage) that the returned partition may have from a globally maximum-modularity partition of the input graph.

Except for Bayan and GNN, we use the Python implementations of the remaining eight algorithms (collectively referred to as heuristics) which are accessible in the Community Discovery library (*CDlib*) version 0.2.6 [39]. For Bayan, we use the *bayanpy* version 0.7.6 library in Python. And we use the GNN as implemented in its public GitHub repository referenced in [27].

3. Methods and materials

In this paper, we evaluate eight modularity-based heuristics [25,29,30,32–36], two variations of a graph neural network algorithm [27], and nine variations of the approximate Bayan algorithm [28]. We quantify the extent to which these ten algorithms and their variations succeed in returning an optimal partition or a partition resembling an optimal partition.

To achieve this objective, we quantify the proximity of their results to the globally optimal partition(s), which we obtain using an exact Integer Programming (IP) model [38,40,41]. After describing the mathematical preliminaries, the IP model is discussed in Section 3.5.

3.1. Modularity matrix of a graph

Consider the simple (undirected and unweighted) graph $G = (V, E)$ with $|V| = n$ nodes, $|E| = m$ edges, and adjacency matrix entries a_{ij}. The modularity matrix of graph G is represented by $\mathbf{B} = [b_{ij}]$ whose entries are $b_{ij} = a_{ij} - \gamma d_i d_j / 2m$. In this formula, d_i represents the degree of node i and γ is the resolution parameter.[1]

3.2. Modularity of a partition

For graph $G = (V, E)$, consider the partition $X = \{V_1, V_2, \ldots, V_k\}$ of the node set V into any unspecified number k of (non-overlapping) communities. The modularity function $Q_{(G,X)}$, proposed by Newman [3],

[1] Without loss of generality, we set $\gamma = 1$ for all the analysis in this paper.

maps each partition of a graph to a real number in the range $[-0.5, 1]$ according to Eq. (1).

$$Q_{(G,X)} = \frac{1}{2m} \sum_{(i,j)\in V^2} \left(a_{ij} - \gamma \frac{d_i d_j}{2m} \right) \delta(i,j) \tag{1}$$

The modularity function $Q_{(G,X)}$ is based on the modularity matrix $\mathbf{B}$ of graph G and the partition X applied on the node set of graph G. In Eq. (1), the Kronecker delta, $\delta(i,j)$, is 1 if nodes i and j are in the same community according to partition X, otherwise it is 0.

3.3. Modularity maximization

The modularity maximization problem for the input graph $G = (V, E)$ involves finding a partition $X^*_{(G)}$ whose modularity is maximum over all possible partitions: $X^*_{(G)} = \arg\max_X Q_{(G,X)}$.

3.4. Optimal and sub-optimal partitions

For graph G, we refer to any partition that satisfies the definition of $X^*_{(G)} = \arg\max_X Q_{(G,X)}$ as an *optimal* partition (i.e., a maximum-modularity partition). Any partition that in not an optimal partition is a *sub-optimal* partition. Different sub-optimal partitions X_1, X_2 of graph G are distinguished based on their corresponding modularity values $Q_{(G,X_1)}, Q_{(G,X_2)}$ as well as by their similarity to an optimal partition of G. If G has multiple optimal partitions, we conservatively use the similarity to the optimal partition that is closest to the sub-optimal partition under evaluation. We use three different metrics (described in 3.6) for quantifying similarity to a reference partition.

3.5. Sparse IP formulation of modularity maximization

Consider the simple graph $G = (V, E)$ with modularity matrix entries b_{ij}, obtained using the resolution parameter γ. Consider the binary decision variable x_{ij} for each pair of distinct nodes $(i,j), i < j$. In a given partition, the community membership of the nodes i and j is either the same (represented by $x_{ij} = 0$) or different (represented by $x_{ij} = 1$). Accordingly, Dinh and Thai [38] have formulated the IP model in Eq. (2) for maximizing the modularity of input graph G.

$$\begin{aligned} \max_{x_{ij}} Q = \frac{1}{2m} & \left(\sum_{(i,j)\in V^2, i<j} 2b_{ij}(1 - x_{ij}) + \sum_{(i,i)\in V^2} b_{ii} \right) \\ \text{s.t.} \quad & x_{ik} + x_{jk} \geq x_{ij} \quad \forall (i,j) \in V^2, i < j, k \in K(i,j) \\ & x_{ij} \in \{0,1\} \quad \forall (i,j) \in V^2, i < j \end{aligned} \tag{2}$$

In Eq. (2), b_{ii} is the diagonal entry in row i and column i of the modularity matrix $\mathbf{B}$ for graph G which does not depend on the decision variables and is therefore separated from the off-diagonal entries. The optimal objective function value obtained from Eq. (2) equals the maximum modularity for the input graph G. An optimal community assignment is characterized by the values of the x_{ij} variables in an optimal solution to the IP model in Eq. (2). $K(i,j)$ indicates a minimum-cardinality *separating set* [38] for the nodes i, j.

Using $K(i,j)$ in the IP model of this problem leads to a more efficient formulation with $\mathcal{O}(n^2)$ constraints [38] instead of $\mathcal{O}(n^3)$ constraints as in earlier IP formulations of the problem [40,41]. Solving this optimization problem is NP-hard [7,40].

To obtain the baseline of our comparative analysis, we use the *Gurobi* solver (version 10.0) [42] to solve this NP-hard problem to global optimality for the small and mid-sized instances outlined in Section 3.8. For each network instance, we first obtain the optimal partitions by solving the IP model in Eq. (2) using the Gurobi solver and a termination criterion that ensures global optimality [42].

In the next step, we evaluate the ten modularity-based algorithms based on the proximity of their partitions to an optimal partition. On each network instance, we quantify the following for each algorithm (or algorithm variation): (1) the ratio of their output modularity to the maximum modularity for that network and (2) three measures of similarity between their output partition and an optimal partition of that network. These calculations lead to four values (GOP, AMI, RMI, and ECS) which are described in 3.6.

3.6. Measures for evaluating the algorithms

For a quantitative measure of proximity to global optimality, we define and use the *Global Optimality Percentage* (GOP) as the fraction of the modularity returned by an algorithm for a network divided by the globally maximum modularity for that network (obtained by solving the IP model in Eq. (2)). In cases where an algorithm returns a partition with a negative modularity value, we set GOP = 0 to facilitate easier interpretation of proximity to optimality based on non-negative GOP values.

We use three measures to quantify the similarity of a partition to an optimal partition. Two of them (AMI and RMI) are grounded in information theory and are shown to be reliable measures of partition similarity [43]. We use *adjusted mutual information* [44] and normalize it symmetrically [43]. The symmetrically normalized adjusted mutual information (AMI for short) [44] is a measure of similarity between two partitions of the same network. We also use *reduced mutual information* [45] and normalize it asymmetrically [43]. The asymmetrically normalized reduced mutual information (RMI for short) [45] is a measure of similarity between two partitions of the same network.

Unlike the commonly used [13,14,23,46–48] yet problematic [43–45,49] normalized mutual information (NMI) [44], AMI and RMI adjust the measurement based on the similarity that the two partitions may have by pure chance. AMI and RMI for a pair of identical partitions (or permutations of the same partition) equal 1. For two extremely dissimilar partitions, however, AMI and RMI take values close to 0.

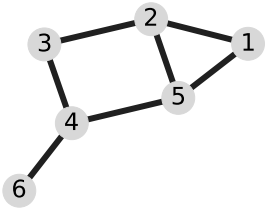
To ensure the reliability of our results on similarities of partitions, we also use the *Element-Centric Similarity* (ECS) as a third measure of partition similarity [49]. ECS differs from AMI and RMI in that it uses an alternative method for quantifying the similarity between two partitions that is grounded in common membership of nodes induced by the partition as opposed to overlaps between clusters [49]. We use ECS[2] because of the methodological advantages it offers compared to most commonly used metrics including the NMI, the Jaccard index, the Fowlkes-Mallows index, the adjusted Rand index, and the F-measure [49].

While the partition that maximizes modularity is often unique [50], some graphs have multiple optimal partitions. We obtain all optimal partitions of the networks using the Gurobi solver by running it with a special configuration for finding all optimal partitions [42]. In cases of networks with multiple optimal partitions, we calculate AMI, RMI, and ECS for the partition of each algorithm and each of the multiple globally optimal partitions of that graph. We then conservatively report the maximum AMI, maximum RMI, and maximum ECS of each algorithm on that network to quantify the similarity between that partition and its closest optimal partition. Consequently, a low value of AMI, RMI, or ECS reported for a partition indicates its dissimilarity to any optimal partition of that network.

3.7. Illustrative example

Fig. 1 shows a toy example of one graph partitioned by 19 different method to demonstrate sub-optimal and optimal partitions as well as values taken by modularity, GOP, AMI, RMI, and ECS. The graph shown in Fig. 1 has six nodes and seven edges. In our analysis, each network instance is partitioned by an exact method as well as ten modularity-based algorithms and their variations (19 inexact methods).

[2] For computing ECS, we use the value of 0.9 for its α parameter as suggested in [49] and used in the documentation of the CluSim Python library.



19 Methods (algorithms and their variations) and an exact method	Partition (obtained by each method)	Modularity	Optimality percentage (GOP)	Maximum similarity to any optimal partition AMI	RMI	ECS
Method 1		Q_1=0.122	Q_1/Q^*=0.60	0.36	0.28	0.64
Method 2		Q_2=0.122	Q_2/Q^*=0.60	0.36	0.28	0.64
...	...	...	...	...	...	...
Method 19		Q_{19}=0.092	Q_{19}/Q^*=0.45	0.30	0.23	0.56
Exact method (Integer Programming)	All optimal partitions including	Q^*=0.204	1	1	1	1

Fig. 1. A toy example demonstrating several sub-optimal and one optimal partition for a graph and the corresponding modularity, GOP, AMI, RMI, and ECS values. All information and values are related to the graph shown on top.

The first row in Fig. 1 shows the partition and values pertaining to method 1. Method 1 produces the partition [[1,2,3,5],[4,6]] in a failed attempt to maximize modularity for the input graph. The modularity of this sub-optimal partition equals $Q_1 = 0.122$. The same graph is also partitioned by solving the exact IP model. This method returns all optimal partitions of the input graph. In case of this graph, there is only one optimal partition which is [[1,2,5],[3,4,6]]. The modularity of this optimal partition equals $Q^* = 0.204$ which is the maximum value that the modularity function can possibly take for this input graph.

By comparing the results from method 1 and the exact method, four measures are calculated for method 1: GOP, AMI, RMI, and ECS. GOP for the partition obtained by method 1 equals $0.122/0.204 = 0.60$ which means the partition for method 1 is within 1-GOP= 40% of the maximum modularity. The AMI between the optimal partition and the partition obtained by method 1 equals 0.36. RMI and ECS are also calculated which reflect other perspectives on the extent of similarity between this sub-optimal partition and the optimal partition.

Note that for any inexact method that happens to return an optimal partition for a graph, GOP, AMI, RMI, and ECS will all take the value 1 which is the desired value for all four measures. In a practical setting, AMI, RMI, and ECS (as defined in this paper) are only available when an optimal partition is available which is not always the case. Like the differences between partitions and values for Method 1 and Method 19 in Fig. 1, Our comparative analysis reveals the differences between the performance of algorithms w.r.t maximizing modularity. Method 1 has relatively more success than Method 19 according to GOP, AMI, RMI, and ECS. This is consistent with the intuitions from visually inspecting the partitions in Fig. 1.

3.8. Specifications of network data and computing resources

For our computational experiments, we consider 54 real networks[3] from a wide range of contexts and domains including online and offline social relations, social affiliation, social collaboration, animal interactions, biological, neural, informational, technological, fictional, geographical, organizational, communications, and terrorism. We also use 50 structurally diverse synthetic networks that have modular structures.

To create synthetic graphs with modular structures, we use two benchmark graph generation models: *Lancichinetti-Fortunato-Radicchi* (LFR) benchmark graphs [51] and *Artificial Benchmarks for Community Detection* (ABCD) graphs [52]. These two families of synthetic benchmark graphs are designed for evaluating the performance of CD algorithms based on their success in retrieving a planted (ground-truth) partition (see [14,28,47] for the common use case). However, we deploy these two models for generating synthetic graphs with controllable modular structures and do not use the planted partition information. LFR and ABCD each has a distinct mixing parameter which controls the association between the structure and the planted communities. This association in turn impacts the strength of the modular structure (relatively higher density of intra-community edges compared to the density of inter-community edges). ABCD is the more recent alternative to the LFR model and offers additional benefits including higher scalability and better control for adjusting the mixing parameter [52].

[3] The 54 real networks are accessible from the Netzschleuder repository with the details in the Appendix.

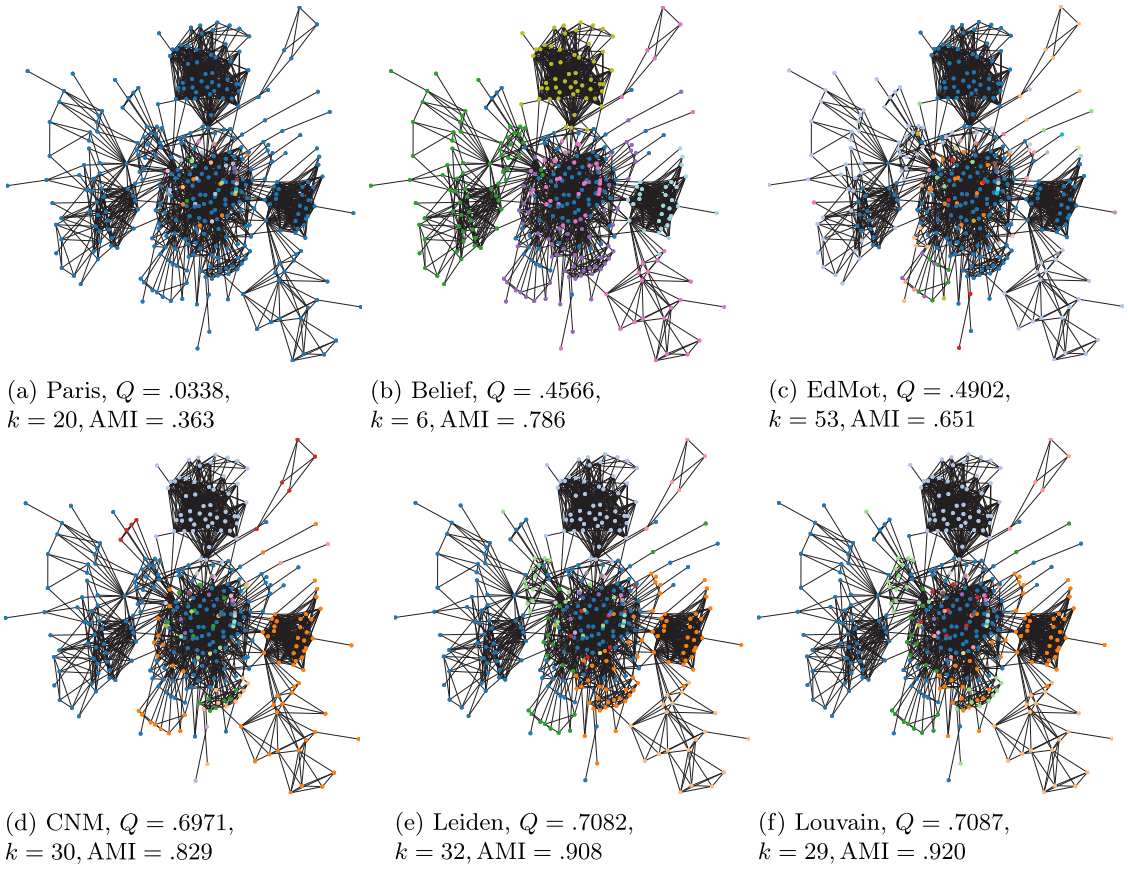


Fig. 2. Modularity maximization for one network using six methods leading to six sub-optimal partitions (panels a–f) with increasing Q, different k, and different AMI values. Only the giant component is shown. (Magnify the high-resolution color figure on screen for more details.)

The 50 synthetic networks include 20 LFR graphs and 30 ABCD graphs with up to 1000 edges. The real networks considered have at most 2812 edges. These instance sizes were chosen to ensure that globally optimal partitions can be obtained in a reasonable time. Publicly available links and additional details for both real and synthetic networks are provided in the Appendix.

The computational experiments were implemented in Python 3.9 using a notebook computer with an Intel Core i7-11800H @ 2.30 GHz CPU and 64 GB of RAM running Windows 10.

4. Results

We present the main results from our experiments in the following five subsections. In Section 4.1, we compare partitions from 12 modularity maximization methods on a single network to illustrate the differences between different methods (algorithms and their variations) for solving the same underlying optimization problem. In Section 4.2, we use AMI, RMI, and ECS to investigate the cost of sub-optimality in terms of dissimilarity of partitions from an optimal partition. In Section 4.3, we summarize the distributions of AMI, RMI, and ECS for each algorithm on all 104 networks. In Section 4.4, we compare the solve times of all the algorithms and their variations. Finally, in Section 4.5, we investigate the success rate of all the algorithms and their variations in returning an optimal partition.

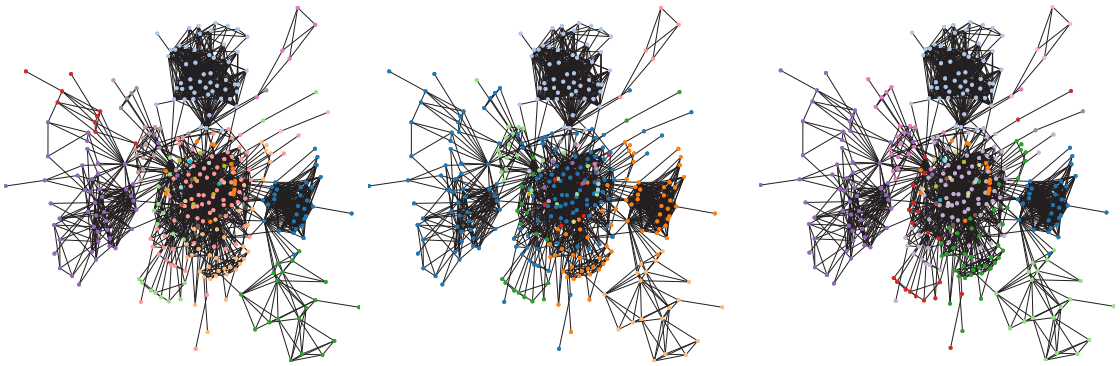
4.1. Comparing partitions from different algorithms on one network

Figs. 2–3 show the largest connected component (the giant component) of one network that is partitioned by twelve modularity-based CD methods. This network[4] represents an anonymized Facebook *egocentric network*[5] from which the ego node has been removed. Nodes represent Facebook users, and an edge exists between any pair of users who were friends on Facebook in April 2014 [53]. Partitions of nodes into communities are shown using node colors.

Comparing Figs. 2–3, the partitions from the six algorithms in Fig. 2 have more substantial differences from the optimal values in Q, AMI, and k (number of communities) as shown by the values in the corresponding subcaptions in Fig. 2. Fig. 3(f) shows an optimal partition of the network obtained using the Bayan approximate algorithm with an approximation tolerance of $1e-3$. This optimal partition has $k = 28$ communities, and a modularity value of $Q^* = 0.7157714$ (the maximum modularity for this network). The partitions from the all the other eleven methods are sub-optimal. Compared to other heuristics, the two heuristics Combo and LN have more success in achieving proximity to an optimal partition. LN returns a partition with $k = 28$ communities and a modularity of $Q = 0.7153755$ which has the highest AMI among all heuristics (0.971). The relative success of the Combo algorithm is in returning a particularly high-modularity partition with $Q = 0.7157709$, but with $k = 13$ communities and a lower AMI (0.949) compared to the AMI of LN. The two variations of the GNN algorithm return sub-optimal partitions with 19 and 16 communities. Similar observations can be made from the RMI and ECS values of these partitions which are not reported in the interest of brevity.

[4] *facebook_friends* network [53] from the Netzschleuder repository.

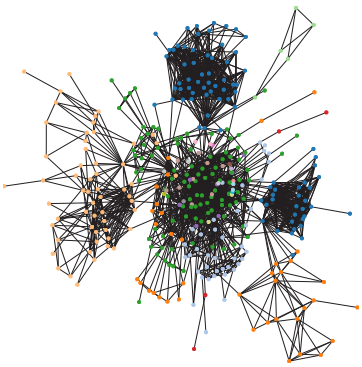
[5] A network of one person's social ties to other persons and their ties to each other.



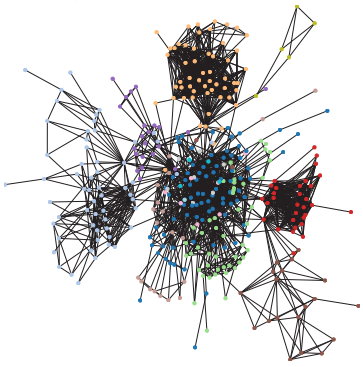
(a) GNN-100, $Q = .713294$, $k = 19$, AMI $= .941$

(b) LN, $Q = .7153755$, $k = 28$, AMI $= .971$

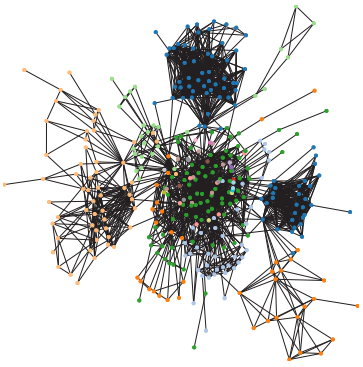
(c) GNN-25K, $Q = .715628$, $k = 16$, AMI $= .957$



(d) Bayan 0.9, $Q = .7156751$, $k = 30$, AMI $= .987$



(e) Combo, $Q = .7157709$, $k = 13$, AMI $= .949$



(f) Bayan 1e-3, $Q^* = .7157714$, $k = 28$, AMI $= 1$

Fig. 3. Modularity maximization for one network using six methods leading to one optimal partition (panel f) and five sub-optimal partitions (panels a-e) with increasing Q, different k, and different AMI values. Only the giant component is shown. (Magnify the high-resolution color figure on screen for more details.)

4.2. The disproportionate cost of sub-optimality

We use three scatter-plots in Figs. 4–6 to investigate the cost of sub-optimality from the three perspectives of AMI, RMI, and ECS (in terms of dissimilarity from an optimal partition) for each of the algorithms based on all the 104 networks.

Fig. 4 has one panel for each algorithm which shows GOP on the y-axis and AMI on the x-axis. Each of the 104 data points in a panel of Fig. 4 corresponds to one of the 104 networks. The 45-degree lines in Fig. 4 indicate the areas where the GOP and AMI are equal. In other words, the 45-degree lines show areas where the extent of sub-optimality (1-GOP) is associated with a dissimilarity (1-AMI) of the same proportion between the sub-optimal partition and the closest optimal partition.

Looking at the y-axes values in Fig. 4, we observe that there is a substantial variation in the values of GOP (i.e., variations in the extent of sub-optimality) for most of the eight heuristic algorithms. The Belief algorithm returns partitions associated with negative modularity values for 22 of the 104 instances (leading to the corresponding data points having GOP= 0 and being concentrated at the bottom of the scatter-plot). The Paris and EdMot algorithms return partitions with modularity values substantially smaller than the maximum modularity values. Among the eight heuristics, the four algorithms with the highest and increasing performance in returning close-to-maximum modularity values are LN, Leiden, Louvain, and Combo. Despite that these instance are graphs with no more than 2812 edges, they are, according to Fig. 4, challenging instances for these heuristic algorithms to optimize. Given that modularity maximization is an NP-hard problem [7,40], one can argue that the performance of these heuristics, in achieving proximity to an optimal partition, does not improve for larger networks. The y-axes values for different variations of Bayan and GNN have a much lower variability and are closer (than partitions of most heuristics) to 1. This indicates that these two algorithms return partitions with modularity values closer to the maximum modularity values of these networks.

The x-axes values for the heuristics in Fig. 4, except Combo, show considerable dissimilarity (from an AMI perspective) between the sub-optimal partitions of these heuristics and any optimal partition for these 104 instances. Some sub-optimal partitions obtained by these heuristics have AMI values smaller than 0.5. These are substantially different from any optimal partition. Even for the data points concentrated at the top of the scatter-plots which have $0.95 < \text{GOP} < 1$, we see some substantially small AMI values. They indicate that some high-modularity partitions are particularly dissimilar to any optimal partition. Compared to the other seven heuristics, Combo appears to consistently return partitions with high AMIs on a larger number of these 104 instances. The twelve panels for Combo and different variations of GNN and Bayan show fewer instances of low AMI values indicating that these three algorithms are overall more successful at returning partitions highly similar to an optimal partition. The panels for Bayan show that decreasing the approximation tolerance (gradually from 0.9 to $1e{-}5$) leads to a gradual increase in the resulting AMI values. Unlike heuristics whose performance cannot be controlled through a user-specified parameter, Bayan provides the user with the flexibility to obtain approximations closer to optimal by reducing the tolerance (at the cost of additional computations).

The most important pattern in Fig. 4 is observed when we focus on the positions of data points with respect to the 45-degree lines. We observe that the data points are mostly located above their corresponding 45-degree line. This indicates that, irrespective of the algorithm, sub-optimal partitions tend to be disproportionately dissimilar to any optimal partition (as foreshadowed in [54]). This result goes against

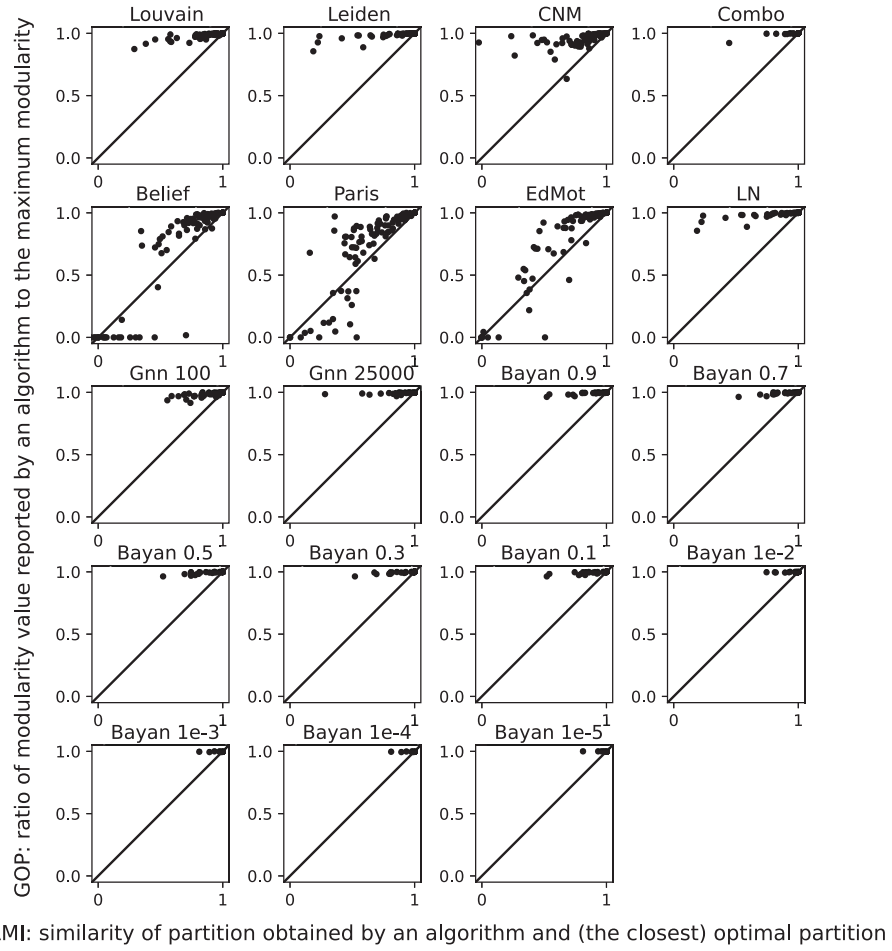


Fig. 4. Global optimality percentage (GOP) and normalized adjusted mutual information (AMI) measured for each algorithm by comparing its results with (all) globally optimal partitions. (Magnify the high-resolution figure on screen for more details.)

the naive viewpoint that close-to-maximum modularity partitions are also close to an optimal partition. Our results are aligned with previous concerns that modularity maximization heuristics have a high risk of algorithmic failure [6] and may result in degenerate solutions far from the underlying community structure [26].

To ensure that the observations made are not artefacts of using AMI, we also report the RMI and ECS values of the same partitions. Figs. 5–6 have x-axes values based on RMI and ECS respectively and show GOP on their y-axes. Similar observations can be made from the RMI and ECS values of these partitions: (1) x-axes values in Figs. 5–6 for partitions of all heuristics except Combo show substantial dissimilarity to any optimal partitions of these networks. (2) GNN, Combo, and Bayan return partitions with higher similarity to the optimal partitions, also when RMI or ECS are used. (3) Most data point are above their corresponding 45-degree lines indicating that sub-optimal partitions tend to be disproportionately dissimilar to any optimal partition (from RMI and ECS perspectives).

4.3. Distribution of partition similarity measures for each algorithm

In the scatter-plots Figs. 4–6, data points overlap with each other and therefore distributions are not visible. Fig. 7 complements the observations made from Figs. 4–6. Fig. 7 illustrates for each algorithm the box plots of AMI, RMI, and ECS values, obtained on all 104 networks. Each box in Fig. 7 shows: the first quartile (Q_1), the median (Q_2), and the third quartile (Q_3) of the distribution for one similarity measure and one algorithm. The whiskers are drawn from the nearest hinge (Q_1 or Q_3) to the farthest datapoint within the 1.5 interquartile range ($\pm 1.5(Q_3 - Q_1)$).

The distributions for the three measures AMI, RMI, and ECS are quite similar reaffirming that the differences between algorithms observed in Section 4.2 are irrespective of the choice of partition similarity measure. The alignment between our AMI and RMI results is consistent with the results in [43] while that study recommends using RMI.

Belief, Edmot, and Paris have the three widest distributions for all three similarity measures. For Paris, the medians of all three measures are below 0.8 indicating that its failure (in returning partitions that are at least 80% similar to optimal) happens on half of these instances. The median ECS for Belief is also below 0.8 which can be interpreted similarly. For CNM and EdMot, the medians are around 0.85 and 0.9 showing the same issue but to a lesser degree.

All the distributions are left-skewed indicating higher variability among values below the median. Compared to the other heuristics, Louvain, Leiden, LN, and Combo have distributions with smaller ranges and higher medians. Both variations of the GNN algorithm have wider distributions than Combo. The nine variations of the Bayan algorithm have medians extremely close to 1 with some of them also having the smallest ranges among all the algorithms considered. We reobserve the expected pattern that reducing the approximation threshold of Bayan (from 0.9 to $1e-5$) generally leads to better performance (higher similarity to an optimal partition with lower variation).

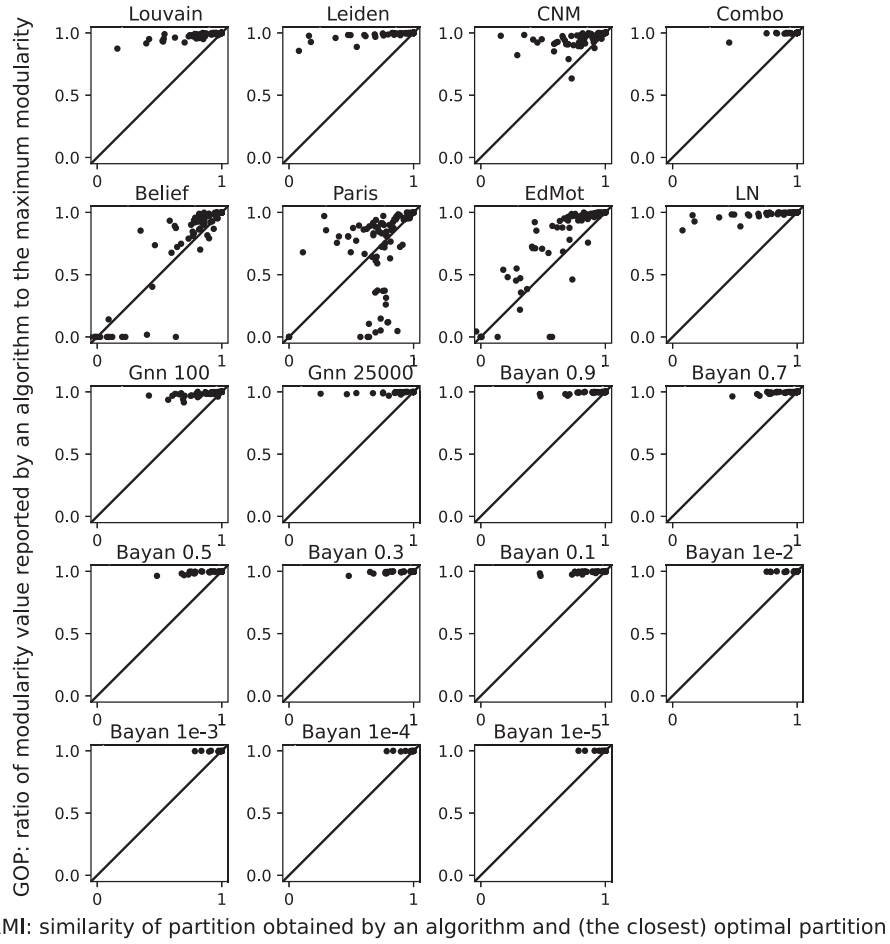


Fig. 5. Global optimality percentage (GOP) and normalized reduced mutual information (RMI) measured for each algorithm by comparing its results with (all) globally optimal partitions. (Magnify the high-resolution figure on screen for more details.)

4.4. Empirical time comparison of the algorithms

All these algorithms attempt to solve the same optimization problem: maximizing modularity, but their ways of exploring the feasible space are widely different leading to considerably different solve times for their specific computations (which are inherently unalike). Fig. 8 shows for each algorithm a box plot of its empirical solve times measured on the 104 networks. Note that the y-axis of Fig. 8 is on a logarithmic scale. These solve times are empirical and depend on the computing resources used, but are comparable to each other because the same computing resources were used for all the algorithms.

Box plots of two algorithms stand out: GNN-25K has the longest solve time (and a median of 247 s) followed by the Belief algorithm (that has a median of 8.4 s). The median solve time of GNN-100 (0.56 s) and that of five Bayan variations are similar (those with approximation tolerances from 0.1 to 0.9). Other variations of Bayan have a median solve time above one second with the slowest variation (Bayan $1e-5$) having a median of 1.97 s. The distributions of solve times for all variations of Bayan are left-skewed with their Q_1 solve time often being smaller than their median solve time by an order of magnitude. The widest boxes belong to the solve times of different Bayan variations for which the Q_3 is often an order of magnitude larger than Q_2. Except for Belief, the heuristic algorithms are 1–2 orders of magnitude faster than GNN and Bayan; with Leiden and LN being the fastest algorithms in our analysis.

The solve time distributions of all algorithms have outliers (values larger than the top whisker of the box plot). All 104 instances considered are networks with tens to a few thousands of edges. Therefore, the outliers existing for almost all algorithms indicates that solving some instances take much longer than the typical solve time of that algorithm for that range of input size. This can be partially explained by the differences in graph structures. Some graphs have a structure close to the structure of a random graph (and far from a modular structure). Finding a high-modularity partition for such graphs takes orders-of-magnitude longer (than the typical time for networks of that size range) irrespective of the algorithm used.

Fig. 9 shows the same solve times, but with different arrangement and visualization. It shows how the average empirical solve time of each algorithm changes as the input size increases. For Fig. 9, we group the 104 networks into four bins based on their graph sizes (number of edges) and plot the average solve time of each algorithm for each of the four bins of instances. Note that the y-axis in Fig. 9 is in logarithmic scale. It can be observed that GNN-25k has the highest solve time among all algorithms; on average, it takes over 1000 s for graphs with more than 750 edges.

The solve time of these algorithms are generally increasing with input size, but Fig. 9 does not show a monotonic increase for some methods like Bayan 1e−5 and Bayan 1e−4. In case of Bayan, this can be partly explained by considering that some larger instances in our analysis have modular structures that have facilitated the exploration of Bayan for approximating an optimal solution. Different variations of Bayan, Belief, and GNN-100 take order-of-magnitude longer than other modularity-based algorithms. At the other extreme, LN and Leiden are the fastest among all algorithms in our analysis. Counting the horizontal gridlines in Fig. 9, there are five orders of magnitude difference in empirical solve time between the slowest (GNN-25k) and the fastest

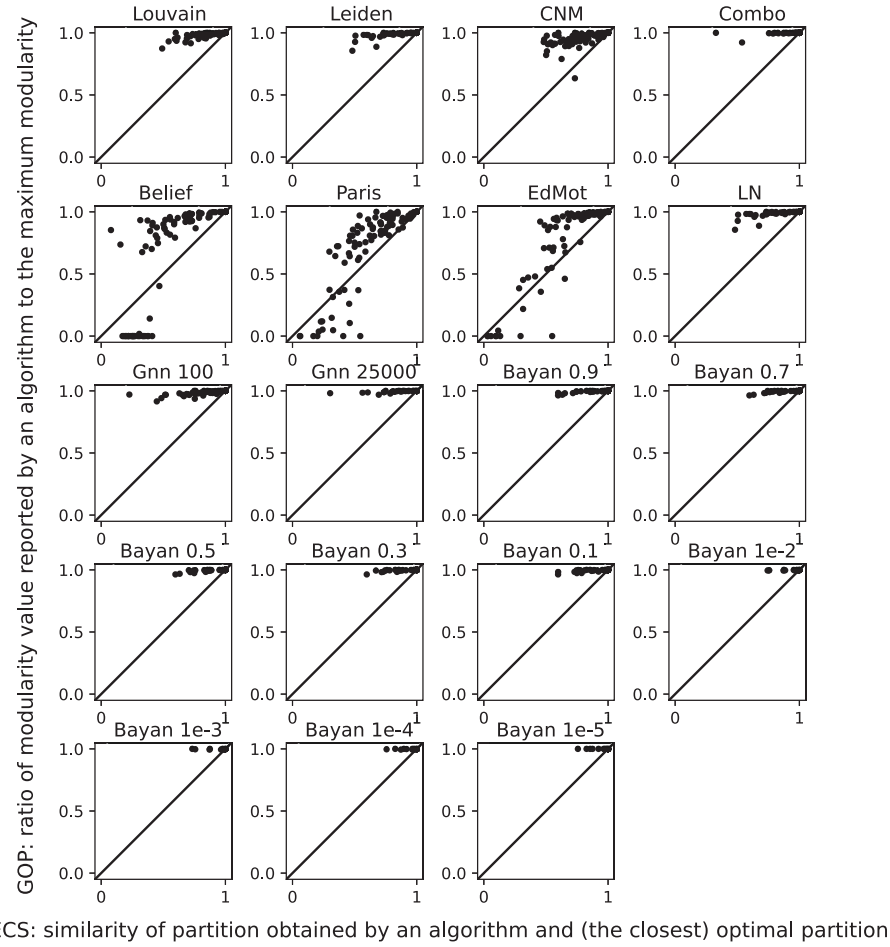


Fig. 6. Global optimality percentage (GOP) and element-centric similarity (ECS) measured for each algorithm by comparing its results with (all) globally optimal partitions. (Magnify the high-resolution figure on screen for more details.)

(Leiden and LN) algorithms for processing instances of comparable sizes.

4.5. Success rates of the algorithms in maximizing modularity

The GOP values help us answer a fundamental question about these modularity-based algorithms: how often does each algorithm return an optimal partition? We report the fraction of networks (out of 104) for which a given algorithm returns an optimal partition. A similar assessment of some of these algorithms on a different set of networks is provided in [50].

Fig. 10 shows the success rates of all algorithms and their variations in achieving global optimality on the 104 networks considered. Among the eight heuristics, Combo has the highest success rate, returning an optimal partition for 90.4% of the networks. The average success rate of all 8 heuristics combined is 43.9%. With the exception of Combo, the rates for these heuristics are arguably low success rates for what the name *modularity maximization algorithm* implies or the idea of discovering network communities through maximizing a function. The two variations of GNN have markedly different success rates and their average success rate is 68.7%. The least restricted version of approximate Bayan (Bayan 0.9) returns optimal partitions on 75% of networks and is more successful than the average heuristic and the average GNN variation. Bayan's success rate increases to 91.3% when a sufficiently small approximation threshold is chosen. This indicates that in the context of solving this NP-hard graph optimization problem, an optimization procedure (branch and cut which is used in Bayan) can be more successful than other existing alternatives (e.g., heuristics and GNNs). After all, modularity maximization is a mathematical optimization task and therefore it is safe to assume that guaranteed approximate optimization methods would likely be hard to outperform in optimality success rate; acknowledging that they most likely take longer and can only be used for small and mid-sized networks. Fig. 10 also shows that using Bayan with a smaller approximation tolerance value often leads to a higher success rate.

Earlier in Figs. 4–6, we observed that near-optimal partitions tend to be disproportionately dissimilar to any optimal partition. In other words, close-to-maximum modularity partitions are rarely close to any optimal partition. Taken together with the low success rates of the average heuristic in maximizing modularity, our results indicate a crucial mismatch between the design philosophy of these methods and their average capability: most heuristic modularity maximization algorithms rarely return an optimal partition or a partition resembling an optimal partition even on graphs with modular structures.

5. Discussions and future directions

Understanding modularity capabilities and limitations has been complicated by the under-explored performance of modularity maximization algorithms on the task that is literally in their name: modularity maximization. Previous methodological studies [11,55–57], which have shed light on other aspects, have rarely disentangled the unguaranteed aspect of the inexact optimization from the fundamental concept of modularity. Our study is a continuation of previous efforts [26] in separating the effects of sub-optimality (or the choice of using greedy

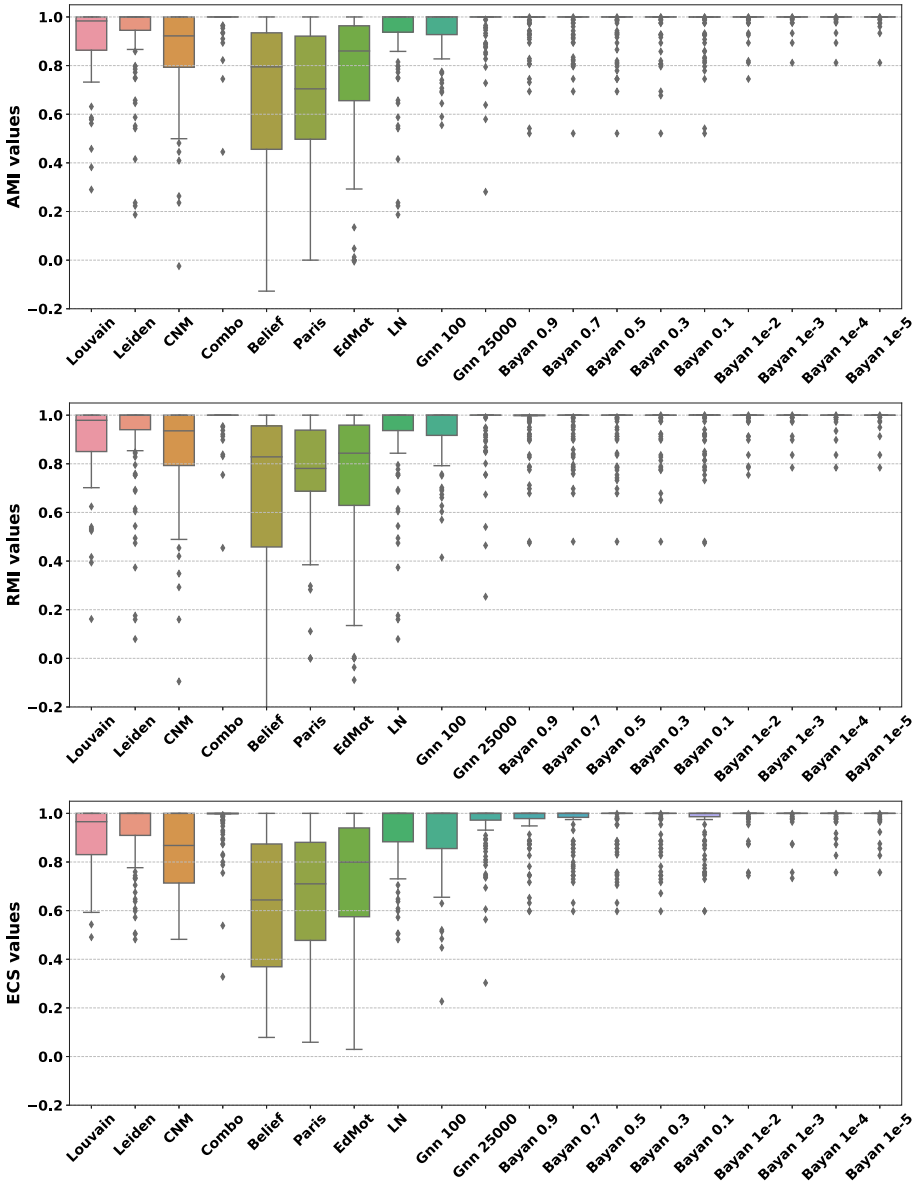


Fig. 7. The box plots of similarity to an optimal partition (AMI in the top panel, RMI in the center panel, and ECS in the bottom panel) for each algorithm based on all 104 network instances. (Magnify the high-resolution figure on screen for more details.)

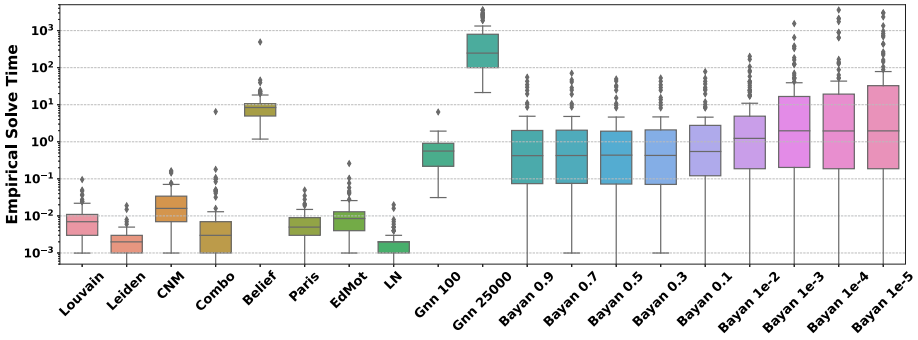


Fig. 8. Box plots representing the empirical solve time of the algorithms for the 104 networks. The y-axis is in logarithmic scale. (Magnify the high-resolution color figure on screen for more details.)

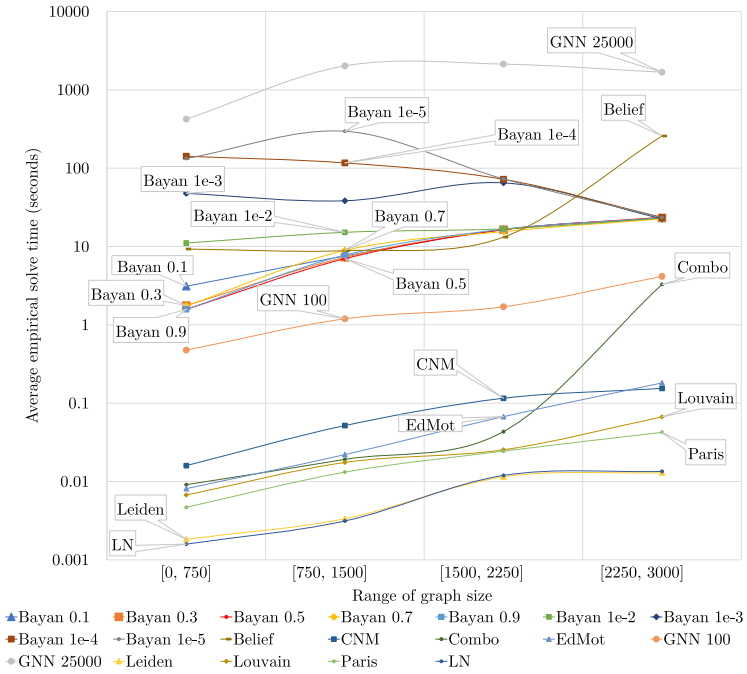


Fig. 9. Average solve time of different methods based on bins of input size. The *y*-axis is in logarithmic scale. (Magnify the high-resolution color figure on screen for more details.)

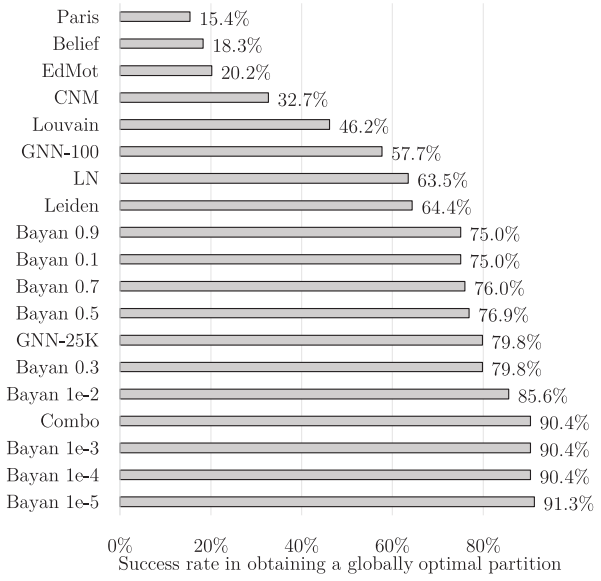


Fig. 10. Ten algorithms and their variations ranked based on their success rate in achieving global optimality on the 104 networks considered.

algorithms [6]) from the effects of using modularity on the fundamental task of detecting communities.

We analyzed the effectiveness of eight heuristics [25,29,30,32–36], two variations of a GNN [27], and several variations of an approximation algorithm [28] in maximizing modularity. While our findings are limited to only a handful of algorithms, their combined usage by tens of thousands of peer-reviewed studies [4] motivates the importance of this assessment. Most heuristic algorithms for modularity maximization tend to scale well for large networks [58]. They are widely used not only because of their scalability or ease of implementation [6], but also because their high risk of algorithmic failure is not well understood [6]. The scalability of these heuristics comes at a cost: their partitions have no guarantee of proximity to an optimal partition [26] and, as our results showed, the average heuristic rarely returns an optimal partition even on modular graph structures. Moreover, we showed that their sub-optimal partitions tend to be disproportionately dissimilar to any optimal partition irrespective of the partition similarity metric chosen.

Our approach of quantifying similarity between partitions using AMI, RMI, and ECS has some merits over our earlier study [50], but it still has some limitations. An alternative approach involves identifying *building blocks* across different candidate partitions of a network [59]. These building blocks are groups of nodes that are usually found together in the same community. Riolo and Newman propose a method for finding building blocks and suggest that building blocks obtained in their results are largely invariant for a given network while different arrangements of the same building blocks lead to different partitions of that network [59]. While this is a viable explanation of some of the variations between some community detection algorithms, our results do not fully match with this interpretation of partition dissimilarities. For example, the communities (shown by colors) in Fig. 2 can only be interpreted as a specific arrangement of very small building blocks including blocks made up of a single node. In this case with such building block sizes, finding a suitable arrangement of these extremely small building blocks is arguably the whole task of community detection on which modularity-based algorithms perform differently.

Neither using modularity nor succeeding in maximizing it is required for CD at the big-picture level. A common narrative in the literature is debating whether modularity is suitable or not [11,60]. We argue that such a debate is an oversimplification because suitability of using modularity depends on the task[6], the context, and several other factors including (1) whether it used as an objective function [38] or as a partition quality function [60]; (2) how the maximization is operationalized; and (3) what advantages are offered by the alternatives to use instead of modularity. Our results shed light on the first two questions, but are not related to alternative methods that do not use modularity. A recent study claims modularity maximization is the most problematic CD method and considers it harmful [11].

[6] See [1] for four widely different tasks that are all referred to as community detection.

Another study shows that, given computational feasibility, exact maximization of modularity outperforms 30 other CD methods in accurate and stable retrieval of ground-truth communities in both LFR and ABCD benchmarks [28] suggesting the relevance of modularity for CD.

Our results were based on small and mid-sized networks with no more than 2812 edges. We showed the extent to which each modularity-based method succeeds in returning optimal partitions or partitions resembling optimal partitions. Given that modularity maximization is an NP-hard problem, it is not reasonable to expect that the performance of these inexact methods suddenly increases for large-scale networks. The extent of their failures on large-scale networks is not quantified yet. However, the average success rate of 43.9% suggests the performance in maximizing modularity that can be expected from these heuristics. This expectation is realistic for small and mid-sized networks, and it is arguably optimistic for large-scale networks. Our findings suggest that if modularity is to be used for detecting communities, using approximation [28,54,61,62] and exact [28,63,64] algorithms is recommendable for a more methodologically sound usage of modularity within its applicability limits.

A promising path forward could be using the advances in integer programming to develop better approximation algorithms (outperforming approximate Bayan) for solving the mathematical models of modularity maximization [38,40,41] on networks of practical relevance within the limits of computational feasibility. New heuristic algorithms that strike a balance between accurate computations and scalability (achieving or surpassing the performance of methods like Combo and Leiden, but with higher scalability) may also be useful particularly for large-scale networks.

CRediT authorship contribution statement

Samin Aref: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Mahdi Mostajabdaveh:** Formal analysis, Methodology, Resources, Software, Supervision, Validation, Visualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Authors are thankful to the three anonymous referees of ICCS-2023 and to the two anonymous referees of this special issue for their helpful comments on earlier version of this article. We also thank Max Jerdee for providing his RMI code and we acknowledge Santo Fortunato, Mark Newman, and Tiago P. Peixoto for the helpful correspondence and discussions.

Appendix

Justifying the use of AMI, RMI, and ECS and the exclusion of other popular measures of partition similarity

We use AMI, RMI, and ECS because they are shown to be reliable measures of partition similarity [43–45,49].

We avoid using the Normalized Mutual Information (NMI) because, despite its common use, several studies indicate that its usage leads to incorrect assessments [44,45,49] and incorrect evaluation of competing algorithms [43]. For example, consider two arbitrarily dissimilar partitions A and B, each made up of two communities and their NMI taking the value 0.1 (for the sake of argument). If we take partition B and split each of its communities into several communities, the NMI between the resulting partition and partition A will increase; and this increase is monotonic with the more we split partition B [49]. Despite that this undesirable bias of the NMI towards the number of clusters has been well documented for several years [49], NMI is still among the most popularly used [13,14,23,46–48] metrics for quantifying the similarity of partitions [43].

Similar to the NMI, other popularly used measures of partition similarity suffer from at least one form of undesirable bias [49]. We also avoid using any of the four measures: the Jaccard index, the Fowlkes-Mallows index, the adjusted Rand index, and the F-measure because they suffer from a bias that favors skewed cluster sizes [49].

Accessing the data for real and synthetic networks

The data on the 50 LFR and ABCD graphs used in this study are available in a *FigShare* data repository [65].

The LFR benchmarks used in this study were randomly generated based on the following parameters: number of nodes (n) randomly chosen from the range $[20, 300]$, maximum degree $\lfloor 0.3n \rfloor$, maximum community size $\lfloor 0.5n \rfloor$, power law exponent for the degree distribution $\tau_1 = 3$, power law exponent for the community size distribution $\tau_2 = 1.5$, and average degree of 4. The parameter μ (LFR mixing parameter) was chosen from the set $\{0.01, 0.1\}$ (10 LFR graphs for each value of μ).

The ABCD benchmarks were randomly generated based on the following parameters: number of nodes (n) randomly chosen from the range $[10, 500)$; minimum degree d_{min} and minimum community size k_{min} randomly chosen from the range $[1, n/4)$; maximum community size chosen randomly from $[k_{min} + 1, n)$; maximum degree chosen randomly from $[d_{min} + 1, n)$; and power law exponents for the degree distribution and community size distribution randomly from $(1, 8)$ and then rounded off to 2 decimal places. The parameter ξ (ABCD mixing parameter) was chosen from the set $\{0.01, 0.1, 0.3\}$ (10 ABCD graphs for each value of ξ).

The 54 real networks were loaded as simple unweighted and undirected graphs. They are available in the publicly accessible network repository Netzschleuder with the 54 names below:

dom, packet_delays, sa_companies, ambassador, florentine_families, rhesus_monkey, kangaroo, internet_top_pop, high_tech_company, moviegalaxies, november17, moreno_taro, sp_baboons, bison, dutch_school, zebras, cattle, moreno_sheep, 7th_graders, college_freshmen, hens, freshmen, karate, dutch_criticism, montreal, ceo_club, windsurfers, elite, macaque_neural, sp_kenyan_households, contiguous_usa, cs_department, dolphins, terrorists_911, train_terrorists, highschool, law_firm, baseball, blumenau_drug, lesmis, sp_office, polbooks, game_thrones, football, football_tsevans, sp_high_school_new, revolution, student_cooperation, interactome_pdz, physician_trust, malaria_genes, marvel_partnerships, facebook_friends, netscience

For more information on each network and its original source, one may check the Netzschleuder website by adding the network name at the end of the url: https://networks.skewed.de/net/. For example, https://networks.skewed.de/net/malaria_genes provides additional information for the *malaria_genes* network. In cases of multiple networks existing with the same name in Netzschleuder, we have only used the lexicographically first network (e.g. we have only used the *HVR_1* network from https://networks.skewed.de/net/malaria_genes).

References

[1] M.T. Schaub, J.C. Delvenne, M. Rosvall, R. Lambiotte, The many facets of community detection in complex networks, Appl. Netw. Sci. 2 (1) (2017) 1–13.
[2] S. Fortunato, M.E.J. Newman, 20 Years of network community detection, Nat. Phys. 18 (2022) 848–850.
[3] M.E.J. Newman, Modularity and community structure in networks, Proc. Natl. Acad. Sci. 103 (23) (2006) 8577–8582, http://dx.doi.org/10.1073/pnas.0601602103.

[4] A. Kosowski, D. Saulpic, F. Mallmann-Trenn, V.P. Cohen-Addad, On the power of Louvain for graph clustering, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems 33, NeurIPS'20, 2020.

[5] M.E.J. Newman, Equivalence between modularity optimization and maximum likelihood methods for community detection, Phys. Rev. E 94 (5) (2016) 052315, http://dx.doi.org/10.1103/PhysRevE.94.052315.

[6] T. Kawamoto, Y. Kabashima, Counting the number of metastable states in the modularity landscape: Algorithmic detectability limit of greedy algorithms in community detection, Phys. Rev. E 99 (1) (2019) 010301.

[7] K. Meeks, F. Skerman, The parameterised complexity of computing the maximum modularity of a graph, Algorithmica 82 (8) (2020) 2174–2199.

[8] S. Fortunato, D. Hric, Community detection in networks: A user guide, Phys. Rep. 659 (2016) 1–44, http://dx.doi.org/10.1016/j.physrep.2016.09.002.

[9] R. Guimerà, M. Sales-Pardo, L.A.N. Amaral, Modularity from fluctuations in random graphs and complex networks, Phys. Rev. E 70 (2004) 025101.

[10] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (3–5) (2010) 75–174.

[11] T.P. Peixoto, Descriptive vs. inferential community detection in networks: Pitfalls, myths and half-truths, in: Elements in the Structure and Dynamics of Complex Networks, Cambridge University Press, 2023.

[12] S. Fortunato, M. Barthélemy, Resolution limit in community detection, Proc. Natl. Acad. Sci. 104 (1) (2007) 36–41.

[13] P. Schumm, C. Scoglio, Bloom: a stochastic growth-based fast method of community detection in networks, J. Comput. Sci. 3 (5) (2012) 356–366.

[14] M.M.D. Khomami, A. Rezvanian, M.R. Meybodi, A new cellular learning automata-based algorithm for community detection in complex social networks, J. Comput. Sci. 24 (2018) 413–426.

[15] B. Karrer, M.E.J. Newman, Stochastic blockmodels and community structure in networks, Phys. Rev. E 83 (2011) 016107.

[16] T.P. Peixoto, Efficient Monte Carlo and Greedy heuristic for the inference of stochastic block models, Phys. Rev. E 89 (1) (2014) 012804.

[17] X. Liu, B. Yang, H. Chen, K. Musial, H. Chen, Y. Li, W. Zuo, A scalable redefined stochastic blockmodel, ACM Trans. Knowl. Discov. Data (TKDD) 15 (3) (2021) 1–28.

[18] B. Serrano, T. Vidal, Community detection in the stochastic block model by mixed integer programming, 2021, arXiv preprint arXiv:2101.12336.

[19] M. Rosvall, C.T. Bergstrom, An information-theoretic framework for resolving community structure in complex networks, Proc. Natl. Acad. Sci. 104 (18) (2007) 7327–7331, http://dx.doi.org/10.1073/pnas.0611034104.

[20] M. Rosvall, C.T. Bergstrom, Maps of random walks on complex networks reveal community structure, Proc. Natl. Acad. Sci. 105 (4) (2008) 1118–1123, http://dx.doi.org/10.1073/pnas.0706851105.

[21] V.A. Traag, R. Aldecoa, J.C. Delvenne, Detecting communities using asymptotical surprise, Phys. Rev. E 92 (2) (2015) 022816.

[22] R. Aldecoa, I. Marín, Deciphering network community structure by surprise, PLoS One 6 (9) (2011) 1–8, http://dx.doi.org/10.1371/journal.pone.0024195.

[23] I. Hamid, Y. Wu, Q. Nawaz, R. Zhao, A fast heuristic detection algorithm for visualizing structure of large community, J. Comput. Sci. 25 (2018) 280–288.

[24] E. Marchese, G. Caldarelli, T. Squartini, Detecting mesoscale structures by surprise, Commun. Phys. 5 (1) (2022) 1–16.

[25] S. Sobolevsky, R. Campari, A. Belyi, C. Ratti, General optimization technique for high-quality community detection in complex networks, Phys. Rev. E 90 (1) (2014) 012811.

[26] B.H. Good, Y.A. De Montjoye, A. Clauset, Performance of modularity maximization in practical contexts, Phys. Rev. E 81 (4) (2010) 046106.

[27] S. Sobolevsky, A. Belyi, Graph neural network inspired algorithm for unsupervised network community detection, Appl. Netw. Sci. 7 (1) (2022) 1–19.

[28] S. Aref, H. Chheda, M. Mostajabdaveh, The Bayan algorithm: Detecting communities in networks through exact and approximate optimization of modularity, 2022, arXiv preprint arXiv:2209.04562.

[29] A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (6) (2004) 066111.

[30] V.D. Blondel, J.L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, J. Stat. Mech. Theory Exp. 2008 (10) (2008) P10008, http://dx.doi.org/10.1088/1742-5468/2008/10/P10008.

[31] J. Reichardt, S. Bornholdt, Statistical mechanics of community detection, Phys. Rev. E 74 (1) (2006) 016110, http://dx.doi.org/10.1103/PhysRevE.74.016110.

[32] E.A. Leicht, M.E.J. Newman, Community structure in directed networks, Phys. Rev. Lett. 100 (11) (2008) 118703, http://dx.doi.org/10.1103/PhysRevLett.100.118703.

[33] P. Zhang, C. Moore, Scalable detection of statistically significant communities and hierarchies, using message passing for modularity, Proc. Natl. Acad. Sci. 111 (51) (2014) 18144–18149.

[34] T. Bonald, B. Charpentier, A. Galland, A. Hollocou, Hierarchical graph clustering using node pair sampling, in: MLG 2018-14th International Workshop on Mining and Learning with Graphs. London, UK, 2018.

[35] V.A. Traag, L. Waltman, N.J. van Eck, From Louvain to Leiden: guaranteeing well-connected communities, Sci. Rep. 9 (1) (2019) http://dx.doi.org/10.1038/s41598-019-41695-z.

[36] P.Z. Li, L. Huang, C.D. Wang, J.H. Lai, EdMot: An edge enhancement approach for motif-aware community detection, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 479–487.

[37] B.W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, Bell Syst. Tech. J. 49 (2) (1970) 291–307.

[38] T.N. Dinh, M.T. Thai, Toward optimal community detection: From trees to general weighted networks, Internet Math. 11 (3) (2015) 181–200.

[39] G. Rossetti, L. Milli, R. Cazabet, CDlib: a Python library to extract, compare and evaluate communities from complex networks, Appl. Netw. Sci. 4 (1) (2019) 1–26.

[40] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, D. Wagner, On modularity clustering, IEEE Trans. Knowl. Data Eng. 20 (2) (2007) 172–188.

[41] G. Agarwal, D. Kempe, Modularity-maximizing graph communities via mathematical programming, Eur. Phys. J. B 66 (3) (2008) 409–418, http://dx.doi.org/10.1140/epjb/e2008-00425-1.

[42] Gurobi Optimizer Reference Manual, Gurobi Optimization Inc., 2023, url: https://gurobi.com/documentation/10.0/refman/index.html date accessed 16 Feb 2023.

[43] M. Jerdee, A. Kirkley, M.E.J. Newman, Normalized mutual information is a biased measure for classification and community detection, 2023, arXiv preprint arXiv:2307.01282.

[44] N.X. Vinh, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance, J. Mach. Learn. Res. 11 (95) (2010) 2837–2854, http://jmlr.org/papers/v11/vinh10a.html.

[45] M.E. Newman, G.T. Cantwell, J.G. Young, Improved mutual information measure for clustering, classification, and community detection, Phys. Rev. E 101 (4) (2020) 042304.

[46] Z. Roozbahani, J. Rezaeenour, A. Katanforoush, Community detection in multi-relational directional networks, J. Comput. Sci. 67 (2023) 101962.

[47] D.K. Singh, S. Nandi, T. Chakraborty, P. Choudhury, Disintegrating constant communities in complex networks, J. Comput. Sci. 61 (2022) 101634.

[48] M. Sattari, K. Zamanifar, A cascade information diffusion based label propagation algorithm for community detection in dynamic social networks, J. Comput. Sci. 25 (2018) 122–133.

[49] A.J. Gates, I.B. Wood, W.P. Hetrick, Y.Y. Ahn, Element-centric clustering comparison unifies overlaps and hierarchy, Sci. Rep. 9 (1) (2019) 8574.

[50] S. Aref, M. Mostajabdaveh, H. Chheda, Heuristic modularity maximization algorithms for community detection rarely return an optimal partition or anything similar, in: J. Mikyška, C. de Mulatier, M. Paszynski, V.V. Krzhizhanovskaya, J.J. Dongarra, P.M. Sloot (Eds.), Computational Science – ICCS 2023, Springer Nature, Switzerland, Cham, 2023, pp. 612–626, http://dx.doi.org/10.1007/978-3-031-36027-5_48.

[51] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (4) (2008) 046110, http://dx.doi.org/10.1103/PhysRevE.78.046110.

[52] B. Kamiński, P. Prałat, F. Théberge, Artificial benchmark for community detection (ABCD)—Fast random graph model with community structure, Netw. Sci. 9 (2) (2021) 153–178, http://dx.doi.org/10.1017/nws.2020.45.

[53] B.F. Maier, D. Brockmann, Cover time for random walks on arbitrary complex networks, Phys. Rev. E 96 (4) (2017) 042307.

[54] T.N. Dinh, X. Li, M.T. Thai, Network clustering via maximizing modularity: Approximation algorithms and theoretical limits, in: 2015 IEEE International Conference on Data Mining, 2015, pp. 101–110, http://dx.doi.org/10.1109/ICDM.2015.139, iSSN: 1550-4786.

[55] A. Lancichinetti, S. Fortunato, Limits of modularity maximization in community detection, Phys. Rev. E 84 (6) (2011) 066122, http://dx.doi.org/10.1103/PhysRevE.84.066122.

[56] T. Chen, P. Singh, K.E. Bassler, Network community detection using modularity density measures, J. Stat. Mech. Theory Exp. 2018 (5) (2018) 053406, http://dx.doi.org/10.1088/1742-5468/aabfc8.

[57] S. Chen, Z.Z. Wang, L. Tang, Y.N. Tang, Y.Y. Gao, H.J. Li, J. Xiang, Y. Zhang, Global vs local modularity for network community detection, PLoS One 13 (10) (2018) 1–21, http://dx.doi.org/10.1371/journal.pone.0205284.

[58] X. Zhao, J. Liang, J. Wang, A community detection algorithm based on graph compression for large-scale social networks, Inform. Sci. 551 (2021) 358–372.

[59] M.A. Riolo, M.E.J. Newman, Consistency of community structure in complex networks, Phys. Rev. E 101 (2020) 052306, http://dx.doi.org/10.1103/PhysRevE.101.052306.

[60] P. Miasnikof, A.Y. Shestopaloff, A.J. Bonner, Y. Lawryshyn, P.M. Pardalos, A density-based statistical analysis of graph clustering algorithm performance, J. Complex Netw. 8 (3) (2020) 1–33.

[61] S. Cafieri, A. Costa, P. Hansen, Reformulation of a model for hierarchical divisive graph modularity maximization, Ann. Oper. Res. 222 (2014) 213–226.

[62] Y. Kawase, T. Matsui, A. Miyauchi, Additive approximation algorithms for modularity maximization, J. Comput. System Sci. 117 (2021) 182–201, http://dx.doi.org/10.1016/j.jcss.2020.11.005.

[63] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, S. Perron, L. Liberti, Column generation algorithms for exact modularity maximization in networks, Phys. Rev. E 82 (4) (2010) 046112, http://dx.doi.org/10.1103/PhysRevE.82.046112.

[64] M.J. Brusco, D. Steinley, A.L. Watts, On maximization of the modularity index in network psychometrics, Behav. Res. Methods 55 (7) (2023) 3549–3565, http://dx.doi.org/10.3758/s13428-022-01975-5.

[65] S. Aref, M. Mostajabdaveh, Dataset of synthetic modular graphs from LFR and ABCD benchmark models for community detection, 2023, http://dx.doi.org/10.6084/m9.figshare.24257293, FigShare.