Spectral Graph Multisection Through Orthogonality

Huanyang Zheng and Jie Wu
Department of Computer and Information Sciences
Temple University, Philadelphia, PA 19122
{huanyang.zheng, jiewu}@temple.edu

ABSTRACT

Although the spectral modularity optimization algorithm works well in most cases, it is not perfect, due to the characteristic of its recursive bisection, which loses "global" view. In this paper, we propose a spectral multisection algorithm, which cuts the graph into multisections directly, with acceptable time complexity. Instead of using -1 and $+1$ in the modularity bisection algorithm, we propose using orthogonal vectors of the Hadamard matrix, as to denote the group assignments in the graph division. Then the modularity matrix is "inflated" to higher order through the Kronecker product, which is able to coordinate with the vectors that represent the group assignments of the nodes. The relaxation method is also employed in our algorithm. The eigenvector, which corresponds to the largest eigenvalue of the inflated modularity matrix, reflects the final group assignment of the nodes. The proposed algorithm can be viewed as a natural extension of the original bisection algorithm, which also succeeds its properties. In sparse graphs, the time complexity of the proposed algorithm is $O(K^4n^2)$, where K is a carefully designed input parameter that reveals the estimated number of communities. Finally, the simulations show that the proposed algorithm achieves outstanding performances in the LFR benchmarks of different settings.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering*; G.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms*

General Terms

Algorithms, Design, Performance

Keywords

Community detection, orthogonality, relaxation, modularity, spectral multisection.

1. INTRODUCTION

Rather than being randomly wired together, the components of complex network systems represent a community structure, where highly concentrated edges are found within special groups of vertices, and low concentrated edges exist between different groups [1]. Communities, also called *modules*, are groups of vertices which have common features and similar network structures. Generally speaking, the communities of a graph can be viewed as highly independent ingredients, such as organs in a human body. Therefore, community detection has been a significant problem in network-related sciences, with huge efforts taking place over the past few decades. The applications of community detection range from social network science and biomedical component analysis, to business investment analysis.

However, community detection is an NP-hard problem, and is not yet perfectly solved. Although outstanding performances have been achieved by some algorithms such as [2], the time complexities of these algorithms are quite high. Meanwhile, algorithms [3] with linear time complexities do not have satisfactory performances. A tradeoff between the performance and the time complexity is desired, resulting in the emergence of the modern *spectral modularity optimization* algorithm [4]. This algorithm has both a competitive performance and a time complexity of $O(n^2 \log n)$ in sparse graphs, by exploring the relationship between community assignments of the nodes and the eigenvectors of the modularity matrix. Compared to [5], the spectral modularity optimization has worse time complexity, but its division results are of far better quality. The reason for its outstanding performance is that the spectral modularity optimization is a "relaxed" optimal division.

The spectral modularity optimization algorithm is a recursive bisection algorithm, which divides the graph into more than two communities through a repeated bisection process [6–8]. As described in [8], the recursive bisection algorithm is not optimal, since the recursion is a greedy process, regardless of the "global" structure information. As illustrated in Fig. 1(a), the spectral modularity optimization does the bisection in the first round, which cuts the graph into two halves (four left nodes in one community, and the remaining four right nodes in another). Then, in the second round, it stops, since the modularity does not increase if it further cuts the two communities. However, this division is not optimal, as illustrated in Fig. 1(b), which cuts the graph into three parts. Obviously, even the repeated optimal bisection algorithm would never find the global optimality, due to the loss of "global" view.

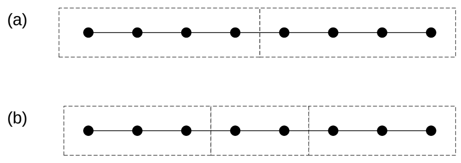


Figure 1: An illustration of dividing a graph into three communities. (a) The division result of the recursive bisection algorithm (spectral modularity optimization). (b) The optimal division which maximizes the whole modularity.

Naturally, spectral multisection algorithms [9] are focused, which removes the recursive division part, and cuts the graph into multisections directly. Theoretically, this approach can find better divisions due to the "global" view, while in practice it is more complicated and generally brings a longer running time. Therefore, our research motivation is to design *a "global" multisection algorithm with acceptable time complexity.* The proposed algorithm achieves better performances with almost the same time complexity of the recursive bisection algorithm. Instead of using $\{-1, +1\}$ to denote group assignment in the bisection algorithm, we propose employing orthogonal vectors of the Hadamard matrix to do it. Meanwhile, the modularity matrix is randomly "*inflated*" to higher orders through the Kronecker product, as to coordinate with these orthogonal vectors. Therefore, the graph can be cut into more than two sections directly, with acceptable time complexity.

The main contributions of this paper are summarized as follows: we introduce Hadamard matrix for orthogonal vectors generation and Kronecker product for matrix operation; then a spectral multisection algorithm is proposed, which directly cuts the graph into multisections, as to get rid of greedy division; finally, the time complexity of the proposed algorithm is analyzed, which is $O(K^4n^2)$ in sparse graphs (K is a predefined parameter which shows the estimated number of communities).

The remainder of the paper is organized as follows: in Section II, we present the preliminaries, which describes the spectral modularity optimization algorithm. Then we describe our multisection approach in Section III, including the Hadamard matrix and the Kronecker product. Then the evaluation of our algorithm is shown in Section V, which is based on the LFR benchmark. Finally, we conclude the paper in Section VI.

2. PRELIMINARIES

In this section, we will introduce the traditional spectral modularity maximization algorithm proposed by [4]. First, we discuss the literature of the community detection algorithms, where the concept of modularity and the traditional spectral bisection technology is presented. Then, we discuss the traditional spectral multisection technology, which is done through recursive bisection. In the following discussion, we tacitly assume the graph is undirected and unweighted, with n nodes and m edges.

2.1 Modularity

In the first generation graph partition algorithms, the objective is to minimize the number of edges between vertices in different groups, which is generally denoted as the *cut size.* However, minimizing cut size fails to recognize the complete network structure. For example, one optimal but incorrect solution for minimizing cut size can be dividing the original graph into two subgraphs: an empty subgraph which contains no node, and a subgraph which contains all the nodes. Since no node exists in the empty set, the cut size is 0 for the divided two groups of nodes, which is clearly meaningless. Intuitively, instead of simply minimizing the cut size (denoted as C_s), a better idea is to consider the group size (denoted as n_1, n_2) along with the cut size (for example, minimizing $C_s/(n_1n_2)$). In the literature, this approach is called *ratio cut partitioning*, which effectively avoids incorrect partitioning. However, in the ratio cut partitioning, the denominator n_1n_2 has no remarkable physical meanings. A simple but powerful argument is why we use $C_s/(n_1n_2)$ rather than $C_s/(n_1^2n_2^2)$ (or something else) as the partition metric. Mathematically, the denominator may be anything if it monotonically decreases with increasing $|n_2 - n_1|$. Moreover, the ratio cut partitioning becomes even more unreasonable, when more than two communities exist in the graph, since the denominator is mathematically defined without real-world meanings.

Rather than using cut size, a better idea is to develop a new graph partition measurement which reflects the network structure more. Instead of simply minimizing cut size, we can minimize the difference between the actual and the *expected* number of edges across different groups, which is called *modularity.* Assume A_{ij} is denoted as the element of the row i and column j in the graph adjacency matrix A, k_i is the degree of node i, and m is the number of edges in the graph. Then modularity Q is calculated as:

$$Q=\frac{1}{2m}\sum_{ij}\left(A_{ij}-\frac{k_ik_j}{2m}\right)\delta(c_i,c_j)=\frac{1}{2m}\sum_{ij}B_{ij}\delta(c_i,c_j) \qquad (1)$$

where A_{ij} and $k_ik_j/2m$ are, respectively, the actual and the expected number of edges between nodes i and j. Additionally, c_i is the group assignment of node i. $\delta(c_i, c_j)$ is the Kronecker delta. Here $B_{ij} = A_{ij} - k_ik_j/2m$ is considered as an element of a matrix B, which is called *modularity matrix.* Then we can minimize the modularity between different groups, or maximize the modularity in the same groups, as to detect communities. This idea is called modularity maximization, as presented in the next subsection.

2.2 Spectral Bisection

Let us start with the spectral bisection algorithm [8], which divides the graph into two non-overlay parts with maximized modularity Q. To represent Eq. 1 better, here we use s_i to denote the group assignment of node i rather than c_i:

$$s_i = \begin{cases} +1 & \text{if node } i \text{ belongs to group 1.} \\ -1 & \text{if node } i \text{ belongs to group 2.} \end{cases} \qquad (2)$$

Since $\delta(c_i, c_j) = \frac{1}{2}(s_is_j + 1)$ and $\sum_{ij} B_{ij} = 0$, we have

$$Q = \frac{1}{4m}\sum_{ij} B_{ij}(s_is_j + 1) = \frac{1}{4m}\sum_{ij} B_{ij}s_is_j \qquad (3)$$

If we use s to denote the vector with elements s_i, where $s = [s_1, ..., s_n]$, then Eq. 3 can be rewritten in the matrix form of $Q = \frac{1}{4m}s^TBs$. Since the constraint $s_i \in \{-1, +1\}$ is hard to deal with, we *relax* this constraint to be $s^Ts = \sum_i s_i^2 = n$, where n is the number of nodes in the graph. This method is called *relaxation method.* Then the modularity maximization is a straightforward optimization problem, which can be solved through a simple Lagrange multiplier β:

$$\frac{\partial}{\partial s_i}\left[\sum_{ij} B_{ij}s_i s_j + \beta(n - \sum_i s_i^2)\right] = 0 \quad (4)$$

which gives $\sum_j B_{ij}s_j = \beta s_i$, or in matrix notation, $Bs = \beta s$. It can be seen that the relaxed s should be an eigenvector of the modularity matrix B, and then we have

$$Q = \frac{1}{4m}s^T Bs = \frac{1}{4m}s^T \beta s = \frac{n}{4m}\beta \quad (5)$$

Since we are maximizing modularity, the relaxed s is selected to be the eigenvector u_1 corresponding to the largest eigenvalue of the modularity matrix B. Meanwhile, $s = u_1$ is "rounded" to be ± 1 (for each element in $s = u_1$, we round it to be 1 if it is larger than 0, and round it to be -1 if it is smaller than 0) since we have relaxed the original constraint $s_i \in \{-1, +1\}$. Then, the ith element in the rounded vector s represents the group assignment of node i, as shown in Eq. 2. It can be seen that this graph partition method is a good approximation of the optimal partitioning, since we have only relaxed the constraint a little. The time complexity of finding the leading eigenvector of a matrix is $O(mn)$, which is equivalent to $O(n^3)$ in a dense matrix, as opposed to $O(n^2)$ in a sparse one. However, there is a more efficient method for finding the eigenvector [4]. In consideration of the general case, we assume the graph is sparse and the time complexity of the bisection algorithm is $O(n^2)$.

2.3 Multisection through Recursive Bisection

Through spectral bisection algorithm, the graph is divided into two groups of unspecified size. However, the number of the natural groupings of nodes, or communities, may not be only two. And we would like to find the most "natural" division of the graph, without restrictions of the size and number of the partitions. In principle, the modularity maximization is still valid for graph multisection, as described in Eq. 1. However, spectral multisection algorithm is not as easy as it is supposed to be, since we have used $s_i \in \{-1, +1\}$ to represent group assignment in the bisection algorithm. Some multisection algorithms, such as [9], have been developed; they are promising, but bring higher time complexity.

Instead of directly finding the maximum modularity over divisions into any number of groups, an alternative but popular method is to do the recursive bisection [8]. That is, do the bisection recursively until the modularity is no longer increased. Note that, here we cannot proceed directly and treat the subdivisions as totally new graphs, applying the bisection algorithm to the subdivisions. Instead, we need to consider the entire change of modularity over the whole network [8]. Therefore, the modularity change $\triangle Q$ on further bisection of community c is calculated as:

$$\triangle Q = \frac{1}{4m}\left[\sum_{ij\in c} B_{ij}s_i s_j - \sum_{ij\in c} B_{ij}\right] \quad (6)$$

The recursive bisection algorithm works well in many situations, although optimality is not guaranteed, as shown in Fig. 1. The recursive depth of this algorithm depends on the average depth of the graph dendrogram, which is $O(\log n)$. Therefore, the time complexity of the whole algorithm is $O(n^2 \log n)$. This algorithm runs slower, but has much better performance than the linear partition algorithms, as the tradeoff between time complexity and partition accuracy.

Therefore, algorithms that improve the recursive bisection have been focused. In [6], the authors reformulated the relaxation process, as to pursue a lower time complexity. However, this algorithm does not get rid of "local" optimization, where the graph division is done through a greedy iteration. The same problem can be found in [7]. In the next section, we will introduce a direct multisection scheme through orthogonality to pursue better performance. Different from former algorithms, our approach does not include the iteration. It "globally" cuts the graph into multisections directly, which gets rid of "local" optimization.

3. SPECTRAL MULTISECTION

In this section, a novel multisection scheme is presented, which is based on the orthogonality. Firstly, the Hadamard matrix [10] is introduced to produce orthogonal vectors, with the Kronecker product to extend the matrix to high orders. Then the detailed algorithm is presented, which employs the orthogonal vectors to represent the group assignment of nodes rather than simply using $\{-1, +1\}$. Finally, the subsequent processing of the proposed algorithm is attached, as to achieve better division.

3.1 Hadamard Matrix

The Hadamard matrix is a square matrix composed by elements of $+1$ or -1. Its key feature is that the rows of the matrix are mutually orthogonal, which is used to produce orthogonal vectors. Moreover, every two different rows in Hadamard matrix have matching entries in exactly half of their columns, and mismatched entries in the remaining columns. For simplicity, we will use $K = 2^k$ in the following discussions. A Hadamard matrix H_{2K} can be calculated by

$$H_{2K} = \begin{bmatrix} +H_K & +H_K \\ +H_K & -H_K \end{bmatrix} \quad (7)$$

where $H_1 = [1]$. Hadamard matrix satisfies $H_K^T = H_K$ and $H_K^T H_K = nI_K$. Meanwhile, a Hadamard matrix has a maximal determinant among matrices with entries of an absolute value less than or equal to 1. For further analysis, we also define h_i to represent the ith row of H_K. Note that we have $h_i h_i^T = K$, since the length of h_i is K.

Here, we also define a new matrix operation called *matrix inflation*, which extends an original $n \times n$ matrix M to a $Kn \times Kn$ matrix $\overline{M}_K$ (K is a specified input parameter). This operation replaces each element in M by a $K \times K$ diagonal matrix. Essentially, the inflation operation is the Kronecker product of the original matrix M and a $K \times K$ identity matrix. An example for $K = 2$ is listed below:

$$M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \text{and} \quad \overline{M}_2 = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 3 & 0 & 4 & 0 \\ 0 & 3 & 0 & 4 \end{bmatrix} \quad (8)$$

Meanwhile, we also introduce a matrix operation called *randomized matrix inflation*, which is based on the matrix inflation. After inflating M to $\overline{M}_K$, this operation respectively adds a random number λ_{ij} to the element $\overline{M}_{K,ij}$ of the matrix $\overline{M}_K$, resulting in the matrix of $\widetilde{M}_K$. Here, λ_{ij} follows the uniform distribution of interval $[-\varepsilon, +\varepsilon]$, and are independent of one another. Therefore, we have $\sum_{ij} \overline{M}_{K,ij} = \sum_{ij} \widetilde{M}_{K,ij} + \sum_{ij} \lambda_{ij}$, where $\sum_{ij} \lambda_{ij} \to 0$ with increased size of $\overline{M}_K$, due to the central limit theorem. If Kn is large enough, $\delta = \sum_{ij} \lambda_{ij}$ would intend to follow the normal distribution of $N(0, Kn\varepsilon^2/3)$.

3.2 Multisection through Orthogonality

In the proposed algorithm, the number of communities needs to be pre-estimated (denoted as $K = 2^k$ communities), which is an input parameter. Here the predefined parameter K means the graph is divided into *at most* K communities, rather than exactly K communities. Then, instead of using $s_i \in \{-1, +1\}$ in the bisection algorithm, we use the rows of H_K to denote the group assignment of node i (to be different from s_i in the bisection, here we use $\bar{s}_i$):

$$\bar{s}_i = \begin{cases} h_1 & \text{if node } i \text{ belongs to group 1.} \\ h_2 & \text{if node } i \text{ belongs to group 2.} \\ \vdots & \vdots \\ h_K & \text{if node } i \text{ belongs to group } K. \end{cases} \tag{9}$$

Since $\delta(c_i, c_j) = \frac{1}{K}\bar{s}_i\bar{s}_j^T$, Eq. 1 changes to be

$$Q = \frac{1}{2m}\sum_{ij} B_{ij}\delta(c_i, c_j) = \frac{1}{2Km}\sum_{ij} B_{ij}\bar{s}_i\bar{s}_j^T \tag{10}$$

In Eq. 10, both $\bar{s}_i$ and $\bar{s}_j^T$ are vectors, and thus Eq. 10 is different from Eq. 3. Similarly, we define $\bar{s}$, which is equal to $[\bar{s}_1, \bar{s}_2, ..., \bar{s}_K]^T$. If we put matrix inflation operation on matrix B to obtain $\overline{B}_K$, then Eq. 10 is extended to be

$$Q = \frac{1}{2Km}\bar{s}^T\,\overline{B}_K\bar{s} \tag{11}$$

Note that the coefficient in Eq. 11 is $\frac{1}{2Km}$ rather than $\frac{1}{4m}$ in Eq. 5, which is due to the difference between s_i and $\bar{s}_i$. However, Eq. 11 is very similar to Eq. 3, since we just count in the same elements for K times ($\sum_{ij}\overline{B}_K = K\sum_{ij} B$). Then the following process is very similar to the bisection algorithm. We also relax the constraint that $\bar{s}_i$ belongs to a row of the Hadamard matrix to be $\bar{s}^T\bar{s} = \sum_i \bar{s}_i^T\bar{s}_i = Kn$. Through the method of Lagrange multipliers, we can also obtain the equation of $\overline{B}_K\bar{s} = \beta\bar{s}$, meaning that $\bar{s}$ should be the eigenvector $\bar{u}_1$ corresponding to the largest eigenvalue of the matrix $\overline{B}_K$. Meanwhile, $\bar{s}_i$ in $\bar{s} = \bar{u}_1$ is "rounded" to be the most "similar" vector in Eq. 9, as to represent the group assignment of the node i. In other words, $\bar{s}_i$ is rounded to h_j, if $\bar{s}_i h_j^T$ has the largest value among all rows of H_K.

It can be seen that the classic bisection algorithm is a special case of the proposed multisection algorithm, which subtly utilizes the orthogonality of the rows of the Hadamard matrix to represent the group assignments of the nodes. The proposed algorithm is a natural extension of the spectral bisection algorithm. However, there is a fatal flaw in the proposed spectral multisection algorithm. Due to the nature of the matrix inflation operation, the eigenvectors of $\overline{B}_K$ are greatly influenced by the eigenvectors of B, leading to meaningless $\bar{u}_1$ of $\overline{B}_K$. In the following subsection, this problem is described and then solved.

3.3 Subsequent Processing

The multisection algorithm proposed in the former section seems to work well, however, this algorithm does not work, due to the inherent defect of the matrix inflation operation. The eigenvectors of matrix $\overline{M}_K$ are related to the eigenvectors of the original matrix M. If $[e_1, e_2, ..., e_n]$ is an eigenvector of the matrix M, then it can be extended to be K different eigenvectors of matrix $\overline{M}_K$ by respectively filling z_1 zeros before, and z_2 zeros after, each element in the eigenvector of M. Here z_1 and z_2 are non-negative integers, and $z_1 + z_2 = K - 1$. For example, if $K = 2$, the eigenvector $[e_1, e_2, ..., e_n]$ of M can be extended to $[e_1, 0, e_2, 0, ..., e_n, 0]$ and $[0, e_1, 0, e_2, ..., 0, e_n]$, which are the eigenvectors of $\overline{M}_2$. This structure severely destroys the process of "rounding" $\bar{s} = \bar{u}_1$ to the rows of the corresponding Hadamard matrix, leading to an ineffective relaxation.

To overcome this flaw, $\overline{M}_K$ is replaced by $\widetilde{M}_K$ (randomized matrix inflation), which avoids the strong relationship between M and $\overline{M}_K$. The intuition behind this trick is to *protect the relaxation effectiveness in the inflation operation.* Since the $\widetilde{M}_K$ is randomized, we run the algorithm t times, and then pick out the best result (the highest Q) as the final partitioning result. To prove the effectiveness of this method, there are two points: (1) prove that the Q calculated by the $\widetilde{M}_K$ has limited error from the Q calculated by the $\overline{M}_K$; (2) make sure that a small t is good enough to obtain a qualified result, otherwise, the time complexity of the proposed algorithm is too large to be useful.

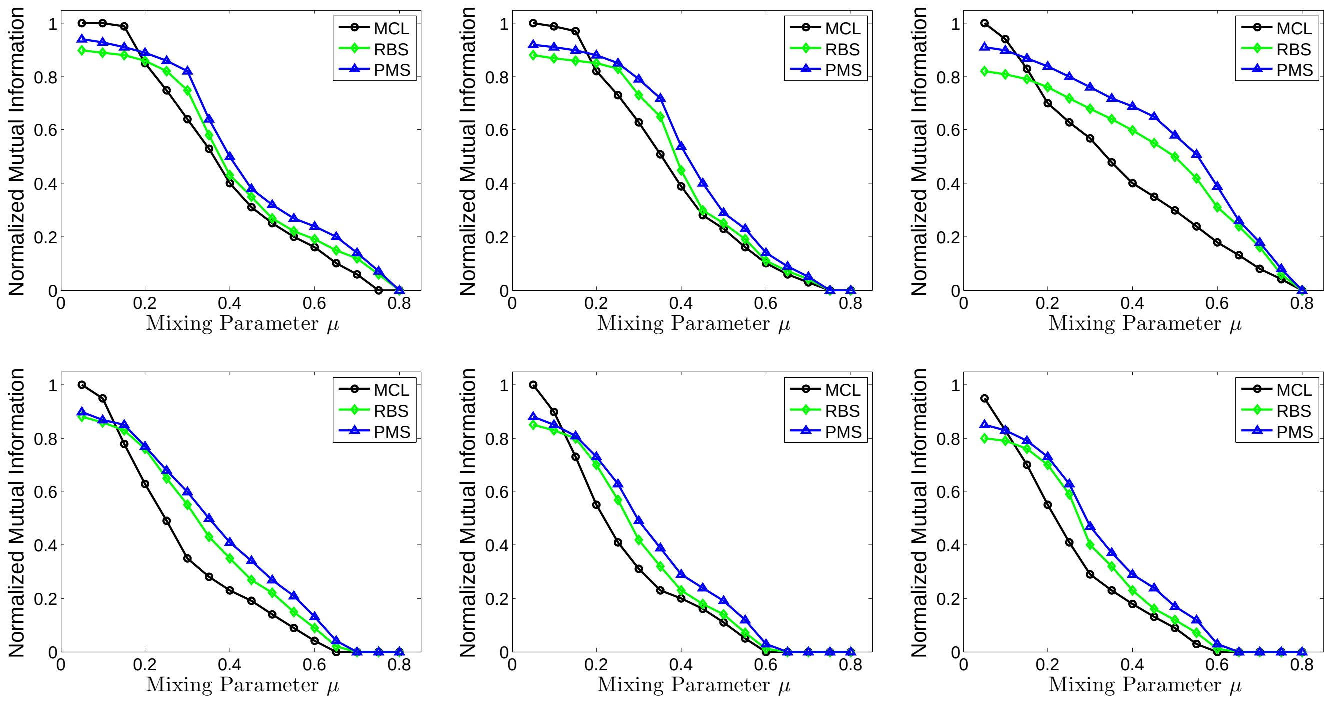
For the first point, it can be proved by central limit theorem. The variance $\triangle Q$ can be calculated by

$$\triangle Q = \frac{1}{2Km}\bar{s}^T(\widetilde{B}_K - \overline{B}_K)\bar{s} = \frac{1}{2Km}\delta \tag{12}$$

where δ is the summation of K^2n^2 random variables ($\delta = \sum_{ij}\lambda_{ij}$). These random variables are independent to each other, but follow the same uniform distribution $U[-\varepsilon, +\varepsilon]$. Here we have utilized the property, in that $-\lambda_{ij}$ have the same distribution as $+\lambda_{ij}$. Central limit theorem shows that δ tends asymptotically toward normal distribution of $N(0, Kn\varepsilon^2/3)$. If we use $\varepsilon = 1/n$, then the normal distribution becomes $N(0, K/3n)$ Since n is generally very large, it can be seen that $K/3n \to 0$. So we have $\triangle Q \to 0$, which proves the first point. Moreover, each element in B has very limited variance, when inflated to be a submatrix in $\widetilde{B}_K$.

For the second point, we empirically declare that $t \in O(K^2)$ is a good choice. Randomized matrix inflation effectively avoids equivalent treatment for the different original elements in M, each element of which is inflated to be a $K \times K$ submatrix in $\widetilde{M}$. Intuitively, the size of the submatrix dominates the value of t, and this is the reason why we choose $t \in O(K^2)$. Due to the difficulty of the mathematical analysis, we will show its efficiency in the simulations.

The proposed algorithm has a time complexity of $O(K^2n^2)$ for one loop. Since we have $t \in O(K^2)$ loops, the total time complexity is $O(K^4n^2)$. Note that the recursive bisection has a time complexity of $O(n^2 \log n)$, since the recursive depth is $O(\log n)$. Note that K is the estimated number of communities. Although, theoretically, the multisection algorithm is valid for K with large values, larger K brings longer running time, due to more information loss in the relaxation. Since generally we have $K \ll n$, meaning the number of communities is much smaller than the number of total nodes, time complexity of $O(K^4n^2)$ is acceptable as a tradeoff to pursue better graph division. In the next section, we will show the effectiveness of our proposal through extensive simulations.



(a) $\gamma = 3$, $\beta = 2$ (b) $\gamma = 2$, $\beta = 2$ (c) $\gamma = 2$, $\beta = 3$

Figure 2: Simulation results in LFR benchmarks of $n = 128$ (top) and $n = 256$ (bottom) nodes.

4. SIMULATION

In this section, extensive simulations are conducted to evaluate the proposed algorithm, which is based on the LFR benchmark [11]. LFR benchmark generated networks are composed of communities with different sizes. In LFR benchmark, both the node degrees and the community sizes follow power-law distribution. After presenting the settings used in our simulations, we show the algorithms for comparison. Finally, the evaluation results are shown from different perspectives to provide insightful conclusions.

4.1 Settings

In our simulation, LFR benchmark [11] of n nodes is introduced as the testbed, instead of the traditional GN benchmark. In LFR benchmark, the node degree and the community size follow power-law distribution, with exponents γ and β, respectively. Meanwhile, k_{min} and k_{max} are selected to make the average degree of the generated network to be $\langle k \rangle$. The minimal and maximal community sizes are also restricted as s_{min} and s_{max}, where $s_{min} > k_{min}$ and $s_{max} > k_{max}$ (each node can be surely included in a community). Then, the nodes are linked to each other. Links between nodes in the same community are called internal links, and links between nodes in different communities are called external links. A *mixing parameter* μ is defined as the ratio of the external node degree to the total node degree. Through a subtle rewiring process, the ratio between external and internal degrees of each node can be approximated to the designed mixing parameter μ. LFR benchmarks of undirected and unweighted graphs with $n = 128$ and $n = 256$ nodes are employed for our tests. For $n = 128$ nodes, we set $k_{min} = 8$, $k_{max} = 32$, $s_{min} = 16$ and $s_{max} = 64$. For $n = 256$ nodes, we set $k_{min} = 16$, $k_{max} = 64$, $s_{min} = 32$ and $s_{max} = 128$. In addition, for the proposed multisection algorithm specifically, we set $\varepsilon = 1$, $K = 8$, and $t = K^2 = 64$. Our simulations do not include real-world data, since LFR benchmark is highly recognized as a clustering benchmark.

To compare the partition results between different algorithms, the metric of *mutual information* is introduced, which is based on Shannon information theory. Let the set $\{x_i\}$ denote the community assignments of node i of the designed algorithm, and let set $\{y_i\}$ denote the correct results generated by the LFR benchmark; then mutual information $I(X, Y)$ shows how much information is mutual for the information from $\{x_i\}$ and $\{y_i\}$. This is done by assuming that labels $\{x_i\}$ and $\{y_i\}$ are values of the random variable X and Y, respectively. The distributions of X and Y are calculated by $P(x) = n_x^X/n$ and $P(y) = n_y^Y/n$, with their joint distribution being $P(x, y) = n_{xy}/n$. Here n_x^X is the size of the community labeled by x in $\{x_i\}$, n_y^Y is the size of the community labeled by y in $\{y_i\}$, and n_{xy} is their overlap [12]. Then $I(X, Y)$ can be calculated by

$$I(X, Y) = \sum_x \sum_y P(x, y) \log \frac{P(x, y)}{P(x)P(y)} \tag{13}$$

To avoid the influence brought by the absolute value of $I(X, Y)$, it is normalized through

$$I_{norm}(X, Y) = \frac{2I(X, Y)}{H(X) + H(Y)} \tag{14}$$

where $H(X)$ and $H(Y)$ are the entropies of X and Y, respectively. Through the normalized mutual information I_{norm}, the performances of different community detection algorithms are effectively compared.

4.2 Algorithms in Comparison

Two algorithms in the following are for comparison with the proposed multi-section algorithm, which is denoted as PMS. The first one is the original recursive bisection algorithm (denoted as RBS) in [4], which recursively cuts the graph until the modularity is no longer increased. The other method is the Markov Cluster algorithm (denoted as MCL) proposed in papers [3, 13, 14], which simulates a peculiar diffusion process of expansions and inflations on the graph. The time complexity of the MCL is $O(K^2n)$, which is essentially a linear algorithm. Note that, the time complexity of RBS is $O(n^2 \log n)$, and the time complexity of PMS is $O(K^4n^2)$. Compared to some other algorithms [15–17], these three algorithms are not excellent for the partition accuracy, as a tradeoff to pursue a lower time complexity.

4.3 Evaluation Results

The simulation results are shown in Fig. 2, where γ and β respectively describe the power-law distribution exponents of the node degree and the community size in the LFR benchmarks. We set $2 \leq \gamma \leq 3$ and $2 \leq \beta \leq 3$, which increase the difficulty of community detection. Meanwhile, the real world power-law exponents are mostly reported to be in the range of $[2, 3]$. It can be seen that PMS outperforms the other algorithms for most values of μ. The normalized mutual information of the MCL is quite high when μ is small, however, it decreases quickly with increased μ. Similar results on the MCL are observed in the paper [12]. Although PMS and RBS are better for larger μ, they cannot achieve 100% correct partition, even if μ is small enough, due to the information loss resulting from the relaxation. PMS outperforms RBS over the entire range of μ, since PMS is based on the "global" view of the partition, and RBS is based on the "local" view. All three algorithms become ineffective, when the mixing parameter μ is larger than 0.5. The normalized mutual information obtained by MCL, on the whole, starts with high value and decreases quickly with the increased μ. Similarly, the normalized mutual information obtained by PMS and RBS, on the whole, starts with a lower value than MCL, but decreases more slowly with the increased μ.

In terms of the normalized mutual information, PMS is still not good enough, since many algorithms, such as [15], are still able to obtain 0.9 normalized mutual information when μ is larger than 0.3. However, PMS is essentially an algorithm with a time complexity of $O(K^4n^2)$, which is qualified enough if considering its execution time. PMS and RBS are approximation algorithms for the optimal partitioning, which only loses information in the relaxation method. A promising method for the further improvement of PMS and RBS is to pursue a better relaxation with lower time complexity, which can keep more information of s_i and $\bar{s}_i$, as is done in [6].

5. CONCLUSION

In this paper, we propose a spectral graph multisection algorithm through the orthogonality of Hadamard matrix, which overcomes the flaw of the traditional spectral modularity optimization algorithm. Instead of doing recursive modularity bisection, the proposed algorithm directly cuts the graph into multisections. The basic idea of the proposed algorithm is to extend the modularity matrix to a higher order, and to use carefully designed vectors to represent the group assignments of the nodes for the relaxation. The proposed algorithm can be viewed as an extension of the bisection algorithm, which also succeeds the properties of the bisection algorithm. In sparse graphs, the time complexity of the proposed algorithm is $O(K^4n^2)$, which is low among all the community detection algorithms. Finally, the simulations show that the proposed algorithm achieves good performances in the LFR benchmarks of different settings.

References

[1] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75 – 174, 2010.

[2] R. Guimera and L. A. Nunes Amaral, "Functional cartography of complex metabolic networks," *Nature*, vol. 433, no. 7028, pp. 895–900, 2005.

[3] S. Dongen, "Performance criteria for graph clustering and markov cluster experiments," Amsterdam, The Netherlands, The Netherlands, Tech. Rep., 2000.

[4] M. E. J. Newman, "Modularity and community structure in networks," *PNAS*, vol. 103, no. 23, pp. 8577–8582, Jun. 2006.

[5] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, p. 066111, 2004.

[6] S. White and P. Smyth, "A spectral clustering approach to finding communities in graphs," in *Proc. of SDM 2005*, pp. 76–84.

[7] J. Ruan and W. Zhang, "An efficient spectral algorithm for network community discovery and its applications to biological and social networks," in *Proc. of ICDM 2007*, pp. 643–648.

[8] M. Newman, *Networks: An Introduction.* New York, NY, USA: Oxford University Press, Inc., 2010.

[9] P. Sanders and C. Schulz, "Engineering multilevel graph partitioning algorithms," in *Proc. of ESA 2011*, pp. 469–480.

[10] J. A. Tropp, "Improved analysis of the subsampled randomized hadamard transform." *Advances in Adaptive Data Analysis*, vol. 3, no. 1-2, pp. 115–126, 2011.

[11] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E*, vol. 78, no. 4, 2008.

[12] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis," *Physical Review E*, vol. 80, p. 056117, 2009.

[13] U. Brandes, M. Gaertler, and D. Wagner, "Experiments on graph clustering algorithms," in *Proc. of ESA 2003*, pp. 568–579.

[14] V. Satuluri, S. Parthasarathy, and D. Ucar, "Markov clustering of protein interaction networks with improved balance and scalability," in *Proc. of ACM BCB 2010*, pp. 247–256.

[15] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *PNAS*, vol. 105, no. 4, pp. 1118–1123, 2008.

[16] A. Lancichinetti and S. Fortunato, "Consensus clustering in complex networks," *Scientific Reports*, vol. 2, Mar. 2012.

[17] A. McDaid and N. Hurley, "Detecting highly overlapping communities with model-based overlapping seed expansion," in *Proc. of ASONAM 2010*, pp. 112–119.