Community Detection via Maximization of Modularity and Its Variants

Mingming Chen, Konstantin Kuzmin, Student Member, IEEE, and Boleslaw K. Szymanski, Fellow, IEEE

Abstract—In this paper, we first discuss the definition of modularity \(Q\) used as a metric for community quality and then we review the modularity maximization approaches which were used for community detection in the last decade. Then, we discuss two opposite yet coexisting problems of modularity optimization: in some cases, it tends to favor small communities over large ones while in others, large communities over small ones (so called the resolution limit problem). Next, we overview several community quality metrics proposed to solve the resolution limit problem and discuss Modularity Density \(Q_{ds}\) which simultaneously avoids the two problems of modularity. Finally, we introduce two novel fine-tuned community detection algorithms that iteratively attempt to improve the community quality measurements by splitting and merging the given network community structure. The first of them, referred to as Fine-tuned \(Q\), is based on modularity \(Q\) while the second one is based on Modularity Density \(Q_{ds}\) and denoted as Fine-tuned \(Q_{ds}\). Then, we compare the greedy algorithm of modularity maximization (denoted as Greedy \(Q\)), Fine-tuned \(Q\), and Fine-tuned \(Q_{ds}\) on four real networks, and also on the classical clique network and the LFR benchmark networks, each of which is instantiated by a wide range of parameters. The results indicate that Fine-tuned \(Q_{ds}\) is the most effective among the three algorithms discussed. Moreover, we show that Fine-tuned \(Q_{ds}\) can be applied to the communities detected by other algorithms to significantly improve their results.

Index Terms—Community Detection, Modularity, Maximization, Fine-tuned.

1 INTRODUCTION

Many networks, including Internet, citation networks, transportation networks, email networks, and social and biochemical networks, display community structure which identifies groups of nodes within which connections are denser than between them [1]. Detecting and characterizing such community structure, which is known as community detection, is one of the fundamental issues in the study of network systems. Community detection has been shown to reveal latent yet meaningful structure in networks such as groups in online and contact-based social networks, functional modules in protein-protein interaction networks, groups of customers with similar interests in online retailer user networks, groups of scientists in interdisciplinary collaboration networks, etc. [2].

In the last decade, the most popular community detection methods have been to maximize the quality metric known as modularity [1], [3]–[5] over all possible partitions of a network. Such modularity optimization algorithms include greedy algorithms [6]–[9], spectral methods [3], [10]–[15], extremal optimization [16], simulated annealing [17]–[20], sampling technique [21], and mathematical programming [22]. Modularity measures the difference between the actual fraction of edges within the community and such fraction expected in a randomized graph with the same number of nodes and the same degree sequence. It is widely used as a measurement of strength of the community structures detected by the community detection algorithms. However, modularity maximization has two opposite yet coexisting problems. In some cases, it tends to split large communities into two or more small communities [23], [24]. In other cases, it tends to form large communities by merging communities that are smaller than a certain threshold which depends on the total number of edges in the network and on the degree of inter-connectivity between the communities. The latter problem is also known as the resolution limit problem [23]–[25].

To solve these two issues of modularity, several community quality metrics were introduced, including Modularity Density \(Q_{ds}\) [23], [24] which simultaneously avoids both of them. We then propose two novel fine-tuned community detection algorithms that repeatedly attempt to improve the quality measurements by splitting and merging the given community structure. We denote the corresponding algorithm based on modularity \(Q\) as Fine-tuned \(Q\) while the one based on Modularity Density \(Q_{ds}\) is referred to as Fine-tuned \(Q_{ds}\). Finally, we evaluate the greedy algorithm of modularity maximization (denoted as Greedy \(Q\)), Fine-tuned \(Q\), and Fine-tuned \(Q_{ds}\) by using seven community quality metrics based on ground truth communities. These evaluations are conducted on four real networks, and also on the classical clique network and the LFR benchmark networks, each of which is instantiated by a wide range of parameters. The results indicate that Fine-tuned \(Q_{ds}\) is the most effective method and can also dramatically improve the community detection results of other algorithms. Further, all seven quality measurements based on ground truth communities are consistent with \(Q_{ds}\), but not consistent with \(Q\), which implies the superiority of Modularity Density over the original modularity.
2 REVIEW OF MODULARITY RELATED LITERATURE

In this section, we first review the definition of modularity and the corresponding optimization approaches. Then, we discuss the two opposite yet coexisting problems of modularity maximization. Finally, we overview several community quality measurements proposed to solve the resolution limit problem and then discuss Modularity Density ($Q_{ds}$) [23], [24] which simultaneously avoids these two problems.

2.1 Definition of Modularity

Comparing results of different network partitioning algorithms can be challenging, especially when network structure is not known beforehand. A concept of modularity defined in [1] provides a measure of the quality of a particular partitioning of a network. Modularity ($Q$) quantifies the community strength by comparing the fraction of edges within the community with such fraction when random connections between the nodes are made. The justification is that a community should have more links between themselves than a random gathering of people. Thus, the $Q$ value close to 0 means that the fraction of edges inside communities is no better than the random case, and the value of 1 means that a network community structure has the highest possible strength.

Formally, modularity ($Q$) can be defined as [1]:

$$Q = \sum_{c_i \in C} \left[ \frac{|E_{in}^{c_i}|}{|E|} - \left( \frac{2|E_{in}^{c_i}| + |E_{out}^{c_i}|}{2|E|} \right)^2 \right],$$  

(1)

where $C$ is the set of all the communities, $c_i$ is a specific community in $C$, $|E_{in}^{c_i}|$ is the number of edges between nodes within community $c_i$, $|E_{out}^{c_i}|$ is the number of edges from the nodes in community $c_i$ to the nodes outside $c_i$, and $|E|$ is the total number of edges in the network.

Modularity can also be expressed in the following form [3]:

$$Q = \frac{1}{2|E|} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2|E|} \right) \delta_{c_i,c_j},$$  

(2)

where $k_i$ is the degree of node $i$, $A_{ij}$ is an element of the adjacency matrix, $\delta_{c_i,c_j}$ is the Kronecker delta symbol, and $c_i$ is the label of the community to which node $i$ is assigned.

Since larger $Q$ means a stronger community structure, several algorithms which we will discuss in the next section, are based on modularity optimization.

The modularity measure defined above is suitable only for undirected and unweighted networks. However, this definition can be naturally extended to apply to directed networks as well as to weighted networks. Weighted and directed networks contain more information than undirected and unweighted ones and are therefore often viewed as more valuable but also as more difficult to analyze than their simpler counterparts.

The revised definition of modularity that works for directed networks is as follows [4]:

$$Q = \frac{1}{|E|} \sum_{ij} \left( A_{ij} - \frac{k_i^{in} k_j^{out}}{|E|} \right) \delta_{c_i,c_j},$$  

(3)

where $k_i^{in}$ and $k_j^{out}$ are the in- and out- degrees.

Although many networks can be regarded as binary, i.e. as either having an edge between a pair of nodes or not having it, there are many other networks for which it would be natural to treat edges as having a certain degree of strength or weight.

The same general techniques that have been developed for unweighted networks are applied to its weighted counterparts in [5] by mapping weighted networks onto multigraphs. For non-negative integer weights, an edge with weight $w$ in a weighted graph corresponds to $w$ parallel edges in a corresponding multigraph. Although negative weights can arise in some applications they are rarely useful in social networks, so for the sake of brevity we will not discuss them here. It turns out that an adjacency matrix of a weighted graph is equivalent to that of a multigraph with unweighted edges. Since the structure of adjacency matrix is independent of the edge weights, it is possible to adjust all the methods developed for unweighted networks to the weighted ones.

It is necessary to point out that the notion of degree of a node should also be extended for the weighted graphs. In this case degree of a node is defined as the sum of weights of all edges incident to this node.

It is shown in [5] that the same definitions of modularity that were given above hold for the weighted networks as well if we treat $A_{ij}$ as the value that represents weight of the connection and set $|E| = \frac{1}{2} \sum_{ij} A_{ij}$.

2.2 Modularity Optimization Approaches

In the literature, a high value of modularity ($Q$) indicates a good community structure and the partition corresponding to the maximum value of modularity on a given graph is supposed to have the highest quality, or at least a very good one. Therefore, it is natural to discover communities by maximizing modularity over all possible partitions of a network. However, it is computationally prohibitively expensive to exhaustively search all such partitions for the optimal value of modularity since modularity optimization is known to be NP-hard [26]. However, many heuristic methods were introduced to find high-modularity partitions in a reasonable time. Those approaches include greedy algorithms [6]–[9], spectral methods [3], [10]–[15], extremal optimization [16], simulated annealing [17]–[20], sampling technique [21], and mathematical programming [22]. In this section, we will review those modularity optimization heuristics.
2.2.1 Greedy Algorithms

The first greedy algorithm was proposed by Newman [6]. It is an agglomerative hierarchical clustering method. Initially, every node belongs to its own community, creating altogether $|V|$ communities. Then, at each step, the algorithm repeatedly merges pairs of communities together and chooses the merger for which the resulting modularity is the largest. The change in $Q$ upon joining two communities $c_i$ and $c_j$ is

$$
\Delta Q_{c_i, c_j} = 2 \left( \frac{|E_{c_i, c_j}|}{|E|} - \frac{E_{c_i} E_{c_j}}{4|E|^2} \right), \tag{4}
$$

where $|E_{c_i, c_j}|$ is the number of edges from community $c_i$ to community $c_j$ and $|E_{c_i}| = 2|E_{c_i}^{in}| + |E_{c_i}^{out}|$ is the total degrees of nodes in community $c_i$. $\Delta Q_{c_i, c_j}$ can be calculated in constant time. The algorithm stops when all the nodes in the network are in a single community after $(|V| - 1)$ steps of merging. Then, there are totally $|V|$ partitions, the first one defined by the initial step and each subsequent one resulting from each of the subsequent $(|V| - 1)$ merging steps. The partition with the largest value of modularity, approximating the modularity maximum best, is the result of the algorithm.

At each merging step, the algorithm needs to compute the change $\Delta Q_{c_i, c_j}$ of modularity resulting from joining any two currently existing communities $c_i$ and $c_j$ in order to choose the best merger. Since merging two disconnected communities will not increase the value of modularity, the algorithm checks only the merging of connected pairs of communities and the number of such pairs is at most $|E|$ limiting the complexity of this part to $O(|E|)$. However, the rows and columns of adjacent matrix corresponding to the two merged communities must be updated, which takes $O(|V|)$. Since there are $(|V| - 1)$ iterations, the final complexity of the algorithm is $O((|E| + |V|)|V|)$, or $O(|V|^{2.5})$ for sparse networks.

Although Newman’s algorithm [6] is much faster than the algorithm of Newman and Girvan [1] whose complexity is $O(|V|^{2.5})$, Clauset et al. [7] pointed out that the update of the adjacent matrix at each step contains a large number of unnecessary operations when the network is sparse and therefore its matrix has a lot of zero entries. They introduced data structures for sparse matrices to perform the updating operation more efficiently. In their algorithm, instead of maintaining the adjacent matrix and computing $\Delta Q_{c_i, c_j}$, they maintained and updated the matrix with entries being $\Delta Q_{e_i, e_j}$ for the pairs of connected communities $c_i$ and $c_j$. The authors introduced three data structures to represent sparse matrices efficiently: (1) each row of the matrix is stored as a balanced binary tree in order to search and insert elements in $O(|log|V||)$ time and also as a max-heap so as to locate the largest element of each row in constant time; (2) another max-heap stores the largest element of each row of the matrix so as to locate the largest $\Delta Q_{e_i, e_j}$ in constant time; (3) a vector is used to save $|E_{c_i}|$ for each community $c_i$. Then, in each step, the largest $\Delta Q_{c_i, c_j}$ can be found in constant time and the update of the adjacent matrix after merging two communities $c_i$ and $c_j$ takes $O((k_{c_i} + k_{c_j})|log|V|)$, where $k_{c_i}$ and $k_{c_j}$ are the numbers of neighboring communities of communities $c_i$ and $c_j$, respectively. Thus, the total running time is at most $O(|log|V|)$ times the sum of the degrees of nodes in the communities along the dendrogram created by merging steps. This sum is in the worst case the depth of the dendrogram times the sum of the degrees of nodes in the network. Suppose the dendrogram has depth $d$, then the running time is $O(d|E| |log|V|)$, or $O(|V| |log|^2|V|)$ when the network is sparse and the dendrogram is almost balanced ($d \sim |log|V|$).

However, Wakita and Tsurumi [8] observed that the greedy algorithm proposed by Clauset et al. is not scalable to networks with sizes larger than 500,000 nodes. They found that the computational inefficiency arises from merging communities in an unbalanced manner, which yields very unbalanced dendrograms. In such cases, the relation $d \sim |log|V|$ does not hold any more, causing the algorithm to run at its worst-case complexity. To balance the merging of communities, the authors introduced three types of consolidation ratios to measure the balance of the community pairs and used it with modularity to perform the joining process of communities without bias. This modification enables the algorithm to scale to networks with sizes up to 10,000,000. It also approximates the modularity maximum better than the original algorithm.

Another type of greedy modularity optimization algorithm different from those above was proposed by Blondel et al., and it is usually referred to as Louvain [9]. It is divided into two phases that are repeated iteratively. Initially, every node belongs to the community of itself, so there are $|V|$ communities. In this first phase, every node, in a certain order, is considered for merging into its neighboring communities and the merger with the largest positive gain is selected. If all possible gains associated with the merging of this node are negative, then it stays in its original community. This merging procedure repeats iteratively and stops when no increase of $Q$ can be achieved.

After the first phase, Louvain reaches a local maximum of $Q$. Then, the second phase of Louvain builds a community network based on the communities discovered in the first phase. The nodes in the new network are the communities from the first phase and there is an edge between two new nodes if there are edges between nodes in the corresponding two communities. The weights of those edges are the sum of the weights of the edges between nodes in the corresponding two communities. The edges between nodes of the same community of the first phase result in a self-loop for this community node in the new network. After the community network is generated, the algorithm applies the first phase again on this new network. The two phases repeat iteratively and stop when there is no more change and consequently a maximum modularity is obtained. The number of iterations of this algorithm is usually very small and
most of computational time is spent in the first iteration. Thus, the complexity of the algorithm grows like \( O(|E|) \). Consequently, it is scalable to large networks with the number of nodes up to a billion. However, the results of Louvain are impacted by the order in which the nodes in the first phase are considered for merging [27].

2.2.2 Spectral Methods

There are two categories of spectral algorithms for maximizing modularity: one is based on the modularity matrix [3], [10], [11]; the other is based on the Laplacian matrix of a network [12]–[14].

A. Modularity optimization using the eigenvalues and eigenvectors of the modularity matrix [3], [10], [11].

Modularity \( (Q) \) can be expressed as [3]

\[
Q = \frac{1}{4|E|} \sum_{i,j} \left( A_{ij} - \frac{k_i k_j}{2|E|} \right) (s_i s_j + 1)
\]

\[
= \frac{1}{4|E|} \sum_{i,j} \left( A_{ij} - \frac{k_i k_j}{2|E|} \right) s_i s_j = \frac{1}{4|E|} s^T B s,
\]

(5)

where \( A_{ij} \) are the elements of adjacent matrix \( A \) and \( s \) is the column vector representing any division of the network into two groups. Its elements are defined as \( s_i = +1 \) if node \( i \) belongs to the first group and \( s_i = -1 \) if it belongs to the second group. \( B \) is the modularity matrix with elements

\[
B_{ij} = A_{ij} - \frac{k_i k_j}{2|E|}.
\]

(6)

Representing \( s \) as a linear combination of the normalized eigenvectors \( u_i \) of \( B \): \( s = \sum_{i=1}^{[V]} a_i u_i \) with \( a_i = u_i^T \cdot s \), and then plugging the result into Equation (5) yield

\[
Q = \frac{1}{4|E|} \sum_i a_i u_i^T B \sum_j a_j u_j = \frac{1}{4|E|} \sum_i a_i^2 \beta_i,
\]

(7)

where \( \beta_i \) is the eigenvalue of \( B \) corresponding to eigenvector \( u_i \). To maximize \( Q \) above, Newman [3] proposed a spectral approach to choose \( s \) proportional to the leading eigenvector \( u_1 \) corresponding to the largest (most positive) eigenvalue \( \beta_1 \). The choice assumes that the eigenvalues are labeled in decreasing order \( \beta_1 \geq \beta_2 \geq \ldots \geq \beta_{|V|} \). Nodes are then divided into two communities according to the signs of the elements in \( s \) with nodes corresponding to positive elements in \( s \) assigned to one group and all remaining nodes to another. Since the row and column sums of \( B \) are zero, it always has an eigenvector \((1, 1, 1, \ldots)\) with eigenvalue zero. Therefore, if it has no positive eigenvalue, then the leading eigenvector is \((1, 1, 1, \ldots)\), which means that the network is indivisible. Moreover, Newman [3] proposed to divide network into more than two communities by repeatedly dividing each of the communities obtained so far into two until the additional contribution \( \Delta Q \) to the modularity made by the subdivision of a community \( c \)

\[
\Delta Q = \frac{1}{2|E|} \left[ \frac{1}{2} \sum_{i,j \in c} B_{ij} (s_i s_j + 1) - \sum_{i,j \in c} B_{ij} \right]
\]

(8)

is equal to or less than 0. \( B^{(c)} \) in the formula above is the generalized modularity matrix. Its elements, indexed by the labels \( i \) and \( j \) of nodes within community \( c \), are

\[
B^{(c)}_{ij} = B_{ij} - \delta_{ij} \sum_{k \in c} B_{ik}.
\]

(9)

Then, the same spectral method can be applied to \( B^{(c)} \) to maximize \( \Delta Q \). The recursive subdivision process stops when \( \Delta Q \leq 0 \), which means that there is no positive eigenvalue of the matrix \( B^{(c)} \). The overall complexity of this algorithm is \( O((|E| + |V|)|V|) \).

However, the spectral algorithm described above has two drawbacks. First, it divides a network into more than two communities by repeated division instead of getting all the communities directly in a single step. Second, it only uses the leading eigenvector of the modularity matrix and ignores all the others, losing all the useful information contained in those eigenvectors. Newman later proposed to divide a network into a set of communities \( C \) with \(|C| \geq 2 \) directly using multiple leading eigenvectors [10]. Let \( S = (s_c) \) be an \(|V| \times |C| \) “community-assignment” matrix with one column for each community \( c \) defined as

\[
S_{i,c} = \begin{cases} 1 & \text{if node } i \text{ belongs to community } c, \\ 0 & \text{otherwise.} \end{cases}
\]

(10)

then the modularity \( (Q) \) for this direct division of the network is given by

\[
Q = \frac{1}{2|E|} \sum_{i,j=1}^{[V]} \sum_{c \in C} B_{ij} S_{i,c} S_{j,c} = \frac{1}{2|E|} \text{Tr}(S^T B S),
\]

(11)

where \( \text{Tr}(S^T B S) \) is the trace of matrix \( S^T B S \). Defining \( B = U \Sigma U^T \), where \( U = (u_1, u_2, \ldots) \) is the matrix of eigenvectors of \( B \) and \( \Sigma \) is the diagonal matrix of eigenvalues \( \Sigma_{ii} = \beta_i \), yields

\[
Q = \frac{1}{2|E|} \sum_{i=1}^{[V]} \sum_{c \in C} \beta_i (u_i^T s_c)^2.
\]

(12)

Then, obtaining \(|C| \) communities is equivalent to selecting \(|C| - 1 \) independent, mutually orthogonal columns \( s_c \). Moreover, \( Q \) would be maximized by choosing the columns \( s_c \) proportional to the leading eigenvectors of \( B \). However, only the eigenvectors corresponding to the positive eigenvalues will contribute positively to the modularity. Thus, the number of positive eigenvalues, plus 1, is the upper bound of \(|C| \). More general modularity maximization is to keep the leading \( p \) (1 \leq p \leq |V|)
eigenvectors. $Q$ can be rewritten as

$$Q = \frac{1}{2|E|} \left( |V| \alpha + \text{Tr}(S^T U (\Sigma - \alpha I) U^T S) \right)$$

$$= \frac{1}{2|E|} \left( |V| \alpha + \sum_{j=1}^{|V|} (\beta_j - \alpha) \left( \sum_{i=1}^{|V|} U_{ij} S_{ic} \right)^2 \right),$$

where $\alpha (\alpha \leq \beta_p)$ is a constant related to the approximation for $Q$ obtained by only adopting the first $p$ leading eigenvectors. By selecting $|V|$ node vectors $r_i$ of dimension $p$ whose $j$th component is

$$[r_i]_j = \sqrt{\beta_j - \alpha} U_{ij},$$

modularity can be approximated as

$$Q \simeq \tilde{Q} = \frac{1}{2|E|} \left( |V| \alpha + \sum_{c \in C} |R_c|^2 \right),$$

where $R_c, c \in C$, are the community vectors

$$R_c = \sum_{i \in c} r_i.$$

Thus, the community detection problem is equivalent to choosing such a division of nodes into $|C|$ groups that maximizes the magnitudes of the community vectors $R_c$ while requiring that $R_c \cdot r_i > 0$ if node $i$ is assigned to community $c$. Problems of this type are called vector partitioning problems.

Although [10] explored using multiple leading eigenvectors of the modularity matrix, it did not pursue it in detail beyond a two-eigenvector approach for bipartitioning. Richardson et al. [11] provided a generalization of these recursive bipartitioning methods by considering the best two-way or three-way division at each recursive step to more thoroughly explore the promising partitions. To reduce the number of partitions considered for the eigenvector-pair tripartitioning, the authors adopted a divide-and-conquer method and as a result yielded an efficient approach whose computational complexity is competitive with the two-eigenvector bipartitioning method.

**B. Modularity optimization using the eigenvalues and eigenvectors of the Laplacian matrix** [12]–[14].

Given a partition $C$ (a set of communities) and the corresponding “community-assignment” matrix $S = (s_{ec})$, White and Smyth [12] rewrote modularity ($Q$) as follows:

$$Q \propto \text{Tr}(S^T (W - \bar{D}) S) = -\text{Tr}(S^T L_Q S),$$

where $W = 2|E| A$ and the elements of $\bar{D}$ are $\bar{D}_{ij} = k_i k_j$. The matrix $L_Q = \bar{D} - W$ is called the “Q-Laplacian”. Finding the “community-assignment” matrix $S$ that maximizes $Q$ above is NP-complete, but a good approximation can be obtained by relaxing the discreteness constraints of the elements of $S$ and allowing them to assume real values. Then, $Q$ becomes a continuous function of $S$ and its extremes can be found by equating its first derivative with respect to $S$ to zero. This leads to the eigenvalue equation:

$$L_Q S = \Lambda S,$$

where $\Lambda$ is the diagonal matrix of Lagrangian multipliers. Thus, the modularity optimization problem is transformed into the standard spectral graph partitioning problem. When the network is not too small, $L_Q$ can be approximated well, up to constant factors, by the transition matrix $\tilde{W} = D^{-1} A$ obtained by normalizing $A$ so that all rows sum to one. $D$ here is the diagonal degree matrix of $A$. It can be shown that the eigenvalues and eigenvectors of $\tilde{W}$ are precisely $1 - \lambda$ and $\mu$, where $\lambda$ and $\mu$ are the solutions to the generalized eigenvalue problem $L_{\mu} = \lambda D_{\mu}$ where $L = D - A$ is the Laplacian matrix. Thus, the underlying spectral algorithm here is equivalent to the standard spectral graph partitioning problem which uses the eigenvalues and eigenvectors of the Laplacian matrix.

Based on the above analysis, White and Smyth proposed two clustering algorithms, named “Algorithm Spectral-1” and “Algorithm Spectral-2”, to search for a partition $C$ with size up to $K$ predefined by an input parameter. Both algorithms take the eigenvector matrix $U_K = (u_1, u_2, ..., u_{K-1})$ with the leading $K - 1$ eigenvectors (excluding the trivial all-ones eigenvector) of the transition matrix $\tilde{W}$ as input. Those $K - 1$ eigenvectors can be efficiently computed with the Implicitly Restarted Lanczos Method (IRLM) [28]. “Algorithm Spectral-1” uses the first $k - 1$ ($2 \leq k \leq K$) columns of $U_K$, denoted as $U_{k-1}$, and clusters the row vectors of $U_{k-1}$ using $k$-means to find a $k$-way partition, denoted as $C_k$. Then, the $C_{k^*}$ with size $k^*$ that achieves the largest value of $Q$ is the final community structure.

“Algorithm Spectral-2” starts with a single community ($k = 1$) and recursively splits each community $c$ into two smaller ones if the subdivision produces a higher value of $Q$. The split is done by running $k$-means with two clusters on the matrix $U_{k,c}$ formed from $U_k$ by keeping only rows corresponding to nodes in $c$. The recursive procedure stops when no more splits are possible or when $k = K$ communities have been found and then the final community structure with the highest value of $Q$ is the detection result.

However, the two algorithms described above, especially “Algorithm Spectral-1”, scale poorly to large networks because of running $k$-means partitioning up to $K$ times. Both approaches have a worst-case complexity $O(K^2 |V| + K |E|)$. In order to speed up the calculation while retaining effectiveness in approximating the maximum of $Q$, Ruan and Zhang [13] proposed the $K$cut algorithm which recursively partitions the network to optimize $Q$. At each recursive step, $K$cut adopts a $k$-way partition ($k = 2, 3, ..., l$) to the subnetwork induced by the nodes and edges in each community using “Algorithm Spectral-1” of White and Smyth [12]. Then, it selects the $k$ that achieves the highest $Q$. Empirically, $K$cut with
l as small as 3 or 4 can significantly improve $Q$ over the standard bi-partitioning method and it also reduces the computational cost to $O((|V| + |E|)|\log|C|)$ for a final partition with $|C|$ communities.

Ruan and Zhang later [14] proposed QCUT algorithm that combines $Kcut$ and local search to optimize $Q$. The QCUT algorithm consists of two alternating stages: partitioning and refinement. In the partitioning stage, $Kcut$ is used to recursively partition the network until $Q$ cannot be further improved. In the refinement stage, a local search strategy repeatedly considers two operations. The first one is migration that moves a node from its current community to another one and the second one is the merge of two communities into one. Both are applied to improve $Q$ as much as possible. The partitioning stage and refinement stage are alternating until $Q$ cannot be increased further. In order to solve the resolution limit problem of modularity, the authors proposed HQCUT which recursively applies QCUT to divide the subnetwork, generated with the nodes and edges in each community, into subcommunities. Further, to avoid overpartitioning, they use a statistical test to determine whether a community indeed has intrinsic subcommunities.

C. Equivalence of two categories of spectral algorithms for maximizing modularity [15].

Newman [15] showed that with hyperellipsoid relaxation, the spectral modularity maximization method using the eigenvalues and eigenvectors of the modularity matrix can be formulated as the spectral algorithm that relies on the eigenvalues and eigenvectors of Laplacian matrix. This formulation indicates that the above two kinds of modularity optimization approaches are equivalent. Starting with Equation (5) for the division of a network into two groups, first the discreteness of $s_i$ is relaxed onto a hyperellipsoid with the constraint

$$\sum_i k_i s_i^2 = 2|E|. \quad (19)$$

Then, the relaxed modularity maximization problem can be easily solved by setting the first derivative of Equation (5) with respect to $s_i$ to zero. This leads to

$$\sum_j B_{ij} s_j = \lambda k_i s_i, \quad (20)$$

or in matrix notion

$$Bs = \lambda Ds, \quad (21)$$

where $\lambda$ is the eigenvalue. Plugging Equation (20) into Equation (5) yields

$$Q = \frac{1}{4|E|} \sum_{ij} B_{ij} s_i s_j = \frac{\lambda}{4|E|} \sum_i k_i s_i^2 = \frac{\lambda}{2}. \quad (22)$$

Therefore, to achieve the highest value of $Q$, one should chose $\lambda$ to be the largest (most positive) eigenvalue of Equation (21). Using Equation (6), Equation (20) can be rewritten as

$$\sum_j A_{ij} s_j = k_i (\lambda s_i + \frac{1}{2|E|} \sum_j k_j s_j), \quad (23)$$

or in matrix notion as

$$As = D \left( \lambda s + \frac{k^T s}{2|E|} \right), \quad (24)$$

where $k$ is the vector with element $k_i$ and $1 = (1, 1, 1, ...)$.

Then, multiplying the above equation by $1^T$ results in $\lambda k^T s = 0$. If there is a nontrivial eigenvalue $\lambda > 0$, then the above equation simplifies to

$$As = \lambda Ds. \quad (25)$$

Again, $\lambda$ should be the most positive eigenvalue. However, the eigenvector corresponding to this eigenvalue is the uniform vector $1$ which fails to satisfy $k^T s = 0$. Thus, in this case, one can do the best by choosing $\lambda$ to be the second largest eigenvalue and having $s$ proportional to the corresponding eigenvector. In fact, this eigenvector is precisely equal to the leading eigenvector of Equation (21). Then, after defining a rescaled vector $u = D^{1/2}s$ and plugging it into Equation (25), we get

$$(D^{-1/2} A D^{-1/2}) u = \lambda u. \quad (26)$$

The matrix $L = D^{-1/2} A D^{-1/2}$ is called the normalized Laplacian matrix. (The normalized Laplacian is sometimes defined as $L = I - D^{-1/2} A D^{-1/2}$, but those two differ only by a trivial transformation of their eigenvalues and eigenvectors.)

2.2.3 Extremal Optimization

Duch and Arenas [16] proposed a modularity optimization algorithm based on the Extremal Optimization (EO) [29]. EO optimizes a global variable by improving extremal local variables. Here, the global variable is modularity ($Q$). The contribution of an individual node $i$ to $Q$ of the whole network with a certain community structure is given by

$$q_i = k_{i,c} - k_i \frac{|E_i|}{2|E|}, \quad (27)$$

where $k_{i,c}$ is the number of edges that connect node $i$ to the nodes in its own community $c$. Notice that $Q = \frac{1}{2|E|} \sum_i q_i$ and $q_i$ can be normalized into the interval $[-1, 1]$ by diving it by $k_i$,

$$\lambda_i = \frac{q_i}{k_i} = \frac{k_{i,c}}{k_i} - \frac{|E_i|}{2|E|}, \quad (28)$$

where $\lambda_i$, called fitness, is the relative contribution of node $i$ to $Q$. Then, the fitness of each node is adopted as the local variable.

The algorithm starts by randomly splitting the network into two partitions of equal number of nodes, where communities are the connected components in each partition. Then, at each iteration, it moves the node with the lowest fitness from its own community to
another community. The shift changes the community structure, so the fitness of many other nodes needs to be recomputed. The process repeats until it cannot increase $Q$. After that, it generates sub-community networks by deleting the inter-community edges and proceeds recursively on each sub-community network until $Q$ cannot be improved. Although the procedure is deterministic when given the initialization, its final result in fact depends on the initialization and it is likely to get trapped in local maxima. Thus, a probabilistic selection called $\tau$-EO [29] in which nodes are ranked according to their fitness and a node of rank $r$ is selected with the probability $P(r) \propto r^{-\tau}$ is used to improve the result. The computational complexity of this algorithm is $O(|V|^2 \log^2 |V|)$.

2.2.4 Simulated Annealing

Simulated annealing (SA) [30] is a probabilistic procedure for the global optimization problem of locating a good approximation to the global optimum of a given function in a large search space. This technique was adopted in [17]–[20] to maximize modularity ($Q$). The initial point for all those approaches can be arbitrary partitioning of nodes into communities, even including $|V|$ communities in which each node belongs to its own community. At each iteration, a node $i$ and a community $c$ are chosen randomly. This community could be a currently existing community or an empty community introduced to increase the number of communities. Then, node $i$ is moved from its original community to this new community $c$, which would change $Q$ by $\Delta Q$. If $\Delta Q$ is greater than zero, this update is accepted, otherwise it is accepted with probability $e^{\beta \Delta Q}$ where $\beta$ in [17]–[19] represents the inverse of temperature $T$ and $\beta$ in [20] is the reciprocal of pseudo temperature $\tau$. In addition in [20], there is one more condition for the move of a node when $c$ is not empty, shifting node $i$ to $c$ is considered only if there are some edges between node $i$ and the nodes in $c$. To improve the performance and to avoid getting trapped in local minima, collective movements which involve moving multiple nodes at a time [19], [20], merging two communities [17]–[19], and splitting a community [17]–[19] are employed. Splits can be carried out in a number of different schemes. The best performance is achieved by treating a community as an isolated subnetwork and partitioning it into two and then performing a nested SA on these partitions [17], [18]. Those methods stop when no new update is accepted within a fixed number of iterations.

2.2.5 Sampling Techniques

Sales-Pardo et al. [21] proposed a “box-clustering” method to extract the hierarchical organization of networks. This approach consists of two steps: (1) estimating the similarity, called “node affinity”, between nodes and forming the node affinity matrix; (2) deriving hierarchical community structure from the affinity matrix. The affinity between two nodes is the probability

that they are classified into the same community in the local maxima partitions of modularity. The set of local maxima partitions, called $P_{\text{max}}$, includes those partitions for which neither the moving of a node from its original community to another, nor the merging of two communities will increase the value of modularity. The sample $P_{\text{max}}$ is found by performing the simulated annealing based modularity optimization algorithm of Guimerà and Amaral [17], [18]. More specifically, the algorithm first randomly divides the nodes into communities and then performs the hill-climbing search until a sample with local maximum of modularity is reached. Then, the affinity matrix is updated based on the obtained sample.

The sample generation procedure is repeated until the affinity matrix has converged to its asymptotic value. Empirically, the total number of samples needed is proportional to the size of the network. Before proceeding to the second step, the algorithm assesses whether the network has a significant community structure or not. It is done by computing the $z$-score of the average modularity of the partitions in $P_{\text{max}}$ with respect to the average modularity of the partitions with the local modularity maxima of the equivalent ensemble of null model networks. The equivalent null model is obtained by randomly rewiring the edges of the original network while retaining the degree sequence. Large $z$-score indicates that the network has a meaningful internal community structure. If the network indeed has a significant community structure, the algorithm advances to the second step to group nodes with large affinity close to each other. The goal is to bring the form of the affinity matrix as close as possible to block-diagonal structure by minimizing the cost function representing the average distance of matrix elements to the diagonal. Then, the communities corresponds to the “best” set of boxes obtained by least-squares fitting of the block-diagonal structure to the affinity matrix. The procedure described above can be recursively performed to subnetworks induced by communities to identify the low level structure of each community until no subnetwork is found to have significant intrinsic structure.

2.2.6 Mathematical Programming

Agarwal and Kempe [22] formulated the modularity maximization problem as a linear program and vector program which have the advantage of providing a posteriori performance guarantees. First, modularity maximization can be transformed into the integer program

\[
\text{Maximize } \frac{1}{2|E|} \sum_{ij} B_{ij} (1 - x_{ij})
\]

subject to $x_{ik} \leq x_{ij} + x_{jk}$ for all $i, j, k$

$x_{ij} \in \{0, 1\}$ for all $i, j$,

where $B$ is the modularity matrix and the objective function is linear in the variable $x_{ij}$. When $x_{ij} = 0$, $i$ and $j$ belong to the same community and $x_{ij} = 1$ indicates that they are in different communities. The restriction
Since its inception, the modularity has been used extensively as the measure of the quality of partitions produced by community detection algorithms. In fact, if we adopt modularity as a quality measure of communities, the task of discovering communities is essentially turned into the task of finding the network partitioning with an optimal value of modularity.

However as properties of the modularity were studied, it was discovered that in some cases it fails to detect small communities. There is a certain threshold [25], such that a community of the size below it will not be detected even if it is a complete subgraph connected to the rest of the graph with a single edge. This property of modularity has become known as the resolution limit.

Although the resolution limit prevents detection of small communities, the actual value of the threshold depends on the total number of edges in the network and on the degree of interconnectedness between communities. In fact, the resolution limit can reach the values comparable to the size of the entire network causing formation of a few giant communities (or even a single community) and failing to detect smaller communities within them. It makes interpreting the results of community detection very difficult because it is impossible to tell beforehand whether a community is well-formed or if it can be further split into subcommunities.

Considering modularity as a function of the total number of edges, \(|E|\), and the number of communities, \(m\), makes it possible to find the values of \(m\) and \(|E|\) which maximize this function. It turns out that setting \(m = \sqrt{|E|}\) yields the absolute maximal value of modularity. Consequently, modularity has a resolution limit of order \(\sqrt{|E|}\) which bounds the number and size of communities [25]. In fact, if for a certain community the number of edges inside it is smaller than \(\sqrt{|E|}\), such community cannot be resolved through the modularity optimization. It is also possible for modularity optimization to fail to detect communities of larger size if they have more edges in common with the rest of the network. Therefore, by finding the optimal value of the modularity we are generally not obtaining the best possible structure of communities.

The above arguments can also be applied to weighted networks. In this case, \(|E|\) is the sum of the weights of all the edges in the network, \(|E|_{\text{in}}\) is the sum of the weights of the edges between nodes within community \(c_i\), and \(|E|_{\text{out}}\) is the sum of the weights of the edges from the nodes in community \(c_i\) to the nodes outside \(c_i\).

By introducing an additional parameter, \(\epsilon\), which represents the weight of inter-community edges, Berry et al. showed in [32] that the number of communities in the optimal solution is

\[
m = \sqrt{\frac{|E|}{\epsilon}}.
\]

Correspondingly, any community for which its size

\[
|c_i| < \sqrt{\frac{|E|\epsilon}{2} - \epsilon}
\]

may not be resolved.

Introduction of \(\epsilon\) brings some interesting opportunities. If we can make \(\epsilon\) arbitrarily small, then we can expect maximum weighted modularity to produce any desired number of communities. In other words, given a proper weighting, a much better modularity resolution can be achieved than without weighting. However, in practice, finding a way to set edge weights to...
achieve small values of $\epsilon$ can be challenging. An algorithm for lowering $\epsilon$ proposed by Berry et al. requires $O(m|V| \log |V|)$ time.

### 2.4 Resolving the resolution limit problem

There have been extensive studies done on how to mitigate the consequences of the modularity resolution limit. The main approaches followed are described below.

Localized modularity measure ($LQ$) [33] is based on the observation that the resolution limit problem is caused by modularity being a global measure since it assumes that edges between any pairs of nodes are equally likely, including connectivity between the communities. However, in many networks, the majority of communities have edges to only a few other communities, i.e. exhibit a local community connectivity. Thus, a local version of the modularity measure for a directed network is defined as:

$$LQ = \sum_{c_i \in C} \left[ \frac{|E^{\text{in}}_{c_i}|}{|E^{\text{neigh}}_{c_i}|} - \left( \frac{|E^{\text{in}}_{c_i}| + |E^{\text{out}}_{c_i}|}{|E^{\text{neigh}}_{c_i}|} \right)^2 \right], \quad (33)$$

where $|E^{\text{neigh}}_{c_i}|$ is the total number of edges in the neighboring communities of $c_i$, i.e. in the communities to which all neighbors of $c_i$ belong.

Unlike traditional modularity ($Q$), the local version of modularity ($LQ$) is not bounded above by 1. The more locally connected communities a network has, the bigger its $LQ$ can grow. In a network where all communities are connected to each other, $LQ$ yields the same value as $Q$. $LQ$ considers individual communities and their neighbors, and therefore provides a measure of community quality that is not dependent on other parts of the network. The local connectivity approach can be applied not only to the nearest neighboring communities, but also to the second or higher neighbors as well.

Arenas et al. proposed a multiple resolution method [34] which is based on the idea that it might be possible to look at the detected community structure at different scales. From this perspective, the modularity resolution limit is not a problem but a feature. It allows choosing a desired resolution level to achieve the required granularity of the output community structure using the original definition of modularity.

The multiple resolution method is based on the definition of modularity given by Equation (1). The modularity resolution limit depends on the total weight $2|E|$. By varying the total weight, it is possible to control the resolution limit, effectively performing community detection at different granularity levels. Changing the sum of weights of edges adjacent to every node by some value $r$ results in rescaling topology by a factor of $r$. Since the resolution limit is proportional to $\sqrt{r}$, the growth of the resolution limit is slower than that of $r$. Consequently, it would be possible to achieve a scale at which all required communities would be visible to the modularity optimization problem.

Caution should be exercised when altering the weights of edges in the network to avoid changing its topological characteristics. To ensure this, a rescaled adjacency matrix can be defined as:

$$A_r = A + rI,$$  \quad (34)

where $A$ is the adjacency matrix and $I$ is the identity matrix. Since the original edge weights are not altered, $A_r$ preserves all common features of the network: distribution of sum of weights, weighted clustering coefficient, eigenvectors, etc. Essentially, introducing $r$ results in a self-loop of weight $r$ being added to every node in the network.

Optimizing the modularity for the rescaled topology $A_r$ is performed by using the modularity at scale $r$ as the new quality function:

$$Q_r = \sum_{c_i \in C} \left[ \frac{2|E^{\text{in}}_{c_i}| + r|c_i|}{2|E| + r|V|} - \frac{|E^{\text{in}}_{c_i}| + r|c_i|}{2|E| + r|V|} \right]^2,$$ \quad (35)

where $|c_i|$ is the number of nodes in community $c_i$ and $|E^{\text{in}}_{c_i}| = 2|E_{c_i}| + |E_{c_i}^\text{out}|$. It yields larger communities for smaller values of $r$ and smaller communities for larger values of $r$. By performing modularity optimization for different values of $r$, it is possible to analyze the community structure at different scales.

Parameter $r$ can also be thought of as representing resistance of a node to become part of a community. If $r$ is positive, we can obtain a network community structure that is more granular than what was possible to achieve with the original definition of modularity ($Q$) which corresponds to $r$ being zero. Making $r$ negative zooms out of the network and provides a view of super communities.

Further studies of the multiple resolution approach revealed that it suffers from two major issues outlined in [35]. First, when the value of the resolution parameter $r$ is low it tends to group together small communities. Second, when the resolution is high, it splits large communities. These trends are opposite for networks with a large variation of community sizes. Hence, it is impossible to select a value of the resolution parameter such that neither smaller nor larger communities are adversely affected by the resolution limit. A network can be tested for susceptibility to the resolution problem by examining its clustering coefficient, i.e. a degree to which nodes tend to form communities. If the clustering coefficient has sharp changes, it indicates that communities of substantially different scales exist in this network. The result is that when the value of $r$ is sufficiently large, bigger communities get broken up before smaller communities are found. This applies also to other multiple resolution methods and seems to be a general problem of the methods that are trying to optimize some global measure.

The hierarchical multiresolution method proposed by Granell et al. in [36] overcomes the limitations of the
multiple resolution method on networks with very different scales of communities. It achieves that by introducing a new hierarchical multiresolution scheme that works even in cases of community detection near the modularity resolution limit. The main idea underlying this method is based on performing multiple resolution community detection on essential parts of the network, thus analyzing each part independently.

The method operates iteratively by first placing all nodes in a single community. Then, it finds the minimum value of the resistance parameter $r$ which produces a community structure with the optimal value of modularity. Finally, it runs the same algorithm on each community that was found. The method terminates when no more split of communities is necessary, which usually takes just a few steps.

Another approach to leveraging the results of modularity optimization has been introduced by Chakraborty et al. in [27]. It is based on the observation that a simple change to the order of nodes in a network can significantly affect the community structure. However, a closer examination of the communities produced in different runs of a certain community detection algorithm reveals that for many networks the same invariant groups of nodes are consistently assigned to the same communities. Such groups of nodes are called constant communities. The percentage of constant communities varies depending on the network. Constant communities are detected by trying different node permutations while preserving the degree sequence of the nodes. For networks that have strong community structure, the constant communities detected can be adopted as a preprocessing step before performing modularity optimization. This can lead to higher modularity values and lower variability in results, thus improving the overall quality of community detection.

In the study [37] by Li, Zhang et al., a new quantitative measure for community detection is introduced. It offers several improvements over the modularity ($Q$), including elimination of the resolution limit and ability to detect the number of communities. The new measure called modularity density ($D$) is based on the average degree of the community structure. It is given by:

$$D = \sum_{c_i \in C} \frac{2 |E_{c_i}^{in} | - |E_{c_i}^{out} |}{|c_i|}. \quad (36)$$

The quality of the communities found is then described by the value of the modularity density ($D$). The larger the value of $D$, the stronger the community structure is.

The modularity density ($D$) does not divide a clique into two parts, and it can resolve most modular networks correctly. It can also detect communities of different sizes. This second property can be used to quantitatively determine the number of communities, since the maximum $D$ value is achieved when the network is supposed to correctly partitioned. Although as mentioned in [37] finding an optimal value of modularity density ($D$) is NP-hard, it is equivalent to an objective function of the kernel $k$ means clustering problem for which efficient computational algorithms are known.

Traag et al. in [38] introduce a rigorous definition of the resolution-limit-free method for which considering any induced subgraph of the original graph does not cause the detected community structure to change. In other words, if there is an optimal partitioning of a network (with respect to some objective function), and for each subgraph induced by the partitioning it is also optimal, then such objective function is called resolution-limit-free. An objective function is called additive for a certain partitioning if it is equal to the sum of the values of this objective function for each of the subgraphs induced by the partitioning.

Based on these two definitions it is proved that if an objective function is additive and there are two optimal partitions, then any combination of these partitions is also optimal. In case of a complete graph, if an objective function is resolution-limit-free, then an optimal partitioning either contains all the nodes (i.e. there is only one community which includes all nodes) or consists of communities of size 1 (i.e. each node forms a community of its own). A more general statement for arbitrary objective functions is also true: if an objective function has local weights (i.e. weights that do not change when considering subgraphs) then it is resolution-limit-free. Although the converse is not true, there is only a relatively small number of special cases when methods with non-local weights are resolution-limit-free.

The authors then analyze resolution-limit-free within the framework of the first principle Potts model [39]:

$$H = - \sum_{ij} (a_{ij} A_{ij} - b_{ij} (1 - A_{ij})) \delta_{c_i, c_j}, \quad (37)$$

where $a_{ij}, b_{ij} \geq 0$ are some weights. The intuition behind this formula is that a community should have more edges inside it than edges which connect it to other communities. Thus, it is necessary to reward existing links inside a community and penalize links that are missing from a community. The smaller the value of $H$ is, the more desirable the community structure is. However the minimal value might not be unique.

Given the definition of $H$, it is possible to describe various existing community detection methods with an appropriate choice of parameters, as well as propose alternative methods. The following models are shown to fit into $H$: Reichardt and Bornholdt (RB), Arenas, Fernández, and Gómez (AFG), Ronhovde and Nussinov (RN) as well as the label propagation method. RB approach with a configuration null model also covers the original definition of modularity. The authors also propose a new method called constant Potts model (CPM) by choosing $a_{ij} = w_{ij} - b_{ij}$ and $b_{ij} = \gamma$ where $w_{ij}$ is the weight of the edge between nodes $i$ and $j$, and $\gamma$ is a constant. CPM is similar to RB and RN models but is simpler and more intuitive. CPM and RN have
local weights and are consequently resolution-limit-free, while RB, AFG, and modularity are not.

However, all of the above approaches are aimed at solving only the resolution limit problem. Work done by Chen et al. in [23], [24] adopts a different definition of modularity density which simultaneously addresses two problems of modularity: the problem of favoring small communities and the problem of favoring large communities. It is done by mixing two additional components, Split Penalty (SP) and the community density, into the well-known definition of modularity. Community density includes internal community density and pair-wise community density. Split Penalty (SP) is the fraction of edges that connect nodes of different communities:

$$SP = \sum_{c_i \in C} \sum_{c_j \in C, c_j \neq c_i} \left[ \frac{|E_{c_i,c_j}|}{2|E|} \right].$$  (38)

The value of Split Penalty is subtracted from modularity, while the value of the community density is added to modularity and Split Penalty. Introducing Split Penalty resolves the issue of favoring small communities. Community density eliminates the problem of favoring large communities (also known as the resolution limit problem). The Modularity Density ($Q_{ds}$) is then given by:

$$Q_{ds} = \sum_{c_i \in C} \left[ \frac{|E_{c_i}^{in}|}{|E|} - d_{c_i} - \left( \frac{2|E_{c_i}^{in}| + |E_{c_i}^{out}|}{2|E|} d_{c_i} \right)^2 \right] - \sum_{c_j \in C} \frac{|E_{c_i,c_j}|}{2|E|} d_{c_i,c_j},$$  (39)

where $d_{c_i}$ is the internal density of community $c_i$, and $d_{c_i,c_j}$ is the pair-wise density between community $c_i$ and community $c_j$.

Modularity Density ($Q_{ds}$) avoids falling into the trap of merging two or more consecutive cliques in the ring of cliques network or dividing a clique into two or more parts. It can also discover communities of different sizes. Thus, using $Q_{ds}$ solves both the resolution limit problem of modularity and the problem of splitting larger communities into smaller ones. Hence, $Q_{ds}$ is an very effective alternative to $Q$.

## 3 Fine-tuned Algorithm

In our previous papers [23], [24], we have given the definition of Modularity Density ($Q_{ds}$). With formal proofs and experiments on two real dynamic datasets (Senate dataset [40] and Reality Mining Bluetooth Scan data [41]) we demonstrated that $Q_{ds}$ solves the two opposite yet coexisting problems of modularity: the problem of favoring small communities and the problem of favoring large communities (also called the resolution limit problem).

Moreover, for a given community in $Q_{ds}$ defined by Equation (39), its internal and pair-wise densities and its split penalty are local components, which is related to the resolution-limit-free definition in [38]. Therefore, it is reasonable to expect that maximizing $Q_{ds}$ would discover more meaningful community structure than maximizing $Q$. In this section, we first illustrate why the greedy agglomerative algorithm for increasing $Q_{ds}$ cannot be adopted for optimizing $Q_{ds}$. Then, we propose a fine-tuned community detection algorithm that repeatedly attempts to improve the community quality measurements by splitting and merging the given network community structure to maximize $Q_{ds}$.

### 3.1 Greedy Algorithm Fails to Optimize $Q_{ds}$

In this subsection, we show why the greedy agglomerative algorithm increasing $Q_{ds}$ fails to optimize it. At the first step of the greedy algorithm for increasing $Q_{ds}$, each node is treated as a single community. Then, $Q_{ds}$ of each node or community is $Q_{ds} = -SP$. Therefore, in order to increase $Q_{ds}$ the most, the greedy algorithm would first merge the connected pair of nodes with the sum of their degrees being the largest among all connected pairs. However, it is very likely that those two nodes belong to two different communities, which would finally result in merging those two communities instead of keeping them separate. This will result in a much lower value of $Q_{ds}$ for such a merged community compared to $Q_{ds}$ for its components, demonstrating the reason for greedy $Q_{ds}$ algorithm failure in optimizing $Q_{ds}$.

For example, in the network example in Figure 1, the initial values of $Q_{ds}$ for nodes 1, 2, 4, 6, 7, and 8 with degree 3 are $Q_{ds} = -SP = -\frac{1}{26}$ while the initial values of $Q_{ds}$ for nodes 3 and 5 with degree 4 are $Q_{ds} = -SP = -\frac{1}{26}$. Then, greedy $Q_{ds}$ algorithm would first merge node 3 and node 5, which would finally lead to a single community of the whole eight nodes. However, the true community structure contains two clique communities. Accordingly, the $Q_{ds}$ of the community structure with two clique communities, 0.4183, is larger than that of the community structure with one single large community, 0.2487. So, maximizing $Q_{ds}$ properly should have the ability to discover the true community structure.
3.2 Fine-tuned Algorithm

In this part, we describe a fine-tuned community detection algorithm that iteratively improves a community quality metric $M$ by splitting and merging the given network community structure. We denote the corresponding algorithm based on modularity ($Q$) as Fine-tuned $Q$ and the one based on Modularity Density ($Q_{ds}$) as Fine-tuned $Q_{ds}$. It consists of two alternating stages: split stage and merging stage.

Algorithm 1 Split_Communities($G, C$)

1: Initialize comWeights[$|C|][|C|], comEdges[$|C|][|C|], and comDensities[$|C|][|C|] which respectively contain #weights, #edges, and the density inside the communities and between two communities by using the network $G$ and the community list $C$;
2: //Get the metric value for each community.
3: $Mes[|C|] = GetMetric(C, comWeights, comDensities);$
4: for $i = 0$ to $|C| - 1$
5: $c = C.get(i);$
6: subnet = GenerateSubNetwork($c$);
7: FiedlerVector[$|c|] = LanczosMethod(subnet);
8: nodeId$[c][i] = sort(fiedlerVector, ’descend’);
9: //Form $|c| + 1$ divisions and record the best one.
10: SplitTwoCom.addAll(nodeIds$);
11: for $j = 0$ to $|c| - 1$
12: splitOneCom.add(nodeIds$[$j$]);
13: splitTwoCom.remove(nodeIds$[$j$]);
14: Calculate $M(split)$ for the split at $j$;
15: $\Delta M = M(split) - Mes[i];$
16: if $\Delta M(best) < \Delta M$ (or $\Delta M(best) > \Delta M$) then
17: $\Delta M(best) = \Delta M;$
18: $bestIdx = j;$
19: end if
20: end for
21: if $\Delta M(best) > 0$ (or $\Delta M(best) < 0$) then
22: Clear splitOneCom and splitTwoCom;
23: splitOneCom.addAll(nodeIds$[0:bestIdx]);
24: splitTwoCom.addAll(nodeIds$[bestIdx + 1:|c| - 1]);
25: newC.add(splitOneCom);
26: newC.add(splitTwoCom);
27: else
28: newC.add($c$);
29: end if
30: end if
31: return newC

In the split stage, the algorithm will split a community $c$ into two subcommunities $c_1$ and $c_2$ based on the ratio-cut method if the split improves the value of the quality metric. The ratio-cut method [42] finds the bisection that minimizes the ratio $\frac{|E_{c_1,c_2}|}{|c_1||c_2|}$, where $|E_{c_1,c_2}|$ is the cut size (namely, the number of edges between communities $c_1$ and $c_2$), while $|c_1|$ and $|c_2|$ are sizes of the two communities. This ratio penalizes situations in which either of the two communities is small and thus favors balanced divisions over unbalanced ones. However, graph partitioning based on the ratio-cut method is a NP-complete problem. Thus, we approximate it by using the Laplacian spectral bisection method for graph partitioning introduced by Fiedler [43], [44].

First, we calculate the Fiedler vector which is the eigenvector of the network Laplacian matrix $L = D - A$ corresponding to the second smallest eigenvalue. Then, we put the nodes corresponding to the positive values of the Fiedler vector into one group and the nodes corresponding to the negative values into the other group. The subnetwork of each community is generated with the nodes and edges in that community. Although the ratio-cut approximated with spectral bisection method does allow some deviation for the sizes $|c_1|$ and $|c_2|$ to vary around the middle value, the right partitioning may not actually divide the community into two balanced or nearly balanced ones. Thus, it is to some extent inappropriate and unrealistic for community detection problems.

We overcome this problem by using the following strategies. First, we sort the elements of the Fiedler vector in descending order, then cut them into two communities in each of the $|c| + 1$ possible ways and calculate the corresponding change of the metric values $\Delta M$ of all the $|c| + 1$ divisions. Then, the one with the best value (largest or smallest depending on the measurement) of the quality metric $\Delta M(best)$ among all the $|c| + 1$ divisions is recorded. We adopt this best division to the community $c$ only when $\Delta M(best) > 0$ (or $\Delta M(best) < 0$ depending on the metric). For instance, we split the community only when $\Delta Q_{ds}(best)$ is larger than zero.

The outline of the split stage is shown in Algorithm 1. The input is a network and a community list, and the output is a list of communities after splitting. The initialization part has $O(|E|)$ complexity. Computing Fiedler vector using Lanczos method [28] needs $O(|E|K + |V|K^2h + K^3h)$ steps, where $K$ is the number of eigenvectors needed and $h$ is the number of iterations required for the Lanczos method to converge. Here, $K$ is 2 and $h$ is typically very small although the exact number is not generally known. So, the complexity for calculating Fiedler vector is $O(|E| + |V|)$. Sorting the Fiedler vector has the cost $O(|V|\log|V|)$. The search of the best division from all the $|c| + 1$ possible ones (per community $c$) for all the communities is achieved in $O(|E|)$ time. For the $|c| + 1$ possible divisions of a community $c$, each one differs from the previous one by the movement of just a single node from one group to the other. Thus, the update of the total weights, the total number of edges, and the densities inside those two split communities and between those two communities to other communities can be calculated in time proportional to the degree of that node. Thus, all nodes can be moved in time proportional to the sum of their degrees which is equal to $2|E|$. Moreover, for Fine-tuned $Q_{ds}$, computing $Q_{ds}(split)$ costs $O(|C||V|)$ because all the communities are traversed to update the Split Penalty for each of the $|c| + 1$ divisions of each community $c$. All the other parts have complexity less than or at most $O(|V|)$. Thus, the

In the split stage, the algorithm will split a community $c$ into two subcommunities $c_1$ and $c_2$ based on the ratio-cut method if the split improves the value of the quality metric. The ratio-cut method [42] finds the bisection that minimizes the ratio $\frac{|E_{c_1,c_2}|}{|c_1||c_2|}$, where $|E_{c_1,c_2}|$ is the cut size (namely, the number of edges between communities $c_1$ and $c_2$), while $|c_1|$ and $|c_2|$ are sizes of the two communities. This ratio penalizes situations in which either of the two communities is small and thus favors balanced divisions over unbalanced ones. However,
computational complexity for the split stage of Fine-tuned
Q is \( O(|E| + |V|\log |V|) \) while for Fine-tuned \( Q_{ds} \) it is
\( O(|E| + |V|\log |V| + |C||V|) \).

### Algorithm 2 Merge _Communities_(G, C)

1. Initialize \( \text{comWeights}[|C|][|C|], \text{comEdges}[|C|][|C|] \),
and \( \text{comDensities}[|C|][|C|] \);
2. //Get the metric value for each community.
3. \( \text{Mes}[|C|] = \text{GetMetric}(C, \text{comWeights}, \text{comDensities}) \);
4. for \( i = 0 \) to \( |C| - 1 \) do
5.   for \( j = i + 1 \) to \( |C| - 1 \) do
6.     //Don’t consider disconnected communities.
7.     if \( \text{comWeights}[i][j] == 0 \) \&\&
        \( \text{comWeights}[j][i] == 0 \) then
8.         continue;
9.     end if
10.   end for
11. end for
12. //Merge the community with the one that improves
    the value of the quality metric the most
13. while \( \text{mergedInfos}.\text{hasNext()} \) do
14.   \( [\Delta M, \text{comId1}, \text{comId2}] = \text{mergedInfos.next()} \);
15.   if \( \text{mergedComs}.\text{containsKey}(\text{comId1}) \) \&\&
       \( \text{mergedComs}.\text{containsKey}(\text{comId2}) \) then
16.      \( \text{mergedComs}.\text{put}(\Delta M, \text{comId1}, \text{comId2}) \);
17.      \( \text{mergedComs}.\text{put}(\text{comId2}, \text{comId1}) \);
18.   end if
19. end while
20. for \( i = 0 \) to \( |C| - 1 \) do
21.   \( c_i = C.\text{get}(i) \);
22.   if \( \text{mergedComs}.\text{containsKey}(i) \) then
23.      \( \text{comId2} = \text{mergedComs}.\text{get}(i) \);
24.      if \( i < \text{comId2} \) then
25.        \( c_i.\text{addAll}(C, \text{get}(\text{comId2})) \);
26.      end if
27.   end if
28.   newC.add(c_i);
29. end for
30. return newC;

In the merging stage, the algorithm will merge a
community to its connected communities if the merging
improves the value of the quality metric. If there
are many mergers possible for a community, the one,
unmerged so far, which improves the quality metric
the most is chosen. Hence, each community will only
be merged at most once in each stage. The outline of
the merging stage is shown Algorithm 2. The input is
a network and a community list, and the output is a
list of communities after merging. The initialization part
has the complexity \( O(|E|) \). For Fine-tuned \( Q \), the two
“for loops” for merging any two communities have the
complexity \( O(|C|^22\log |C|) \) because calculating \( Q(\text{merge}) \)
is \( O(1) \) and inserting an element into the red-black tree is
\( O(|C|^2 = O(2|\log C|) \) \&\& \( O(|C|) \) since the maximum
number of elements in the tree is \( \frac{|C|(|C|-1)}{2} = O(|C|^2) \).
For Fine-tuned \( Q_{ds} \), the two “for loops” for merging any
two communities have the complexity \( O(|C|^2) \) because calculating \( Q_{ds}(\text{merge}) \) needs \( O(|C|) \) steps to traverse all
the communities to update the Split Penalty and inserting
an element into the red-black tree is \( O(|C|) \) as well. The other parts all have complexity at most \( O(|C|^2) \).
Thus, the computational complexity for the merging stage of Fine-tuned \( Q \) is \( O(|E| + |C|^22\log |C|) \) and for the
merging stage of Fine-tuned \( Q_{ds} \) is \( O(|E| + |C|^3) \).

### Algorithm 3 Fine-tuned _Algorithm_

1: \( \text{comSize} = |C|; \)
2: \( \text{splitSize} = 0; \)
3: \( \text{mergeSize} = 0; \)
4: while \( \text{comSize}! = \text{splitSize} \) || \( \text{comSize}! = \text{mergeSize} \) do
5:   \( \text{comSize} = |C|; \)
6:   \( C = \text{Split}_\text{Communities}(G, C); \)
7:   \( \text{splitSize} = |C|; \)
8:   \( \text{mergeSize} = |C|; \)
9: end while
10: return \( C \)

The fine-tuned algorithm repeatedly carries out those
two alternating stages until neither split nor merging can
improve the value of the quality metric or until the total
number of communities discovered does not change
after one full iteration. Algorithm 3 shows the outline of
the fine-tuned algorithm. It can detect the community
structure of a network by taking a list with a single
community of all the nodes in the network as the input.
It can also improve the community detection results of
other algorithms by taking a list with their communities
as the input. Let the number of iteration of the fine-tuned
algorithm be denoted as \( T \). Then, the total complexity
for Fine-tuned \( Q \) is \( O(T(|E| + |V|\log |V| + |C|^22\log |C|)) \) while for Fine-tuned \( Q_{ds} \) it is \( O(T(|E| + |V|\log |V| + |C||V| + |C|^3)) \). Assuming that \( T \) and \( |C| \) are constants, the
complexity of the fine-tuned algorithms reduces to
\( O(|E| + |V|\log |V|) \). The only part of the algorithm that
would generate a non-deterministic result is the Lanczos
method of calculating the Fiedler vector. The reason is that
Lanczos method adopts a randomly generated vector
as its starting vector. We solve this issue by choosing
a normalized vector of the size equal to the number
of nodes in the community as the starting vector for the
Lanczos method. Then, community detection results will
stay the same for different runs as long as the input
remains the same.

### 4 Experimental Results

In this section, we first introduce several popular measures
for evaluating the quality of the results of
community detection algorithms. Denoting the greedy algorithm of modularity maximization proposed by Newman [7] as Greedy $Q$, we then use the mentioned above metrics to compare Greedy $Q$, Fine-tuned $Q$, and Fine-tuned $Q_{ds}$. The comparison uses four real networks, the classical clique network and the LFR benchmark networks, each instance of which is defined with parameters each selected from a wide range of possible values. The results indicate that Fine-tuned $Q_{ds}$ is the most effective method among the three, followed by Fine-tuned $Q$. Moreover, we show that Fine-tuned $Q_{ds}$ can be applied to significantly improve the detection results of other algorithms.

In Subsection 2.2.2, we have shown that the modularity maximization approach using the eigenvectors of the Laplacian matrix is equivalent to the one using the eigenvectors of the modularity matrix. This implies that the split stage of our Fine-tuned $Q$ is actually equivalent to the spectral methods. Therefore, Fine-tuned $Q$ with one additional merge operation at each iteration unquestionably has better performance than the spectral algorithms. Hence, we do not discuss them here.

### 4.1 Evaluation Metrics

The quality evaluation metrics we consider here can be divided into three categories: Variation of Information ($VI$) [45] and Normalized Mutual Information ($NMI$) [46] based on information theory; F-measure [47] and Normalized Van Dongen metric ($NVD$) [48] based on cluster matching; Rand Index ($RI$) [49], Adjusted Rand Index ($ARI$) [50], and Jaccard Index ($JI$) [51] based on pair counting.

#### 4.1.1 Information Theory Based Metrics

Given partitions $C$ and $C'$, Variation of Information ($VI$) [45] quantifies the “distance” between those two partitions, while Normalized Mutual Information ($NMI$) [46] measures the similarity between partitions $C$ and $C'$. $VI$ is defined as

$$VI(C, C') = H(C) + H(C') - 2I(C, C') = H(C) - I(C, C'),$$

where $H(.)$ is the entropy function and $I(C, C') = H(C) + H(C') - H(C, C')$ is the Mutual Information. Then, $NMI$ is given by

$$NMI(C, C') = \frac{2I(C, C')}{H(C) + H(C')}.$$  

Using the definitions

$$H(C) = - \sum_{c_i \in C} p(c_i) \log p(c_i) = - \sum_{c_i \in C} \frac{|c_i|}{|V|} \log |c_i|,$$

$$H(C, C') = - \sum_{c_i, c_j' \in C, c_j' \in C'} p(c_i, c_j') \log p(c_i, c_j') = - \sum_{c_i, c_j' \in C, c_j' \in C'} \frac{|c_i \cap c_j'|}{|V|} \log \left( \frac{|c_i \cap c_j'|}{|V|} \right),$$

we can express $VI$ and $NMI$ as a function of counts only as follows:

$$VI(C, C') = - \frac{1}{|V|} \sum_{c_i, c_j \in C, c_j' \in C'} |c_i \cap c_j'| \log \left( \frac{|c_i \cap c_j'|^2}{|c_i||c_j'|} \right),$$

$$NMI(C, C') = - \frac{2}{|V|} \sum_{c_i, c_j \in C, c_j' \in C'} \frac{|c_i \cap c_j'|}{|V|} \log \left( \frac{|c_i \cap c_j'|}{|c_i||c_j'|} \right),$$

where $|c_i|$ is the number of nodes in community $c_i$ of $C$ and $|c_i \cap c_j'|$ is the number of nodes both in community $c_i$ of $C$ and in community $c_j'$ of $C'$.

#### 4.1.2 Clustering Matching Based Metrics

Measurements based on clustering matching aim at finding the largest overlaps between pairs of communities of two partitions $C$ and $C'$. F-measure [47] measures the similarity between two partitions, while Normalized Van Dongen metric ($NVD$) [48] quantifies the “distance” between partitions $C$ and $C'$. F-measure is defined as

$$F-measure(C, C') = \frac{1}{|V|} \sum_{c_i \in C} |c_i| \max_{c_j' \in C'} \frac{2|c_i \cap c_j'|}{|c_i| + |c_j'|}.$$  

$NVD$ is given by

$$NVD(C, C') = 1 - \frac{1}{2|V|} \left( \sum_{c_i \in C} \max_{c_j' \in C'} \frac{|c_i \cap c_j'|}{|c_i|} + \sum_{c_j' \in C'} \max_{c_i \in C} \frac{|c_i \cap c_j'|}{|c_j'|} \right).$$

#### 4.1.3 Pair Counting Based Metrics

Metrics based on pair counting count the number of pairs of nodes that are classified (in the same community or in different communities) in two partitions $C$ and $C'$. Let $a_{ij}$ indicate the number of pairs of nodes that are in the same community in both partitions, $a_{i0}$ denote the number of pairs of nodes that are in the same community in partition $C$ but in different communities in $C'$, $a_{0i}$ be the number of pairs of nodes which are in different communities in $C$ but in the same community in $C'$, $a_{00}$ be the number of pairs of nodes which are in different communities in both partitions. By definition, $A = a_{11} + a_{10} + a_{01} + a_{00} = \frac{|V|(|V|-1)}{2}$ is the total number of pairs of nodes in the network. Then, Rand Index ($RI$) [49] which is the ratio of the number of node pairs placed in the same way in both partitions to the total number of pairs is given by

$$RI(C, C') = \frac{a_{11} + a_{00}}{A},$$

Denote $M = \frac{1}{2}(a_{11} + a_{10})(a_{11} + a_{01})$. Then, $RI$’s corresponding adjusted version, Adjusted Rand Index ($ARI$) [50], is expressed as

$$ARI(C, C') = \frac{a_{11} - M}{\frac{1}{2}[(a_{11} + a_{10}) + (a_{11} + a_{01})] - M}.$$
TABLE 1
Metric values of the community structures discovered by Greedy $Q$, Fine-tuned $Q$, and Fine-tuned $Q_{ds}$ on Zachary’s karate club network (red italic font denotes the best value for each metric).

| Algorithm          | $Q$    | $Q_{ds}$ | $VI$ | $NMI$ | F-measure | $NVD$ | $RI$ | $ARI$ | $JI$ |
|--------------------|--------|----------|------|-------|-----------|-------|------|-------|------|
| Greedy $Q$         | 0.3807 | 0.1809   | 0.7677 | 0.6925 | 0.828     | 0.1471 | 0.8414 | 0.6803 | 0.6833 |
| Fine-tuned $Q$     | 0.4198 | 0.2302   | 0.9078 | 0.6873 | 0.807     | 0.1618 | 0.7736 | 0.5414 | 0.5348 |
| Fine-tuned $Q_{ds}$| 0.4174 | 0.2310   | 0.8729 | 0.6956 | 0.8275    | 0.1471 | 0.7861 | 0.5669 | 0.5604 |

(a) Ground truth communities. 
(b) Communities detected with Greedy $Q$. 
(c) Communities detected with Fine-tuned $Q$. 
(d) Communities detected with Fine-tuned $Q_{ds}$. 

Fig. 2. The community structures of the ground truth communities and those detected by Greedy $Q$, Fine-tuned $Q$, and Fine-tuned $Q_{ds}$ on Zachary’s karate club network.

The Jaccard Index ($JI$) [51] which is the ratio of the number of node pairs placed in the same community in both partitions to the number of node pairs that are placed in the same group in at least one partition is defined as

$$JI(C, C') = \frac{a_{11}}{a_{11} + a_{10} + a_{01}}.$$  \hspace{1cm} (50)

Each of these three metrics quantifies the similarity between two partitions $C$ and $C'$.

4.2 Real Networks

In this subsection, we first evaluate the performance of Greedy $Q$, Fine-tuned $Q$, and Fine-tuned $Q_{ds}$ on two small networks (Zachary’s karate club network [52] and American college football network [53]) with ground truth communities, and then on two large networks (PGP network [54] and AS level Internet) but without ground truth communities.

4.2.1 Zachary’s Karate Club Network

We first compare the performance of Greedy $Q$, Fine-tuned $Q$, and Fine-tuned $Q_{ds}$ on Zachary’s karate club network [52]. It represents the friendships between 34 members of a karate club at a US university over a period of 2 years. During the observation period, the club split into two clubs as a result of a conflict within the organization. The resulting two new clubs can be treated as the ground truth communities whose structure is shown in Figure 2(a) visualized with the opensource software Gephi [55].

Table 1 presents the metric values of the community structures detected by the three algorithms on this network. It shows that Fine-tuned $Q$ and Fine-tuned $Q_{ds}$ achieve the highest value of $Q$ and $Q_{ds}$ respectively. However, most of the seven metrics based on ground truth communities imply that Greedy $Q$ performs the best with only $NMI$ and $NVD$ indicating that Fine-tuned $Q_{ds}$ has the best performance among the three algorithms. Hence, it seems that a large $Q$ or $Q_{ds}$ may not necessary mean a high quality of community structure, especially for $Q$ because Fine-tuned $Q$ achieves the highest $Q$ but has the worst values of the seven metrics described in Subsection 4.1. We argue that the ground truth communities may not be so reasonable because Fine-tuned $Q$ and Fine-tuned $Q_{ds}$ in fact discover more meaningful communities than Greedy $Q$ does. Figures 2(a)-2(d) show...
TABLE 2  
Metric values of the community structures detected by Greedy Q, Fine-tuned Q, and Fine-tuned Q_{ds} on American college football network (red italic font denotes the best value for each metric).

| Algorithm                        | Q   | Q_{ds}  | V1   | NMI  | F-measure | NVD  | RI   | ARI  | JI   |
|----------------------------------|-----|---------|------|------|-----------|------|------|------|------|
| Greedy Q                         | 0.57/75 | 0.3225 | 1.4799 | 0.7624 | 0.6799 | 0.2304 | 0.9005 | 0.5864 | 0.4142 |
| Fine-tuned Q                     | 0.5944 | 0.3986 | 0.9615 | 0.8553 | 0.8067 | 0.1348 | 0.7229 | 0.6045 |
| Fine-tuned Q_{ds}                | 0.6005 | 0.4909 | 0.5067 | 0.9242 | 0.9145 | 0.0751 | 0.9847 | 0.8967 | 0.8264 |

TABLE 3  
Metric values of the community structures of Greedy Q and Fine-tuned Q improved with Fine-tuned Q_{ds} on American college football network (blue italic font indicates improved score).

| Algorithm                        | Q   | Q_{ds}  | V1   | NMI  | F-measure | NVD  | RI   | ARI  | JI   |
|----------------------------------|-----|---------|------|------|-----------|------|------|------|------|
| Greedy Q improved with Fine-tuned Q_{ds} | 0.5839 | 0.4636 | 0.6986 | 0.9013 | 0.8961 | 0.0913 | 0.9793 | 0.8597 | 0.7714 |
| Fine-tuned Q improved with Fine-tuned Q_{ds} | 0.5974 | 0.4793 | 0.5096 | 0.9278 | 0.9166 | 0.06957 | 0.9837 | 0.8907 | 0.8174 |

TABLE 4  
The values of Q and Q_{ds} of the community structures detected by Greedy Q, Fine-tuned Q, and Fine-tuned Q_{ds} on PGP network (red italic font denotes the best value for each metric).

| Algorithm                        | Q   | Q_{ds}  |
|----------------------------------|-----|---------|
| Greedy Q                         | 0.52 | 0.042   |
| Fine-tuned Q                     | 0.84 | 0.0206  |
| Fine-tuned Q_{ds}                | 0.39 | 0.258   |

the community structure of ground truth communities and those detected by Greedy Q, Fine-tuned Q, and Fine-tuned Q_{ds}, respectively. For results of Greedy Q shown in Figure 2(b), we could observe that there are three communities located at the left, the center, and the right side of the network. The ground truth community located on the right is subdivided into the central and right communities, but the node 10 is misclassified as belonging to the central community, while in ground truth network it belongs to community located on the left. Figure 2(c) demonstrates that Fine-tuned Q subdivides both the left and the right communities into two with six nodes separated from the left community and five nodes separated from the right community. Moreover, Figure 2(c) shows that Fine-tuned Q discovers the same number of communities for this network as algorithms presented in [9], [16], [20], [22]. In fact, the community structure it discovers is identical to those detected in [16], [20], [22]. Figure 2(d) shows that the community structure discovered by Fine-tuned Q_{ds} differs from that of Fine-tuned Q only on node 24 which is placed in the larger part of the left community. It is reasonable for it has three connections to the larger part to which it has more attraction than to the smaller part with which it only has two connections.

In addition, analyzing the intermediate results of Fine-tuned Q and Fine-tuned Q_{ds}, reveals that the communities at the first iteration are exactly the ground truth communities, which in another way implies their superiority over Greedy Q. Moreover, NMI and NVD indicate that Fine-tuned Q_{ds} is the best among the three and all the metrics, except Q, show that Fine-tuned Q_{ds} performs better than Fine-tuned Q, supporting the claim that a higher Q_{ds} (but not Q) implies a better quality of community structure.

4.2.2 American College Football Network

We apply the three algorithms also to the American college football network [53] which represents the schedule of games between college football teams in a single season. The teams are divided into twelve “conferences” with intra-conference games being more frequent than inter-conference games. Those conferences could be treated as the ground truth communities whose structure is shown in Figure 3(a).

Table 2 presents the metric values of the community structures detected by the three algorithms. It shows that Fine-tuned Q_{ds} achieves the best values for all the nine metrics. It implies that Fine-tuned Q_{ds} performs best on this football network, followed by Fine-tuned Q. Figures 3(a)-(d) present the community structure of ground truth communities and those discovered by Greedy Q, Fine-tuned Q, and Fine-tuned Q_{ds}. Each color in the figures represents a community. It can be seen that there are twelve ground truth communities in total, seven communities detected by Greedy Q, nine communities discovered by Fine-tuned Q, and exactly twelve communities found by Fine-tuned Q_{ds}.

Moreover, we apply Fine-tuned Q_{ds} on the community detection results of Greedy Q and Fine-tuned Q. The metric values of these two community structures after improvement with Fine-tuned Q_{ds} are shown in Table 3. Compared with those of Greedy Q and Fine-tuned Q in Table 2, we could observe that the metric values are significantly improved with Fine-tuned Q_{ds}. Further, both improved community structures contain exactly twelve communities, the same number as the ground truth communities.

4.2.3 PGP Network

We then apply the three algorithms on PGP network [54]. It is the giant component of the network of users of the Pretty-Good-Privacy algorithm for secure information interchange. It has 10680 nodes and 24316 edges.

Table 4 presents the metric values of the community structures detected by the three algorithms. Since this
network does not have ground truth communities, we only calculate $Q$ and $Q_{ds}$ of these discovered community structures. The table shows that *Greedy Q* and *Fine-tuned $Q_{ds}$* achieve the highest value of $Q$ and $Q_{ds}$, respectively. It is worth to mention that the $Q_{ds}$ of *Fine-tuned $Q_{ds}$* is much larger than that of *Greedy Q* and *Fine-tuned $Q$*, which implies that *Fine-tuned $Q_{ds}$* performs best on PGP network according to $Q_{ds}$, followed by *Greedy Q*.

### 4.2.4 AS Level Internet

The last real network dataset that is adopted to evaluate the three algorithms is AS level Internet. It is a symmetrized snapshot of the structure of the Internet at the level of autonomous systems, reconstructed from BGP tables posted by the University of Oregon Route Views Project. This snapshot was created by Mark Newman from data for July 22, 2006 and has not been previously...
published. It has 22963 nodes and 48436 edges.

Table 5 presents the metric values of the community structures detected by Greedy $Q$, Fine-tuned $Q$, and Fine-tuned $Q_{ds}$ on AS level Internet (red italic font denotes the best value for each metric).}

| Algorithm                      | $Q$  | $Q_{ds}$ | VI   | NMI  | F-measure | NVD | RI  | ARI | JI  |
|-------------------------------|------|----------|------|------|-----------|-----|-----|-----|-----|
| Greedy $Q$                    | 0.8871 | 0.46   | 0.9333 | 0.8949 | 0.6889    | 0.2333 | 0.9687 | 0.6175 | 0.4615 |
| Fine-tuned $Q$                | 0.8871 | 0.46   | 0.9333 | 0.8949 | 0.6889    | 0.2333 | 0.9687 | 0.6175 | 0.4615 |
| Fine-tuned $Q_{ds}$           | 0.8738 | 0.8721 | 0     | 1    | 1         | 0   | 1   | 1   | 1   |

**Table 7**

Metric values of the community structures of Greedy $Q$ and Fine-tuned $Q$ improved with Fine-tuned $Q_{ds}$ on the classical clique network (blue italic font indicates improved score).

| Algorithm                          | $Q$  | $Q_{ds}$ | VI   | NMI  | F-measure | NVD | RI  | ARI | JI  |
|-----------------------------------|------|----------|------|------|-----------|-----|-----|-----|-----|
| Greedy $Q$ improved with Fine-tuned $Q_{ds}$ | 0.8721 | 0       | 1    | 1    | 0         | 1   | 1   | 1   | 1   |
| Fine-tuned $Q$ improved with Fine-tuned $Q_{ds}$ | 0.8721 | 0       | 1    | 1    | 0         | 1   | 1   | 1   | 1   |

4.3 Synthetic Networks

4.3.1 Clique Network

We now apply the three algorithms to the classical network example [23]–[25], displayed in Figure 4, which illustrates modularity ($Q$) has the resolution limit problem. It is a ring network comprised of thirty identical cliques, each of which has five nodes and they are connected by single edges. It is intuitively obvious that each clique forms a single community.

Table 6 presents the metric values of the community structures detected by the three algorithms. It shows that Greedy $Q$ and Fine-tuned $Q$ have the same performance. They both achieve the highest value of $Q$ but get about half of the value of $Q_{ds}$ of what Fine-tuned $Q_{ds}$ achieves. In fact, Fine-tuned $Q_{ds}$ finds exactly thirty communities with each clique being a single community. In contrast, Greedy $Q$ and Fine-tuned $Q$ discover only sixteen communities with fourteen communities having two cliques and the other two communities having a single clique. Also, we take the community detection results of Greedy $Q$ and Fine-tuned $Q$ as the input to Fine-tuned $Q_{ds}$ to try to improve those results. The metric values of the community structures after improvement with Fine-tuned $Q_{ds}$ are recorded in Table 7. This table shows that the community structures discovered are identical to that of Fine-tuned $Q_{ds}$, which means that the results of Greedy $Q$ and Fine-tuned $Q$ are dramatically improved with Fine-tuned $Q_{ds}$. Therefore, it can be concluded from Tables 6 and 7 that a larger value of $Q_{ds}$ (but not $Q$) implies a higher quality of the community structure. Moreover, $Q_{ds}$ solves the resolution limit problem of $Q$. Finally, Fine-tuned $Q_{ds}$ is effective in maximizing $Q_{ds}$ and in finding meaningful community structure.

**Fig. 4. A ring network made out of thirty identical cliques, each having five nodes and connected by single edges.**

4.3.2 LFR Benchmark Networks

To further compare the performance of Greedy $Q$, Fine-tuned $Q$, and Fine-tuned $Q_{ds}$, we choose the LFR benchmark networks [56] which have become a standard in the evaluation of the performance of community detection algorithms and also have known ground truth communities. The LFR benchmark network that we used here has 1000 nodes with average degree 15 and maximum degree 50. The exponent $\gamma$ for the degree sequence varies from 2 to 3. The exponent $\beta$ for the community size distribution ranges from 1 to 2. Then, four pairs of the exponents $(\gamma, \beta) = (2, 1), (2, 2), (3, 1), \text{and } (3, 2)$ are chosen in order to explore the widest spectrum of graph structures. The mixing parameter $\mu$ is varied.
from 0.05 to 0.5. It means that each node shares a fraction \((1 - \mu)\) of its edges with the other nodes in its community and shares a fraction \(\mu\) of its edges with the nodes outside its community. Thus, low mixing parameters indicate strong community structure. Also, we generate 10 network instances for each \(\mu\). Hence, each metric value in Tables 8-12 represents the average metric values of all 10 instances. Since the experimental results are similar for all four pairs of exponents \((\gamma, \beta) = (2,1), (2,2), (3,1), \) and \((3,2),\) for the sake of brevity, we only present the results for \((\gamma, \beta) = (2,1)\) here.

Tables 8-10 show the metric values of the community structures detected with Greedy \(Q\), Fine-tuned \(Q\), and Fine-tuned \(Q_{ds}\), respectively, on the LFR benchmark networks with \((\gamma, \beta) = (2,1)\) and \(\mu\) varying from 0.05 to 0.5. The red italic font in the table denotes that the corresponding algorithm achieves the best value for a certain quality metric among the three algorithms. The results in these tables show that Greedy \(Q\) obtains the best values for all the nine measurements when \(\mu = 0.05\), while Fine-tuned \(Q_{ds}\) achieves the highest values of \(Q_{ds}\) and the best values for almost all the seven metrics based on ground truth communities when \(\mu\) ranges from 0.1 to 0.5. Also, Fine-tuned \(Q\) gets the second best values for \(Q_{ds}\) and almost all the seven metrics in the same range of \(\mu\). However, for \(Q\) the best is Greedy \(Q\), followed by Fine-tuned \(Q\), and Fine-tuned \(Q_{ds}\) is the last.

In summary, the seven measurements based on ground truth communities are all consistent with \(Q_{ds}\), but not consistent with \(Q\). This consistency indicates the superiority of \(Q_{ds}\) over \(Q\) as a community quality metric. In addition, Fine-tuned \(Q_{ds}\) performs best among the three algorithms for \(\mu > 0.05\), which demonstrates that it is very effective and does a very good job in optimizing \(Q_{ds}\).

We then take the community detection results of Greedy \(Q\) and Fine-tuned \(Q\) as the input to Fine-tuned \(Q_{ds}\) to improve those results. The measurement values of the community structures after improvement with Fine-tuned \(Q_{ds}\) are displayed in Tables 11 and 12. The blue italic font in Table 11 and Table 12 implies that the metric value in these two tables is improved compared to the one in
Table 8 and that in Table 9, respectively. Then, compared with those of Greedy Q shown in Table 8 and those of Fine-tuned Q shown in Table 9, all measurements, except in some cases for Q, are significantly improved with Fine-tuned Qds. This again indicates that all the seven metrics described in Subsection 4.1 are consistent with Qds, but not consistent with Q. Interestingly, those results are even better than those of Fine-tuned Qds itself presented in Table 10. Thus, it can be concluded that Fine-tuned Qds is very powerful in improving the community detection results of other algorithms.

5 Conclusion
In this paper, we review the definition of modularity and its corresponding maximization methods. Moreover, we show that modularity optimization has two opposite but coexisting issues. We also review several community quality metrics proposed to solve the resolution limit problem. We then discuss our Modularity Density (Qds) metric which simultaneously avoids those two problems. Finally, we propose an efficient and effective fine-tuned algorithm to maximize Qds. This new algorithm can actually be used to optimize any community quality metric. We evaluate the three algorithms, Greedy Q, Fine-tuned Q based on Q, and Fine-tuned Qds based on Qds, with seven metrics based on ground truth communities.

These evaluations are done on four real networks, and also on the classical clique network and the LFR benchmark networks, each instance of the last is defined with parameters selected from wide range of their values. The results demonstrate that Fine-tuned Qds performs best among the three algorithms, followed by Fine-tuned Q. The experiments also show that Fine-tuned Qds can dramatically improve the community detection results of other algorithms. In addition, all the seven quality metrics based on ground truth communities are consistent with Qds, but not consistent with Q, which indicates the superiority of Qds over Q as a community quality metric.

Acknowledgments
This work was supported in part by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053 and by the the Office of Naval Research Grant No. N00014-09-1-0607. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies either expressed or implied of the Army Research Laboratory or the U.S. Government.

References
[1] M. E. J. Newman and M. Girvan, “Finding and evaluating community structure in networks,” Phys. Rev. E, vol. 69, p. 026113, Feb 2004.
[2] S. Fortunato, “Community detection in graphs,” Physics Reports, vol. 486, pp. 75–174, 2010.
[3] M. E. J. Newman, “Modularity and community structure in networks,” Proceedings of the National Academy of Sciences, vol. 103, no. 23, pp. 8577–8582, 2006.
[4] E. A. Leicht and M. E. J. Newman, “Community structure in directed networks,” Phys. Rev. Lett., vol. 100, p. 118703, Mar 2008.
[5] M. E. J. Newman, “Analysis of weighted networks,” Phys. Rev. E, vol. 70, p. 066131, Nov 2004.
[6] M. E. J. Newman, “Fast algorithm for detecting community structure in networks,” Phys. Rev. E, vol. 69, p. 066133, Jun 2004.
