A SPARSE COMPLETELY POSITIVE RELAXATION OF THE MODULARITY MAXIMIZATION FOR COMMUNITY DETECTION

JUNYU ZHANG *, HAOYANG LIU †, ZAIWEN WEN ‡, AND SHUZHONG ZHANG §

Abstract. In this paper, we consider the community detection problem under either the stochastic block model (SBM) assumption or the degree-correlated stochastic block model (DCSBM) assumption. The modularity maximization formulation for the community detection problem is NP-hard in general. In this paper, we propose a sparse and low-rank completely positive relaxation for the modularity maximization problem, we then develop an efficient row-by-row (RBR) type block coordinate descent (BCD) algorithm to solve the relaxation and prove an $\mathcal{O}(1/\sqrt{N})$ convergence rate to a stationary point where N is the number of iterations. A fast rounding scheme is constructed to retrieve the community structure from the solution. Non-asymptotic high probability bounds on the misclassification rate are established to justify our approach. We further develop an asynchronous parallel RBR algorithm to speed up the convergence. Extensive numerical experiments on both synthetic and real world networks show that the proposed approach enjoys advantages in both clustering accuracy and numerical efficiency. Our numerical results indicate that the newly proposed method is a quite competitive alternative for community detection on sparse networks with over 50 million nodes.

Key words. Community Detection, Degree-Correlated Stochastic Block Model, Completely Positive Relaxation, Proximal Block Coordinate Descent Method, Non-asymptotic Error Bound.

AMS subject classifications. 90C10, 90C26, 90C30

1. Introduction.

The community detection problem aims to retrieve the underlying community/cluster structure of a network from the observed nodes connection data. This network structure inference technique has been reinvigorated because of the modern applications of large scale networks, including social networks, energy distribution networks and economic networks, etc. The standard frameworks for studying community detection in networks are the stochastic block model (SBM) [14] and the degree-correlated stochastic block model (DCSBM) [17, 8]. Both of them are random graph models based on an underlying disjoint community structure. To solve the community detection problem, there have been a variety of greedy methods which aim to maximize some quality function of the community structure; see [33] and the references therein. Although these methods are numerically efficient, their theoretical analysis is still largely missing.

An important class of methods for community detection are the so-called spectral clustering methods, which exploit either the spectral decomposition of the adjacency matrix [21, 30, 16], or the normalized adjacency or graph Laplacian matrix [8, 7, 13, 29, 3, 27]. A significant advantage of the spectral clustering methods is that they are numerically efficient and scalable. Although many of them are shown to be statistically consistent with the SBM and the DCSBM conditions under certain conditions, their numerical performance on synthetic and real data does not fully support such hypothesis. This can also be observed in the numerical experiments where the spectral methods SCORE [16] and OCCAM [35] are tested. It is worth mentioning that spectral clustering methods may fail to detect the communities for large sparse networks.

Another closely connected class of methods are based on the nonnegative matrix factorization (NMF) methods. In [10], an interesting equivalence is shown between some specific NMF-based methods, the kernel K-means method and a specific spectral clustering method.

*Department of Industrial and System Engineering, University of Minnesota (zhan4393@umn.edu).

†Beijing International Center for Mathematical Research, Peking University (liuhaoyang@pku.edu.cn).

‡Beijing International Center for Mathematical Research, Peking University (wenzw@pku.edu.cn). Research supported in part by the NSFC grant 11421101, and by the National Basic Research Project under the grant 2015CB856002.

§Department of Industrial and System Engineering, University of Minnesota (zhangs@umn.edu).

There have been extensive research on graph clustering and community detection problems; see e.g. [11, 18, 32, 26, 19, 34]. The clustering performance and scalability of these methods are well appreciated, while their theoretical guarantees are still not fully understood. In [25], the authors address this issue by showing the asymptotic consistency of their NMF-based community detection algorithm under the SBM or the DCSBM. However, a non-asymptotic error bound is still unknown.

In addition to the nonconvex methods, significant advances have also been made by means of convex relaxation. In [23, 5, 6], the authors propose to decompose the adjacency matrix or its variants into a "low-rank + sparse" form by using nuclear norm minimization methods. In [4], an SDP relaxation of the modularity maximization followed by a doubly weighted k-median clustering is proposed. This method is demonstrated to be very competitive with nice theoretical guarantees under the SBM or the DCSBM against the state-of-the-art methods in various datasets; see [2, 12] for more examples. Although these convex methods are numerically stable and statistically efficient under various model settings, they are algorithmically not scalable. This is because solving these problems involves expensive subroutines such as full eigenvalue decomposition or matrix inversion, making these algorithms unsuitable for huge datasets.

In this paper, a sparse and low-rank completely positive relaxation for the modularity maximization problem [22] is proposed. Specifically, we define an assignment matrix Φ for k communities among n nodes as an $n \times k$ binary matrix such that each row of Φ corresponds to a particular node in the network and contains exactly one element equal to one. The column index of each element indicates which community this node belongs to. Consequently, we reformulate the modularity maximization as an optimization over the set of low-rank assignment matrices. Then, we relax the binary constraints for each row of the assignment matrix and impose the nonnegative and spherical constraints. The cardinality constraints can be optionally added in order to handle large scale datasets. To solve the resulting relaxation, we regard each row of the relaxed assignment matrix as a block-variable and propose a row-by-row (RBR) proximal block coordinate descent approach. We show that a closed-form solution for these nonconvex subproblems is available and an $\mathcal{O}(1/\sqrt{N})$ convergence rate to a stationary point can be obtained. One can then proceed with either K-means or K-median clustering to retrieve the community structure from the solution of the relaxed modularity maximization problem. Since these clustering schemes themselves are expensive, a simple yet efficient rounding scheme is constructed for the community retrieval purpose. Non-asymptotic high-probability bounds are established for the misclassification rate under the DCSBM. The properites of the SBM can be obtained as a special case of the DCSBM. We further propose an asynchronous parallel computing scheme as well as an iterative rounding scheme to enhance the efficiency of our algorithm. Finally, numerical experiments on both real world and synthetic networks show that our method is competitive with the SCORE [16], OCCAM [35] and CMM [4] algorithms regarding both numerical efficiency and clustering accuracy. We also perform experiments on extremely large sparse networks with up to 50 million nodes and compare the proposed method with the LOUVAIN algorithm [1]. It turns out that our method is at least comparable to the LOUVAIN algorithm in terms of finding a community assignment with a small number of communities and a high modularity.

The rest of the paper is organized as follows. In Section 2, we set up the community detection problem and introduce the proposed framework. In Section 3, we present our algorithm and the convergence results as well as the asynchronous parallel scheme. In Section 4, we validate the statistical consistency of our methods by non-asymptotically bounding the misclassification rate under the DCSBM. Finally, the numerical experiments and comparisons are presented in Section 5.

2. Model Descriptions.

2.1. The Degree-Correlated Stochastic Block Model. In [14], Holland et al. proposed the so called stochastic block model (SBM), where a non-overlapping true community structure is assumed. This structure is a partition $\{C_1^*, ..., C_k^*\}$ of the nodes set $[n] = \{1, ..., n\}$, that is

$$C_a^* \cap C_b^* = \emptyset, \forall a \neq b \text{ and } \cup_{a=1}^k C_a^* = [n].$$

Under the SBM, the network is generated as a random graph characterized by a symmetric matrix $B \in S^{k\times k}$ with all entries ranging between 0 and 1. Let $A \in \{0,1\}^{n\times n}$ be the adjacency matrix of the network with $A_{ii} = 0, \forall i \in [n]$. Then for $i \in C_a^*, j \in C_b^*, i \neq j$,

$$A_{ij} = \begin{cases} 1, & \text{with probability } B_{ab}, \\ 0, & \text{with probability } 1 - B_{ab}. \end{cases}$$

One significant drawback of the SBM is that it oversimplifies the network structure. All nodes in a same community are homogeneously characterized. The DCSBM is then proposed to alleviate this drawback by introducing the *degree heterogeneity* parameters $\theta = (\theta_1, ..., \theta_n)$ for each node; see [8, 17]. If one keeps the definitions of B and $C_a^*, a = 1, ..., k$ and still let A be the adjacency matrix of the network with $A_{ii} = 0, \forall i \in [n]$, then for $i \in C_a^*, j \in C_b^*, i \neq j$,

$$A_{ij} = \begin{cases} 1, & \text{with probability } B_{ab}\theta_i\theta_j, \\ 0, & \text{with probability } 1 - B_{ab}\theta_i\theta_j. \end{cases}$$

Hence, the heterogeneity of nodes is characterized by this vector θ. Potentially, a node i with larger θ_i has more edges linking to the other nodes.

2.2. A Sparse and Low-Rank Completely Positive Relaxation. For the community detection problem, one popular method is to maximize a community quality function called "modularity" proposed in [22]. We say that a binary matrix X is a partition matrix, if there exists a partition $\{S_1, ..., S_r\}, r \geq 2$ of the index/node set $[n]$ such that $X_{ij} = 1$ for $i, j \in S_t, t \in \{1, ..., r\}$ and $X_{ij} = 0$ otherwise. Then a given community structure of a network, not necessarily the true underlying structure, is fully characterized by its associated partition matrix. We denote the degree vector by d where $d_i = \sum_j A_{ij}, i \in [n]$. Define the matrix

$$C = -(A - \lambda dd^\top), \tag{1}$$

where $\lambda = 1/\|d\|_1$. Therefore the modularity of a community structure is defined as $Q = -\langle C, X\rangle$. Let the set of all partition matrices of n nodes with *no more than* r subsets be denoted by $\mathcal{P}_n^r$. Then the modularity maximization model is formulated as

$$\min_X \langle C, X\rangle \quad \text{s.t.} \quad X \in \mathcal{P}_n^n. \tag{2}$$

Assume that we know the number of communities in the network is k. Then the problem becomes

$$\min_X \langle C, X\rangle \quad \text{s.t.} \quad X \in \mathcal{P}_n^k. \tag{3}$$

Although modularity optimization enjoys many desirable properties, it is an NP-hard problem. A natural relaxation is the convex SDP relaxation considered in [4]:

$$\begin{aligned} \min_{X\in\mathbb{R}^{n\times n}} \quad & \langle C, X\rangle \\ \text{s.t.} \quad & X_{ii} = 1, i = 1, \ldots, n, \\ & 0 \leq X_{ij} \leq 1, \forall i, j, \\ & X \succeq 0. \end{aligned} \tag{4}$$

This formulation exhibits desirable theoretical properties under the SBM or the DCSBM. However, the computational cost is high when the network is large. This paper aims to formulating a model that retains nice theoretical properties while maintaining computationally viable. Recall the definition of an assignment matrix in Section 1 and suppose that the number of communities k is no more than r. Then the true partition matrix X^* can be decomposed as $X^* = \Phi^*(\Phi^*)^\top$, where $\Phi^* \in \{0,1\}^{n\times r}$ is the true assignment matrix. Let us define the set of assignment matrix in $\mathbb{R}^{n\times r}$ to be $\mathcal{A}^{n\times r}$, and define the notations

$$U = (u_1, ..., u_n)^\top, \quad \Phi = (\phi_1, ..., \phi_n)^\top,$$

where the u_i's and ϕ_i's are r-by-1 column vectors. By setting $r = n$ and $r = k$, respectively, problem (2) and (3) can be formulated as

$$\min_{\Phi} \quad \left\langle C, \Phi\Phi^\top \right\rangle \quad \text{s.t. } \Phi \in \mathcal{A}^{n\times r}. \tag{5}$$

A matrix $X \in \mathbb{R}^{n\times n}$ is completely positive (CP) if there exists $U \in \mathbb{R}^{n\times r}, r \geq 1$ such that $U \geq 0$, $X = UU^\top$. Let the set of all $n \times n$ CP matrices be denoted by $\mathcal{CP}_n$. Due to the reformulation (5), by constraining the $\mathcal{CP}$ rank of X, we naturally obtain a rank-constrained $\mathcal{CP}$ relaxation of problem (2) and (3) as

$$\begin{aligned} \min_{X} \quad & \langle C, X\rangle \\ \text{s.t.} \quad & X_{ii} = 1, i = 1, \ldots, n, \\ & X \in \mathcal{CP}_n, \\ & \text{rank}_{\mathcal{CP}}(X) \leq r, \end{aligned} \tag{6}$$

where $\text{rank}_{\mathcal{CP}}(X) := \min\{s : X = UU^\top, U \in \mathbb{R}^{n\times s}_+\}$. Note that the above problem is still NP-hard. By using a decompsition $X = UU^\top$, we propose an alternative formulation:

$$\begin{aligned} \min_{U\in\mathbb{R}^{n\times r}} \quad & \left\langle C, UU^\top \right\rangle \\ \text{s.t.} \quad & \|u_i\|^2 = 1, i = 1, \ldots, n, \\ & \|u_i\|_0 \leq p, i = 1, \ldots, n, \\ & U \geq 0, \end{aligned} \tag{7}$$

where the parameter λ in C can be set to a different value other than $1/\|d\|_1$ and $1 \leq p \leq r$ is an integer. Note that the additional ℓ_0 constraint is added to each row of U to impose sparsity of the solution. When $p = r$, that is, the ℓ_0 constraints vanish, this problem is exactly equivalent to the $\mathcal{CP}$ relaxation (6). Usually, the value of p is set to a small number such that the total amount of the storage of U is affordable on huge scale datasets. Even though (7) is still NP-hard, the decomposition and the structure of the problem enable us to develop a computationally efficient method to reach at least a stationary point.

We summarize the relationships between different formulations as

$$\begin{array}{c} (3) \\ \Updownarrow \\ (5) \end{array} \quad \Rightarrow \ (7) \ \Rightarrow (6) \ \Rightarrow \ (4),$$

where formulations (3) and (5) are equivalent forms of the NP-hard modularity maximization and problem (4) is the SDP relaxation. The "$\Rightarrow$" indicates that our relaxation is tighter than the convex SDP relaxation proposed in [4]. If U is feasible to our problem (7), then $UU^\top$ is automatically feasible to the SDP relaxation (4).

2.3. A Community Detection Framework. Suppose that a solution U is computed from problem (7), we can then use one of the following three possible ways to retrieve the community structure from U.

i. Apply the K-means clustering algorithm directly to the rows of U by solving

$$\begin{aligned} \min \quad & \|\Phi X_c - U^*\|_F^2 \\ \text{s.t.} \quad & \Phi \in \mathcal{A}^{n\times k}, X_c \in \mathbb{R}^{k\times k}, \end{aligned} \tag{8}$$

where the variable Φ contains the community/cluster assignment and X_c contains the corresponding centers.

ii. Apply the weighted K-means clustering over the rows of U, namely, we solve,

$$\begin{aligned} \min \quad & \|D(\Phi X_c - U)\|_F^2 \\ \text{s.t.} \quad & \Phi \in \mathcal{A}^{n\times k}, X_c \in \mathbb{R}^{k\times k}, \end{aligned} \tag{9}$$

where $D = \mathrm{diag}\{d_1, ..., d_n\}$, Φ and X_c are interpreted the same as that in (i).

iii. Apply a direct rounding scheme to generate the final community assignment Φ. Let $\phi_i^\top$ be the ith row of Φ,

$$\phi_i \leftarrow e_{j_0},\ j_0 = \arg\max_j (u_i)_j, \tag{10}$$

where e_{j_0} is the j_0th unit vector in $\mathbb{R}^k$.

All three methods have theoretical guarantees under the SBM or the DCSBM as will be shown in later sections. The overall numerical framework in this paper is as follows:

Step 1. Solve problem (7) starting from a randomly generated initial point and obtain an approximate solution.

Step 2. Recover the community assignment via one of the three clustering methods.

Step 3. Repeat steps 1 and 2 for a couple of times and then pick up a community assignment with the highest modularity.

3. An Asynchrounous Proximal RBR Algorithm.

3.1. A Nonconvex Proximal RBR Algorithm. In this subsection, we setup a basic proximal RBR algorithm without parallelization for (7). Let r be chosen as the community number k. Define the feasible set for the block variable u_i to be $\mathcal{U}_i = \mathcal{S}^{k-1} \cap \mathbb{R}^k_+ \cap M_p^k$, where $\mathcal{S}^{k-1}$ is the $k-1$ dimensional sphere and $M_p^k = \{x \in \mathbb{R}^k : \|x\|_0 \le p\}$. Define $\mathcal{U} = \mathcal{U}_1 \times \cdots \times \mathcal{U}_n$. Then problem (7) can be rewritten as

$$\min_{U\in\mathcal{U}} \quad f(U) \equiv \left\langle C, UU^\top \right\rangle. \tag{11}$$

For the ith subproblem, we fix all except the ith row of U and solve the subproblem

$$u_i = \arg\min_{x\in\mathcal{U}_i} f(u_1, ..., u_{i-1}, x, u_{i+1}, ..., u_n) + \frac{\sigma}{2}\|x - \bar{u}_i\|^2,$$

where $\bar{u}_i$ is some reference point and $\sigma > 0$. Note that the spherical constraint eliminates the second order term $\|x\|^2$ and simplifies the subproblem to

$$u_i = \arg\min_{x\in\mathcal{U}_i} b^\top x, \tag{12}$$

where $b = 2C^i_{-i}U_{-i} - \sigma\bar{u}_i$, and C^i_{-i} is the ith row of C without the ith component, U_{-i} is the matrix U without the ith row $u_i^\top$. For any positive integer p, a closed-form solution can be derived in the next lemma.

LEMMA 1. *For problem* $u = \arg\min\{b^\top x : x \in \mathcal{S}^{k-1} \cap \mathbb{R}^k_+ \cap M^k_p\}$, *define* $b^+ = \max\{b, 0\}, b^- = \max\{-b, 0\}$, *where the* max *is taken component-wisely. Then the closed-form solution is given by*

$$u = \begin{cases} \frac{b^-_p}{\|b^-_p\|}, & \text{if } b^- \neq 0, \\ e_{j_0}, \text{ with } j_0 = \arg\min_j b_j, & \text{otherwise}, \end{cases} \tag{13}$$

where b^-_p *is obtained by keeping the top* p *components in* b^- *and letting the others be zero, and when* $\|b^-\|_0 \leq p$, $b^-_p = b^-$.

Proof. When $b^- \neq 0$, for $\forall u \in \mathbb{R}^k_+, \|u\|^2 = 1, \|u\|_0 \leq p$, we obtain

$$b^\top u = (b^+ - b^-)^\top u \geq -(b^-)^\top u \geq -\|b^-_p\|,$$

where the equality is achieved at $u = \frac{b^-_p}{\|b^-_p\|}$.

When $b^- = 0$, i.e., $b \geq 0$, we have

$$b^\top u = \sum_i b_i u_i \geq \sum_i b_i u_i^2 \geq b_{\min} \sum_i u_i^2 = b_{\min},$$

where $b_{\min} = \min_j b_j$, and the equality is achieved at $u = e_{j_0}, j_0 = \arg\min_j b_j$. ∎

Our proximal RBR algorithm is outlined in Algorithm 1.

Algorithm 1: A Row-By-Row (RBR) Algorithm

Give $U^0, \sigma > 0$. Set $t = 0$.
while *Not converging* **do**
  for *i = 1,...,n* **do**
    $u_i^{t+1} = \arg\min_{x \in \mathcal{U}_i} f(u_1^{t+1}, ..., u_{i-1}^{t+1}, x, u_{i+1}^t, ..., u_n^t) + \frac{\sigma}{2}\|x - u_i^t\|^2$.
  $t \leftarrow t + 1$.

Note that the objective function is smooth with a Lipschitz continuous gradient. When $p = k$, the ℓ_0 constraints hold trivially and thus can be removed. Then problem (7) becomes a smooth nonlinear program. Therefore, the KKT condition can be written as

$$\begin{cases} 2CU^* + 2\Lambda U^* - V = 0, & \\ \|u_i^*\| = 1,\ u_i^* \geq 0,\ v_i \geq 0, & \text{for } i = 1, ..., n, \\ (u_i^*)_j (v_i)_j = 0, & \text{for } i = 1, ..., n \text{ and } j = 1, ..., k, \end{cases}$$

where $\Lambda = \mathrm{diag}\{\lambda\}, V = (v_1, ..., v_n)^\top$ and $\lambda_i \in \mathbb{R}, v_i \in \mathbb{R}^k_+$ are the Lagrange multipliers associated with the constraints $\|u_i\|^2 = 1, u_i \geq 0$, respectively. Then we have the following convergence result.

THEOREM 2. *Suppose that the sequence* $\{U^0, U^1, ..., U^N\}$ *is generated by Algorithm 1 with a chosen proximal parameter* $\sigma > 0$ *and* $p = k$. *Let*

$$t^* := \arg\min_{1 \leq t \leq N} \left\{ \|U^{t-1} - U^t\|_F^2 \right\}. \tag{14}$$

Then there exist Lagrange multipliers $\lambda_i \in \mathbb{R}, v_i \in \mathbb{R}^k_+, i = 1, ..., n$, *such that*

$$\begin{cases} \left\|2CU^{t^*} + 2\Lambda U^{t^*} - V\right\|_F \leq 2\sqrt{\frac{2\|C\|_1(4\|C\|_F^2 + \sigma^2)}{N\sigma}}, & \\ \|u_i^{t^*}\| = 1,\ u_i^{t^*} \geq 0,\ v_i \geq 0, & \text{for } i = 1, ..., n, \\ (u_i^{t^*})_j (v_i)_j = 0, & \text{for } i = 1, ..., n \text{ and } j = 1, ..., k, \end{cases}$$

where $\|C\|_1$ is the component-wise ℓ_1 norm of C.

Proof. The proof consists mainly of two steps. First, we show that $\sum_{t=1}^{N} \|U^t - U^{t-1}\|_F^2$ is moderately bounded. By the optimality of the subproblems, we have that for $i = 1, ..., n$,

$$f(u_1^t, ..., u_{i-1}^t, u_i^{t-1}, ..., u_n^{t-1}) - f(u_1^t, ..., u_i^t, u_{i+1}^{t-1}, ..., u_n^{t-1}) \geq \frac{\sigma}{2}\|u_i^t - u_i^{t-1}\|^2.$$

Summing these inequalities up over i gives

$$f(U^{t-1}) - f(U^t) \geq \frac{\sigma}{2}\|U^t - U^{t-1}\|_F^2,$$

which yields

$$\sum_{t=1}^{N} \|U^t - U^{t-1}\|_F^2 \leq \frac{2}{\sigma}\left(f(U^0) - f(U^N)\right). \tag{15}$$

Note that $|\langle u_i, u_j\rangle| \leq \|u_i\|\|u_j\| = 1$, and so for any feasible U,

$$f(U) = \langle CU, U\rangle = \sum_{i,j} C_{i,j}\langle u_i, u_j\rangle \in [-\|C\|_1, \|C\|_1].$$

Combining with (15) and the definition of t^* in (14), it holds that

$$\|U^{t^*} - U^{t^*-1}\|_F^2 \leq \frac{4}{N\sigma}\|C\|_1. \tag{16}$$

Second, also by the optimality of the subproblems, there exist KKT multipliers $\lambda_i \in \mathbb{R}$ and $v_i \in \mathbb{R}_+^k$ for each subproblem such that,

$$\begin{cases} \nabla_i f(u_1^{t^*}, ..., u_i^{t^*}, u_{i+1}^{t^*-1}, ..., u_n^{t^*-1}) + \sigma(u_i^{t^*} - u_i^{t^*-1}) + 2\lambda_i u_i^{t^*} - v_i = 0, \\ \|u_i^{t^*}\|^2 = 1, u_i^{t^*} \geq 0, v_i \geq 0, \\ (u_i^{t^*})_j (v_i)_j = 0, \text{ for } j = 1, ..., k. \end{cases} \tag{17}$$

Define $\tilde{U}^{t^*,i} = \left[u_1^{t^*}, ...u_i^{t^*}, u_{i+1}^{t^*-1}, ..., u_n^{t^*-1}\right]^\top$, then

$$\begin{aligned} & \|\nabla_i f(U^{t^*}) - \nabla_i f(u_1^{t^*}, ..., u_i^{t^*}, u_{i+1}^{t^*-1}, ..., u_n^{t^*-1})\|^2 \\ &= \|2C_i\left(U^{t^*} - \tilde{U}^{t^*,i}\right)\|^2 \\ &\leq 4\|C_i\|^2\|U^{t^*} - \tilde{U}^{t^*,i}\|_F^2 \\ &\leq 4\|C_i\|^2\|U^{t^*} - U^{t^*-1}\|_F^2, \end{aligned} \tag{18}$$

where C_i is the ith row of matrix C. Consequently,

$$\begin{aligned} & \left\|2CU^{t^*} + 2\Lambda U^{t^*} - V\right\|_F^2 \\ &= \sum_{i=1}^{n} \left\|\nabla_i f(U^{t^*}) + 2\lambda_i u_i^{t^*} - v_i\right\|^2 \\ &= \sum_{i=1}^{n} \left\|\nabla_i f(U^{t^*}) - \nabla_i f(u_1^{t^*}, ..., u_i^{t^*}, u_{i+1}^{t^*-1}, ..., u_n^{t^*-1}) - \sigma(u_i^{t^*} - u_i^{t^*-1})\right\|^2 \end{aligned}$$

$$
\begin{aligned}
(19) \quad &\leq 2\sum_{i=1}^{n}(\|\nabla_i f(U^{t^*}) - \nabla_i f(u_1^{t_1^*}, ..., u_i^{t_i^*}, u_{i+1}^{t_{i+1}^*-1}, ..., u_n^{t_n^*-1})\|^2 + \sigma^2\|u_i^{t_i^*} - u_i^{t_i^*-1}\|^2) \\
&\leq 2(4\|C\|_F^2 + \sigma^2)\|U^{t^*} - U^{t^*-1}\|_F^2 \\
&\leq \frac{8\|C\|_1(4\|C\|_F^2 + \sigma^2)}{T\sigma},
\end{aligned}
$$

where the second equality is due to (17) and the last inequality is due to (16). Combining the bounds in (19) and (17) proves the theorem. ∎

3.2. An Asynchronous Parallel Proximal RBR Scheme. In this subsection, we briefly discuss the parallelization of the RBR Algorithm 1. The overall parallel setup for the algorithm is a shared memory model with many threads. The variable U is in the shared memory so that it can be accessed by all threads. The memory locking is not imposed throughout the process. Namely, even when a thread is updating some row u_i of U, the other threads can still access U whenever needed.

Before explaining the parallel implementation, let us first make clear the sequential case. The main computational cost of updating one row u_i by solving the subproblem (12) is due to the computation of $b = 2C^i_{-i}U_{-i} - \sigma\bar{u}_i$, where $\bar{u}_i$ and U are the current iterates. Note that the definition of C in (1) yields

$$
b^\top = -2A^i_{-i}U_{-i} + 2\lambda d_i d^\top_{-i}U_{-i} - \sigma\bar{u}_i, \tag{20}
$$

where A^i_{-i} is the ith row of A without the ith component. By the sparsity of A, computing $-A^i_{-i}U_{-i}$ takes $\mathcal{O}(d_i p)$ flops. As for the term $\lambda d_i d^\top_{-i}U_{-i}$, we can compute the vector $d^\top U = \sum_{i=1}^{|V|} d_i u_i^\top$ once, then store it throughout all iterations. For each time after a row u_i is updated, we can evaluate this vector at a cost of $\mathcal{O}(p)$ flops. Hence, the total computational cost of one full sweep of all rows is $\mathcal{O}(2(k+p)|V| + p|E|)$.

Our parallel implementation is outlined in Algorithm 2 where many threads working at the same time. The vector $d^\top U$ and matrix U are stored in the *shared* memory and they can be accessed and updated by all threads. Each thread picks up one row u_i at a time and then it reads U and the vector $d^\top U$. Then an individual copy of the vector $b^\top$ is calculated, i.e., $b^\top$ is owned *privately* and cannot be accessed by other threads. Thereafter, the variable u_i is updated and $d^\top U$ is set as $d^\top U \leftarrow d^\top U + d_i(u_i - \bar{u}_i)$ in the shared memory. Immediately without waiting for other threads to finish their computation, it proceeds to another row. Hence, unlike the situation in Algorithm 1, other blocks of variables $u_j, j \neq i$ are not necessarily up to date when a thread is updating a row u_i and $d^\top U$. Moreover, if another thread is just modifying some row u_j or the vector $d^\top U$ when this thread is reading these memories, it will make the update of u_i with the partially updated data. A possible way to avoid the partially updated conflict is to add memory locking when a thread is changing the memories of U and $d^\top U$. In this way, other threads will not be able to access these memories and will wait until the update completes. However, this process may cause many cores to be idle so that it only provides limited speedups or even slows the algorithm down. In our implementation, these memory locking are completely removed. Consequently, our method may be able to provide near-linear speedups.

In order to increase the chance of finding a good solution, we perform the simple rounding scheme (10) in Section 2.3 whenever it is necessary. By applying this technique, we obtain a binary solution after each iteration as a valid community partition. Although the convergence is not guaranteed in this case, it is numerically robust throughout our experiments. Note that the rounding technique can also be asynchronously parallelized because the update of all rows are completely independent.

It is worth noting that our algorithm is similar to the HOGWILD! algorithm [28], where HOGWILD! minimizes the objective function in a stochastic gradient descend style while we adopt the exact block coordinate minimization style. There are no conflicts in our method when updating the variable U compared to the HOGWILD!, since each row can only be modified by up to one thread. However, the issue of conflicts arises when updating the intermediate vector $d^\top U$. Another similar asynchronous parallel framework is CYCLADES [24]. Consider a "variable conflict graph", of which the nodes are grouped as blocks of variables and an edge between two blocks of variables exists if they are coupled in the objective function. The CYCLADES carefully utilizes the sparsity of the variable conflict graph and designs a particular parallel updating order so that no conflict emerges due to the asynchronism with high probability. This enables the algorithm to be asynchronously implemented while keeping the sequence of iterates equivalent to that generated by a sequential algorithm. However, because the block variables are all coupled together in formulation (7), no variable sparsity can be exploited in our case.

Algorithm 2: Asynchronous parallel RBR algorithm

Give U^0, set $t = 0$
while *Not converging* **do**
  for *each row i asynchronously* **do**
    Compute the vector $b_i^\top = -2A^i_{-i}U_{-i} + 2\lambda d_i d_{-i}^\top U_{-i} - \sigma u_i$, and save previous iterate $\bar{u}_i$ in the private memory.
    Update $u_i \leftarrow \arg\min_{x \in \mathcal{U}_i} b_i^\top x$ in the shared memory.
    Update the vector $d^\top U \leftarrow d^\top U + d_i(u_i - \bar{u}_i)$ in the shared memory.
  if *rounding is activated* **then**
    for *each row i asynchronously* **do**
      Set $u_i = e_{j_0}$ where $j_0 = \arg\max(u_i)_j$.
    Compute and update $d^\top U$.

4. Theoretical Error Bounds Under The DCSBM. In this section, we establish theoretical results of our method. Throughout the discussion, we focus mainly on the DCSBM, as these assumptions and results for the SBM can be easily derived from those of the DCSBM as special cases. Define

$$G_a = \sum_{i \in C_a^*} \theta_i,\ J_a = \sum_{i \in C_a^*} \theta_i^2,\ M_a = \sum_{i \in C_a^*} \theta_i^3,\ n_a = |C_a^*|,$$

$$H_a = \sum_{b=1}^{k} B_{ab} G_b, \text{ and } f_i = H_a \theta_i, \text{ given } i \in C_a^*.$$

A direct computation gives $\mathbb{E}d_i = \theta_i H_a - \theta_i^2 B_{aa} \approx f_i$. For the ease of notation, we define $n_{\max} = \max_{1 \le a \le k} n_a$, and define $n_{\min}, H_{\max}, H_{\min}, J_{\max}, J_{\min}, M_{\max}, M_{\min}$ in a similar fashion. We adopt the *density gap* assumption in [4] to guarantee the model's identifiability.

ASSUMPTION 3. *(Chen, Li and Xu, 2016,[4]) Under the DCSBM, the density gap condition holds that*

$$\max_{1 \le a < b \le k} \frac{B_{ab}}{H_a H_b} < \min_{1 \le a \le k} \frac{B_{aa}}{H_a^2}.$$

Define $\|X\|_{1,\theta} = \sum_{ij} |X_{ij}| \theta_i \theta_j$ as the weighted ℓ_1 norm. Then we have the following lemma which bounds the difference between the approximate partition matrix $U^*(U^*)^\top$ and the true partition matrix $\Phi^*(\Phi^*)^\top$, where U^* is the optimal solution to problem (7).

LEMMA 4. *Suppose that Assumption 3 holds and the tuning parameter λ satisfies*

$$\max_{1\le a<b\le k} \frac{B_{ab}+\delta}{H_a H_b} < \lambda < \min_{1\le a\le k} \frac{B_{aa}-\delta}{H_a^2} \tag{21}$$

for some $\delta > 0$. Let U^* be the global optimal solution to problem (7), and define $\Delta = U^*(U^*)^\top - \Phi^*(\Phi^*)^\top$. Then with probability at least $0.99 - 2(e/2)^{-2n}$, we have

$$\|\Delta\|_{1,\theta} \le \frac{C_0}{\delta}\left(1 + \left(\max_{1\le a\le k} \frac{B_{aa}}{H_a^2}\|f\|_1\right)\right)(\sqrt{n\|f\|_1} + n),$$

where $C_0 > 0$ is some absolute constant that does not depend on problem scale and parameter selections.

Proof. Lemma 4 is proved directly from Theorem 1 in [4] for the convex SDP relaxation (4). Here we only show the main steps of the proof and how they can be translated from the original result of Chen, Li and Xu in [4]. By the optimality of U^* and the feasibility of Φ^* to problem (7), we have

$$0 \le \langle \Delta, A - \lambda dd^\top\rangle = \underbrace{\langle \Delta, \mathbb{E}A - \lambda ff^\top\rangle}_{S_1} + \underbrace{\lambda\langle \Delta, ff^\top - dd^\top\rangle}_{S_2} + \underbrace{\langle \Delta, A - \mathbb{E}A\rangle}_{S_3}. \tag{22}$$

The remaining task is to bound the three terms separately.

First, we bound the term S_1. Note that $U^*(U^*)^\top$ is feasible to the SDP problem (4), we still have for $\forall i, j \in C_a^*, a \in [k]$, $\Delta_{ij} \le 0$; for $\forall i \in C_a^*, \forall j \in C_b^*, a \ne b, a, b \in [k]$, $\Delta_{ij} \ge 0$. Combining with (21), the proof in [4] is still valid and yields

$$\Delta_{ij}(\mathbb{E}A - \lambda ff^\top)_{ij} \le -\delta\theta_i\theta_j|\Delta_{ij}|.$$

Hence we have

$$S_1 \le -\delta\|\Delta\|_{1,\theta}.$$

Similarly, the proof goes through for the bounds of S_2 and S_3. It can be shown that

$$S_2 \le C\left(\max_{1\le a\le k} \frac{B_{aa}}{H_a^2}\right)\|f\|_1\left(\sqrt{n\|f\|_1} + n\right)$$

holds with probability at least 0.99, where $C > 0$ is some absolute constant. Let $K_G \le 1.783$ be the constant for the original Grothendieck's inequality. Then

$$S_3 \le 2K_G\sqrt{8n\|f\|_1} + \frac{16K_G}{3}n$$

holds with probability at least $1 - 2(e/2)^{-2n}$. Combining the bounds for S_1, S_2, S_3 with the inequality (22) proves the theorem. ∎

Lemma 4 indicates that $U^*(U^*)^\top$ is close to $\Phi^*(\Phi^*)^\top$. Naturally, we want to bound the difference between the factors U^* and Φ^*. However, note that for any orthogonal matrix Q, $(U^*Q)(U^*Q)^\top = U^*(U^*)^\top$. This implies that in general U^* is not necessarily close to Φ^* unless multiplied by a proper orthogonal matrix. To establish this result, we use a matrix perturbation lemma in [15].

LEMMA 5. *(Corollary 6.3.8, in [15] on page 407) Let $A, E \in \mathbb{C}^{n\times n}$. Assume that A is Hermitian and $A + E$ is normal. Let $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ be the eigenvalues of A, and let*

$\hat{\lambda}_1, \hat{\lambda}_2, ..., \hat{\lambda}_n$ be the eigenvalues of $A + E$ arranged in order $Re(\hat{\lambda}_1) \leq Re(\hat{\lambda}_2) \leq \cdots \leq Re(\hat{\lambda}_n)$. Then

$$\sum_{i=1}^{n} \|\lambda_i - \hat{\lambda}_i\|^2 \leq \|E\|_F^2,$$

where $\|\cdot\|_F$ is the matrix Frobenius norm.

In the real symmetric case, this lemma states that we can bound the difference between every eigenvalue of A and perturbed matrix $A + E$ if the error term $\|E\|_F^2$ can be properly bounded.

THEOREM 6. *Let $\Theta = \mathrm{diag}\{\theta_1, ..., \theta_n\}$, and let U^*, Φ^*, Δ be defined according to Lemma 4. Then there exist orthogonal matrices Q_1^* and Q_2^* such that*

$$\|\Theta(U^*Q_1^* - \Phi^*)\|_F^2 \leq 2\sqrt{k}\frac{M_{\max}}{J_{\min}}\|\Delta\|_{1,\theta}^{\frac{1}{2}},$$

$$\|U^*Q_2^* - \Phi^*\|_F^2 \leq 2\sqrt{k}\frac{n_{\max}}{n_{\min}}\|\Delta\|_1^{\frac{1}{2}} \leq 2\sqrt{k}\frac{n_{\max}}{\theta_{\min}n_{\min}}\|\Delta\|_{1,\theta}^{\frac{1}{2}}.$$

Proof. (i). We first prove the bound for Q_1^*. A direct calculation yields

$$\begin{aligned} v(\theta) &:= \min_{Q^\top Q = I} \|\Theta(U^*Q - \Phi^*)\|_F^2 \\ &= \min_{Q^\top Q = I} \|\Theta U^* Q\|_F^2 + \|\Theta\Phi^*\|_F^2 - 2\left\langle \Theta U^* Q, \Theta\Phi^*\right\rangle \\ &= 2\sum_{i=1}^{n}\theta_i^2 - 2\max_{Q^\top Q = I}\left\langle Q, (U^*)^\top\Theta^2\Phi^*\right\rangle \\ &= 2\left(\sum_{i=1}^{n}\theta_i^2 - \|(U^*)^\top\Theta^2\Phi^*\|_*\right), \end{aligned} \tag{23}$$

where $\|\cdot\|_*$ denotes the nuclear norm of a matrix, and the last equality is due to the following argument. Note that

$$\begin{aligned} Q_1^* &= \arg\max_{Q^\top Q = I}\left\langle Q, (U^*)^\top\Theta^2\Phi^*\right\rangle \\ &= \arg\min_{Q^\top Q = I}\|Q - (U^*)^\top\Theta^2\Phi^*\|_F^2. \end{aligned}$$

Suppose that we have the singular value decomposition $(U^*)^\top\Theta^2\Phi^* = R\Sigma P^\top$, then $Q_1^* = RP^\top$ solves the above nearest orthogonal matrix problem. Hence,

$$\left\langle Q_1^*, (U^*)^\top\Theta^2\Phi^*\right\rangle = \mathrm{Tr}(\Sigma) = \|(U^*)^\top\Theta^2\Phi^*\|_*.$$

For any matrix Z, denote by $\sigma_i(Z)$ its ith singular value, and $\lambda_i(Z)$ its ith eigenvector. For a positive semidefinite matrix Z, $\sigma_i(Z) = \lambda_i(Z)$. Then

$$\begin{aligned} \sigma_i^2((U^*)^\top\Theta^2\Phi^*) &= \lambda_i((\Phi^*)^\top\Theta^2U^*(U^*)^\top\Theta^2\Phi^*) \\ &= \lambda_i((\Phi^*)^\top\Theta^2\Phi^*(\Phi^*)^\top\Theta^2\Phi^* + (\Phi^*)^\top\Theta^{1.5}\Delta_\Theta\Theta^{1.5}\Phi^*), \end{aligned} \tag{24}$$

where $\Delta_\Theta = \Theta^{\frac{1}{2}}(U^*(U^*)^\top - \Phi^*(\Phi^*)^\top)\Theta^{\frac{1}{2}} = \Theta^{\frac{1}{2}}\Delta\Theta^{\frac{1}{2}}$. The theorem is proved by the following three steps and by applying Lemma 5 to bound every $\sigma_i^2((U^*)^\top\Theta^2\Phi^*)$.

Step (a). We derive a tight estimation for every eigenvalue $\lambda_i((\Phi^*)^\top\Theta^2\Phi^*(\Phi^*)^\top\Theta^2\Phi^*)$, $i = 1, ..., k$. Let $v_a, 1 \le a \le k$, be the ath column of Φ^*. Then we have

$$v_a(t) = \begin{cases} 1, & t \in C_a^*, \\ 0, & t \notin C_a^*. \end{cases}$$

Therefore, we obtain

$$((\Phi^*)^\top\Theta^2\Phi^*)_{ab} = \sum_{t=1}^{n} \theta_t^2 v_a(t)v_b(t) = \begin{cases} 0, & a \neq b, \\ J_a, & a = b, \end{cases}$$

which yields

$$((\Phi^*)^\top\Theta^2\Phi^*)^2 = \mathrm{diag}\{J_1^2, ..., J_k^2\} \text{ and } \lambda_i\left(((\Phi^*)^\top\Theta^2\Phi^*)^2\right) = J_i^2.$$

Step (b). We bound the error term $\|(\Phi^*)^\top\Theta^{1.5}\Delta_\Theta\Theta^{1.5}\Phi^*\|_F^2$ for (24):

$$\|(\Phi^*)^\top\Theta^{1.5}\Delta_\Theta\Theta^{1.5}\Phi^*\|_F^2 \le \|\Delta_\Theta\|_F^2\|\Theta^{1.5}\Phi^*(\Phi^*)^\top\Theta^{1.5}\|_2^2 \le \|\Delta\|_{1,\theta}M_{\max}^2, \tag{25}$$

where the second inequality is due to the following argument.

By the feasibility of U^* and Φ^*, it is not hard to see that $\max_{1\le i,j\le n}|\Delta_{ij}| \le 1$. Therefore

$$\|\Delta_\Theta\|_F^2 = \sum_{1\le i,j\le n} \Delta_{ij}^2\theta_i\theta_j \le \sum_{1\le i,j\le n} |\Delta_{ij}|\theta_i\theta_j = \|\Delta\|_{1,\theta}.$$

By properly permuting the row index, the true assignment matrix Φ^* can be written as a block diagonal matrix $\mathrm{diag}\{\mathbf{1}_{n_1\times 1}, ..., \mathbf{1}_{n_k\times 1}\}$, where $\mathbf{1}_{n_i\times 1}, 1 \le i \le k$ is an $n_i \times 1$ vector of all ones. Consequently, $\Phi^*(\Phi^*)^\top = \mathrm{diag}\{\mathbf{1}_{n_1\times n_1}, ..., \mathbf{1}_{n_k\times n_k}\}$ is also a block diagonal matrix, where $\mathbf{1}_{n_i\times n_i}, 1 \le i \le k$ is an $n_i \times n_i$ all-one square matrix. The term $\Theta^{1.5}\Phi^*(\Phi^*)^\top\Theta^{1.5}$ is also a block diagonal matrix. Let $\theta_{(a)} = (\theta_{i_1}, ..., \theta_{i_{n_a}})^\top, \Theta_{(a)} = \mathrm{diag}\{\theta_{(a)}\}$, where $\{i_1, ..., i_{n_a}\} = C_a^*$. Then the ath block of $\Theta^{1.5}\Phi^*(\Phi^*)^\top\Theta^{1.5}$ is

$$\Theta_{(a)}^{1.5}\mathbf{1}\mathbf{1}^\top\Theta_{(a)}^{1.5} = \theta_{(a)}^{1.5}(\theta_{(a)}^{1.5})^\top,$$

whose eigenvalues are M_a and zeros. Hence, we have

$$\|\Theta^{1.5}\Phi^*(\Phi^*)^\top\Theta^{1.5}\|_2^2 = \max_{1\le a\le k} M_a^2 = M_{\max}^2.$$

Therefore, the inequality (25) is proved.

Step (c). We next bound the objective function $v(\theta)$ defined in (23). Note that both matrices $(\Phi^*)^\top\Theta^2\Phi^*(\Phi^*)^\top\Theta^2\Phi^*$ and $(\Phi^*)^\top\Theta^{1.5}\Delta_\Theta\Theta^{1.5}\Phi^*$ are real symmetric, hence are Hermitian and normal. We can apply Lemma 5. For the ease of notation, let the k singular values of $(U^*)^\top\Theta^2\Phi^*$ be $\sigma_1, ..., \sigma_k$, and define $\delta_i = |\sigma_i^2 - J_i^2|$. Lemma 5 implies that

$$\sum_{i=1}^{k} \delta_i^2 \le \|(\Phi^*)^\top\Theta^{1.5}\Delta_\Theta\Theta^{1.5}\Phi^*\|_F^2 \le \|\Delta\|_{1,\theta}M_{\max}^2.$$

Together with

$$\sigma_i \ge \sqrt{J_i^2 - \delta_i} = J_i\sqrt{1 - \frac{\delta_i}{J_i^2}} \ge J_i(1 - \frac{\delta_i}{J_i^2}) = J_i - \frac{\delta_i}{J_i}, \tag{26}$$

we have

$$\begin{aligned} v(\theta) &= 2\left(\sum_{i=1}^{n}\theta_i^2 - \sum_{i=1}^{k}\sigma_i\right) \le 2\left(\sum_{i=1}^{k}J_i - \sum_{i=1}^{k}(J_i - \frac{\delta_i}{J_i})\right) \\ &= 2\sum_{i=1}^{k}\frac{\delta_i}{J_i} \le 2\left(\sum_{i=1}^{k}\frac{1}{J_i^2}\right)^{\frac{1}{2}}\left(\sum_{i=1}^{k}\delta_i^2\right)^{\frac{1}{2}} \\ &\le 2\sqrt{k}\frac{M_{\max}}{J_{\min}}\|\Delta\|_{1,\theta}^{\frac{1}{2}}. \end{aligned}$$

We need to mention that in (26), $\sigma_i > \sqrt{2}J_i > J_i - \frac{\delta_i}{J_i}$ when $J_i^2 - \delta_i < 0$. Consequently, the first inequality of Theorem 6 is established.

(ii) Applying the first inequality with $\Theta = I$ gives

$$\|U^*Q_2^* - \Phi^*\|_F^2 \le 2\sqrt{k}\frac{n_{\max}}{n_{\min}}\|\Delta\|_1^{\frac{1}{2}}.$$

Together with

$$\|\Delta\|_{1,\theta} = \sum_{ij}|\Delta_{ij}|\theta_i\theta_j \ge \theta_{\min}^2\|\Delta\|_1,$$

we prove the second inequality. ∎

The bound on $\|U^*Q_2^* - \Phi^*\|_F^2$ implies that with a proper orthogonal transform, U^* is close to the true community assignment matrix Φ^*. Then if we apply K-means under ℓ_2 norm metric over the rows of U^*, which is equivalent to clustering over rows of $U^*Q_2^*$, the clustering result will intuitively be close to the true community structure. Similar intuition applies to the weighted K-means method (9). Suppose that the solution to the K-means clustering problem (8) is $\bar{\Phi}$. For an arbitrary permutation Π of the columns of $\bar{\Phi}$, we define the set

$$Err_\Pi(\bar{\Phi}) = \{i \mid (\bar{\Phi}\Pi)_i \ne \phi_i^*\},$$

where Π also stands for the corresponding permutation matrix and $(\bar{\Phi}\Pi)_i$ denotes the ith row of the matrix $\bar{\Phi}\Pi$. This set contains the indices of the rows of $\bar{\Phi}\Pi$ and Φ^* that do not match. Since applying any permutation Π to the columns of $\bar{\Phi}$ does not change the cluster structure at all, a fair error measure should be

$$\min_\Pi |Err_\Pi(\bar{\Phi})|,$$

where the minimization is performed over the set of all permutation matrix. Under this measure, the following theorem guarantees the quality of our clustering result.

THEOREM 7. *Suppose that $\bar{\Phi}$ is generated by some K-means clustering algorithm with cluster number k equal to the number of communities. Then under the DCSBM,*

$$\min_\Pi |Err_\Pi(\bar{\Phi})| \le \frac{8\sqrt{k}n_{\max}}{\theta_{\min}n_{\min}}\left(1 + \frac{n_{\max}}{n_{\min}}\right)(1+\alpha(k))\|\Delta\|_{1,\theta}^{\frac{1}{2}},$$

where $\alpha(k)$ is the approximation ratio of the polynomial time K-means approximation algorithm, and $\|\Delta\|_{1,\theta}$ is bounded with high probability by Lemma 4.

Proof. Consider the K-means problem (8). Suppose that the solution generated by some polynomial time approximation algorithm is denoted by $\bar{\Phi}, \bar{X}_c$, and the global optimal solution is denoted by Φ^{opt}, X_c^{opt}. Suppose that the approximation ratio of the algorithm is $\alpha(k)$, where k is the cluster number in the algorithm. Then directly

$$\|\bar{\Phi}\bar{X}_c - U^*\|_F^2 \le \alpha(k)\|\Phi^{opt}X_c^{opt} - U^*\|_F^2.$$

By the optimality of Φ^{opt}, X_c^{opt} and the feasibility of $\Phi^*, (Q_2^*)^\top$, where Q_2^* is defined in Theorem 6, we have

$$\|\Phi^{opt} X_c^{opt} - U^*\|_F^2 \leq \|\Phi^*(Q_2^*)^\top - U^*\|_F^2.$$

Therefore,

$$\begin{aligned}
\|\bar{\Phi}\bar{X}_c - \Phi^*(Q_2^*)^\top\|_F^2 &= \|\bar{\Phi}\bar{X}_c - U^* + U^* - \Phi^*(Q_2^*)^\top\|_F^2 \\
&\leq 2\|\bar{\Phi}\bar{X}_c - U^*\|_F^2 + 2\|\Phi^*(Q_2^*)^\top - U^*\|_F^2 \\
&\leq 2\alpha(k)\|\Phi^{opt} X_c^{opt} - U^*\|_F^2 + 2\|\Phi^*(Q_2^*)^\top - U^*\|_F^2 \\
&\leq 2(1+\alpha(k))\|\Phi^*(Q_2^*)^\top - U^*\|_F^2.
\end{aligned}$$

Denote by Z_i the ith row of a matrix Z. We define the set that contains the potential errors

$$E_a = \left\{ i \in C_a^* | \| \left(\bar{\Phi}\bar{X}_c\right)_i - \left(\Phi^*(Q_2^*)^\top\right)_i \|^2 \geq \frac{1}{2} \right\},$$

$$E = \cup_{a=1}^k E_a, S_a = C_a^* \backslash E_a.$$

A straightforward bound for the cardinality of E gives

$$|E| \leq 2\|\bar{\Phi}\bar{X}_c - \Phi^*(Q_2^*)^\top\|_F^2.$$

Now we define a partition of the community index set $[k] = \{1, ..., k\}$,

$$\begin{aligned}
B_1 &= \{a \mid S_a = \emptyset\}, \\
B_2 &= \{a \mid \forall i, j \in S_a, (\bar{\Phi}\bar{X}_c)_i = (\bar{\Phi}\bar{X}_c)_j\}, \\
B_3 &= [k] \backslash (B_1 \cup B_2).
\end{aligned}$$

For $\forall i, j \in E^c = \cup_{a \in B_2 \cup B_3} S_a$, and $i \in C_a^*, j \in C_b^*, a \neq b$,

$$\begin{aligned}
&\|(\bar{\Phi}\bar{X}_c)_i - (\bar{\Phi}\bar{X}_c)_j\| \\
\geq\ &\|(\Phi^*(Q_2^*)^\top)_i - (\Phi^*(Q_2^*)^\top)_j\| - \|(\bar{\Phi}\bar{X}_c)_i - (\Phi^*(Q_2^*)^\top)_i\| \\
&-\|(\bar{\Phi}\bar{X}_c)_j - (\Phi^*(Q_2^*)^\top)_j\| \\
>\ &\sqrt{2} - \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2} = 0.
\end{aligned}$$

That is, $(\bar{\Phi}\bar{X}_c)_i \neq (\bar{\Phi}\bar{X}_c)_i$, or equivalently $\bar{\Phi}_i \neq \bar{\Phi}_j$. This implies that for all $S_a, a \in B_2$, they belong to different clusters. It further indicates that all nodes in $\cup_{a \in B_2} S_a$ are successfully classified with a proper permutation Π^*. In other words,

$$Err_{\Pi^*}(\bar{\Phi}) \subset \cup_{a \in B_3} S_a \cup E.$$

The above argument also reveals that if $a \neq b, S_a, S_b \neq \emptyset$, then S_a and S_b will correspond to different rows from $\bar{X}_c$. If $a \in B_2$, then S_a corresponds to only one row of $\bar{X}_c$. If $a \in B_3$, then S_a corresponds to at least two different rows of $\bar{X}_c$. But note that $\bar{X}_c$ contains at most k different rows. Therefore,

$$|B_2| + 2|B_3| \leq k = |B_1| + |B_2| + |B_3|,$$

which further implies

$$|B_3| \leq |B_1|.$$

Note that $\cup_{a\in B_1} C_a^* \subset E$, we have $|B_1| n_{\min} \leq |E|$, or equivalently

$$|B_1| \leq \frac{1}{n_{\min}} |E|.$$

Combining all previous results yields

$$\begin{aligned}
|Err_{\Pi^*}(\bar{\Phi})| &\leq |E| + |\cup_{a\in B_3} S_a| \leq |E| + |B_3| n_{\max} \\
&\leq (1 + \frac{n_{\max}}{n_{\min}})|E| \\
&\leq 4(1 + \frac{n_{\max}}{n_{\min}})(1+\alpha(k))\|U^*Q_2^* - \Phi^*\|_F^2 \\
&\leq \frac{8\sqrt{k} n_{\max}}{\theta_{\min} n_{\min}} \left(1 + \frac{n_{\max}}{n_{\min}}\right)(1+\alpha(k))\|\Delta\|_{1,\theta}^{\frac{1}{2}}.
\end{aligned}$$

This completes the proof. ■

Note that the factor $\frac{1}{\theta_{\min}}$ appears in the bound. This term vanishes under the SBM where all $\theta_i = 1$. However, it is undesirable under the DCSBM. To avoid the dependence on this term, we propose to solve the weighted K-means problem (9). In this case, suppose that the solution to problem (9) is $\bar{\Phi}$, then the fair error measure will be

$$\min_{\Pi} \sum_{i\in Err_{\Pi}(\bar{\Phi})} \theta_i^2,$$

where Π is a permutation matrix. This measure means that when a node has smaller θ_i, it potentially has few edges and provides less information for clustering, then misclassifying this node is somewhat forgivable and suffers less penalty. In contrast, if a node has large θ_i, then misclassifying this node should incur more penalty. Now we present the theorem that address the above issue.

THEOREM 8. *Suppose that $\bar{\Phi}$ is generated by some weighted K-means clustering algorithm with the number of clusters k equal to the number of communities. Then under the DCSBM,*

$$\min_{\Pi} \sum_{i\in Err_{\Pi}(\bar{\Phi})} \theta_i^2 \leq \frac{16\sqrt{k} J_{\max}}{H_{\min}^2 J_{\min}} \left(C_H + H_{\max}^2\right)(1+\alpha(k))\|\Delta\|_{1,\theta}^{\frac{1}{2}},$$

where $\alpha(k)$ is the approximation ratio of the polynomial time approximation algorithm for problem (9) and the quantity $C_H \ll H_{\max}^2$.

The proof of this theorem is similar in nature to that of Theorem 7, but is more technically involved. For the sake of being succinct, we omit the proof here.

For large scale networks, solving problem (7) can be very efficient when the ℓ_0 penalty parameter p is set to be moderate and the asynchronous parallel scheme is applied. However, performing the K-means or weighted K-means clustering will be a bottleneck when the number of communities is relatively large. Therefore, the proposed direct rounding scheme will be desirable. In the following theorem, we discuss the theoretical guarantees for this rounding scheme. To invoke the analysis, the following assumption is needed.

ASSUMPTION 9. *Define the set $Err(Q,\delta) = \{i : \|(U^*Q)_i - \phi_i^*\|^2 \geq \delta\}$ and define $T_a(Q,\delta) = C_a^* \backslash Err(Q,\delta)$. Suppose Q_1^*, Q_2^* are defined according to Theorem 6, we assume that all $T_a(Q_1^*, \frac{1}{32p^2}) \neq \emptyset, T_a(Q_2^*, \frac{1}{32p^2}) \neq \emptyset$.*

This assumption states that for each community C_a^*, there is at least one node $i \in C_a^*$ such that U_i^* is close to the true assignment ϕ_i^* after the rotation Q_1^* or Q_2^*. In our extensive numerical experiments, this assumption actually always holds.

THEOREM 10. *Suppose that all Assumption 9 holds and the community assignment $\bar{\Phi}$ is given by a directly rounding U^*. Then under the DCSBM, the error is bounded by*

$$\min_{\Pi} |Err_{\Pi}(\bar{\Phi})| \leq C_2 \frac{p^2\sqrt{k} n_{\max}}{n_{\min}\theta_{\min}} \|\Delta\|_{1,\theta}^{\frac{1}{2}},$$

$$\min_{\Pi} \sum_{i \in Err_{\Pi}(\bar{\Phi})} \theta_i^2 \leq C_2 \frac{p^2\sqrt{k} M_{\max}}{J_{\min}} \|\Delta\|_{1,\theta}^{\frac{1}{2}},$$

where $C_2 > 0$ is some absolute constant.

Proof. (i) We prove the first inequality by showing that all nodes in $\cup_{a=1}^k T_a(Q_2^*, \frac{1}{32p^2})$ are correctly classified under a proper permutation matrix Π. Define $\delta_0 = \frac{1}{32p^2}$. By the Assumption 9, we have

$$|Err(Q_2^*, \delta_0)| \leq \frac{1}{\delta_0} \|U^* Q_2^* - \Phi^*\|_F^2, \text{ and } T_a(Q_2^*, \delta_0) \neq \emptyset, 1 \leq a \leq k.$$

First, for $\forall i, j \in T_a(Q_2^*, \delta_0)$, and for some $a \in [k]$, we have $\phi_i^* = \phi_j^*$. Recall that $(u_i^*)^\top$ stands for the ith row of the solution U^* to problem (7), we further have

$$\begin{aligned} \|u_i^* - u_j^*\| &= \|(Q_2^*)^\top (u_i^* - u_j^*)\| \\ &= \|(Q_2^*)^\top u_i^* - \phi_i^* + \phi_j^* - (Q_2^*)^\top u_j^*\| \\ &\leq \|(Q_2^*)^\top u_i^* - \phi_i^*\| + \|(Q_2^*)^\top u_j^* - \phi_j^*\| \\ &< 2\sqrt{\delta_0}. \end{aligned}$$

Second, for $\forall i, j, a \neq b, i \in T_a(Q_2^*, \delta_0), j \in T_b(Q_2^*, \delta_0)$, we have $\phi_i^* \neq \phi_j^*$. Similarly,

$$\begin{aligned} \|u_i^* - u_j^*\| &= \|(Q_2^*)^\top (u_i^* - u_j^*)\| \\ &= \|(Q_2^*)^\top u_i^* - \phi_i^* + \phi_i^* - \phi_j^* + \phi_j^* - (Q_2^*)^\top u_j^*\| \\ &\geq \|\phi_i^* - \phi_j^*\| - \|(Q_2^*)^\top u_i^* - \phi_i^*\| - \|(Q_2^*)^\top u_j^* - \phi_j^*\| \\ &> \sqrt{2} - 2\sqrt{\delta_0}. \end{aligned}$$

That is,

$$\|u_i^* - u_j^*\| \begin{cases} < 2\sqrt{\delta_0}, & \forall a, \forall i, j \in T_a(Q_2^*, \delta_0), \\ > \sqrt{2} - 2\sqrt{\delta_0}, & \forall a \neq b, \forall i \in T_a(Q_2^*, \delta_0), \forall j \in T_b(Q_2^*, \delta_0). \end{cases} \tag{27}$$

From each set $T_a(Q_2^*, \delta_0)$, we take an arbitrary representative v_a. Then we have k vectors in $\mathbb{R}^k$ such that

$$\|v_i\|^2 = 1, \|v_i\|_0 \leq p, v_i \geq 0, \forall 1 \leq i \leq k,$$

$$\|v_i - v_j\| \geq \sqrt{2} - 2\sqrt{\delta_0}, \forall i \neq j,$$

which further implies that

$$\langle v_i, v_j \rangle \leq \epsilon, \forall i \neq j,$$

where $\epsilon = \frac{1}{2p} \geq 2(\sqrt{2\delta_0} - \delta_0)$. The following proof consists of mainly three steps.

Step (a). Define $m(i) = \arg\max_{1\le j\le k}(v_i)_j$, where $(v_i)_j$ is the jth component of v_i. Then for any $i \neq j$, we prove $m(i) \neq m(j)$. Suppose there exist $i \neq j$ such that $m(i) = m(j)$. Then by the definition of $m(i)$, it is straightforward that

$$(v_i)_{m(i)} \geq \frac{1}{\sqrt{p}}.$$

Therefore, by $v_i, v_j \geq 0$,

$$\langle v_i, v_j\rangle \geq (v_i)_{m(i)} \cdot (v_j)_{m(j)} \geq \frac{1}{p} > \frac{1}{2p} = \epsilon,$$

which leads to a contradiction. We can then choose a proper permutation of index Π such that $m(i) = i$ for $1 \leq i \leq k$. This means that all the k representatives $v_1, ..., v_j$ are correctly classified.

Step (b). Suppose that after a proper permutation of indices, $m(i) = i, 1 \leq i \leq k$. Then in this step, we prove that for all $1 \leq i \leq k$, v_i is sufficiently close to e_i. First, by

$$(v_1)_2 \cdot \frac{1}{\sqrt{p}} \leq (v_1)_2 \cdot (v_2)_2 \leq \langle v_1, v_2\rangle \leq \frac{1}{2p}, \tag{28}$$

we obtain,

$$(v_1)_2 \leq \frac{1}{2\sqrt{p}}. \tag{29}$$

Repeat the same argument for $(v_1)_3, ..., (v_1)_k$ and we have them all less than or equal to $\frac{1}{2\sqrt{p}}$. It follows from $\|v_1\|_0 \leq p$ and $\|v_1\|^2 = 1$ that

$$(v_1)_1^2 = 1 - \sum_{j=2}^{k}(v_1)_j^2 \geq 1 - \frac{p-1}{4p} > \frac{3}{4}. \tag{30}$$

By repeating the same argument for $v_2, ..., v_k$, we have for $1 \leq i \leq k$,

$$\begin{cases} (v_i)_i > \frac{\sqrt{3}}{2}, \\ (v_i)_j < \frac{1}{2\sqrt{p}}, \quad \text{for } \forall j \neq i. \end{cases} \tag{31}$$

Now based on the new lower bounds of $(v_i)_i, 1 \leq i \leq k$, we repeat the process (28), (30) and (31) again. We obtain that for all $1 \leq i \leq k$,

$$\begin{cases} (v_i)_i > \sqrt{1 - \frac{1}{3p}}, \\ (v_i)_j < \frac{1}{\sqrt{3p}}, \text{ for } \forall j \neq i. \end{cases}$$

Step (c). Now we argue that all nodes in $\cup_{1\le a\le k}T_a(Q_2^*, \delta_0)$ are correctly classified in this step. For any other nodes $s \in T_a(Q_2^*, \delta_0)$, due to its proximity to v_a (27), we know that

$$\begin{cases} (u_s^*)_a > \sqrt{1 - \frac{1}{3p}} - \frac{\sqrt{2}}{4p}, \\ (u_s^*)_j < \frac{1}{\sqrt{3p}} + \frac{\sqrt{2}}{4p}, \forall j \neq a. \end{cases}$$

For all $p \geq 2$, $u_s^*(a) > u_s^*(j)$ for $j \neq a$. This means that all nodes in $\cup_{1\le a\le k}T_a(Q_2^*, \delta_0)$ are correctly classified. That is,

$$\min_{\Pi} |Err_{\Pi}(\bar{\Phi})| \leq |Err(Q_2^*, \delta_0)| \leq \frac{1}{\delta_0} \cdot \|U^*Q_2^* - \Phi^*\|_F^2$$

$$\leq C_2 \frac{p^2\sqrt{k} n_{\max}}{n_{\min}\theta_{\min}} \|\Delta\|_{1,\theta}^{\frac{1}{2}},$$

where the last inequality follows from the result of Theorem 6.

(ii). The second inequality of this theorem can be extended directly from the first one:

$$\sum_{i\in Err(Q_1^*,\delta_0)} \theta_i^2 \leq \sum_{i\in Err(Q_1^*,\delta_0)} \theta_i^2 \cdot \frac{1}{\delta_0} \cdot \|(Q_1^*)^\top u_i^* - \phi_i^*\|^2 \leq \frac{1}{\delta_0}\|\Theta(U^*Q_1^* - \Phi^*)\|_F^2.$$

Then by following the bound in Theorem 6, the proof is completed. ■

5. Numerical Result. In this section, we evaluate the performance of our RBR method on both synthetic and real datasets, where the parameter λ is set to $1/\|d\|_1$ in all cases. Our RBR algorithm with rounding and K-means are referred to as RBR(r) and RBR(K), respectively. For synthetic and small-sized real networks, we compare them with the CMM algorithm in [4], the SCORE algorithm in [16] and the OCCAM algorithm in [35]. For large-sized real datasets, we only compare the asynchronous parallel RBR algorithm with the LOUVAIN algorithm in [1], which is an extremely fast algorithm to process large networks.

5.1. Solvers and Platform. The algorithm RBR(r) is implemented in C, with multi-threading support using OpenMP. Due to the usage of K-means, the version RBR(K) is written in MATLAB. The CMM, SCORE and OCCAM for synthetic and small-sized real networks are also written in MATLAB. For large-sized networks, we use the LOUVAIN solver hosted on Google Sites[1], which is implemented in C++. The parameters in all methods are set to their default values unless otherwise specified. Since k, the number of communities of the synthetic and small-sized real networks, is known (usually very small), the parameter r, i.e., the number of columns of U is set to k. The parameter p in the ℓ_0 constraints of our RBR algorithm is also set to k in these tests. A few different values of p are tested on large-size real networks.

All numerical experiments are performed on a workstation with two twelve-core Intel Xeon E5-2680 v3 processors at 2.5 GHz and a total amount of 128 GB shared memory. The efficiency of OpenMP may be affected by NUMA systems greatly. It is much slower for a CPU to access the memory of another CPU. To prevent such allocations of threads, we bind all programs to one CPU, i.e., up to 12 physical cores are available per task.

Other programming environments are:

- gcc/g++ version 6.2.0 is used for compiling our C programs and LOUVAIN executables.
- MATLAB version R2015b is used for running CMM, SCORE, OCCAM and RBR with K-means solvers.
- Intel MKL 2017 for BLAS routines, linked with `libmkl_sequential.so`.

We should mention that there are threaded and sequential versions in the Intel MKL's library. The threaded MKL library should be disabled in favor of our manual parallelization. The reason is that when updating one row, the threaded library will automatically utilize all available computing threads for BLAS, which may interfere with the cores processing other rows. Besides, the sequential library is usually faster than the parallel library with one thread. The reported runtimes are wall-clock times in seconds.

5.2. Results on Synthetic Data. To set up the experiments, we generate the synthetic graph of $n = m \times k$ nodes and split them into k groups $C_1^*, \ldots, C_k^*$, each group with m nodes. These groups are then used as the ground truth of the model. The edges of the graph

[1] See https://sites.google.com/site/findcommunities/

are generated by sampling. For each pair of nodes $i \in C_a^*$ and $j \in C_b^*$, i and j are connected with probability $\min\{1, \theta_i\theta_j B_{ab}\}$, where

$$B_{ab} = \begin{cases} q, & a = b, \\ 0.3q, & a \neq b. \end{cases} \tag{32}$$

For each i, θ_i is sampled independently from a $\mathrm{Pareto}(\alpha, \beta)$ distribution with probability density function $f(x; \alpha, \beta) = \frac{\alpha\beta^\alpha}{x^{\alpha+1}} \mathbf{1}_{\{x \geqslant \beta\}}$. Here α and β are called the *shape* and *scale* parameters respectively. In our testing model, we choose different shape parameter α first, and select the scale parameter β such that $\mathbb{E}(\theta_i) = 1$ for all i. These parameters determine the strength of the cluster structure of the graph. With larger q and α, it is easier to detect the community structure. Otherwise all these algorithms may fail. Since the ground truths of the synthetic networks are known, we compute the misclassification rate to check the correctness of these algorithms. Suppose that $C_1, \ldots, C_k$ are the detected communities provided by any algorithm, then the misclassification rate can be defined as

$$\mathrm{err} := 1 - \frac{\sum_i^k \max_j |C_i \cap C_j^*|}{n}. \tag{33}$$

We shall mention that permuting the columns of U does not change its objective function value. Since it is impractical to enumerate all possible permutations and choose the best one to match the detected communities and the ground truth when k is large, we still use the definition of misclassification rate in (33), and it works in most cases during our numerical experiments.

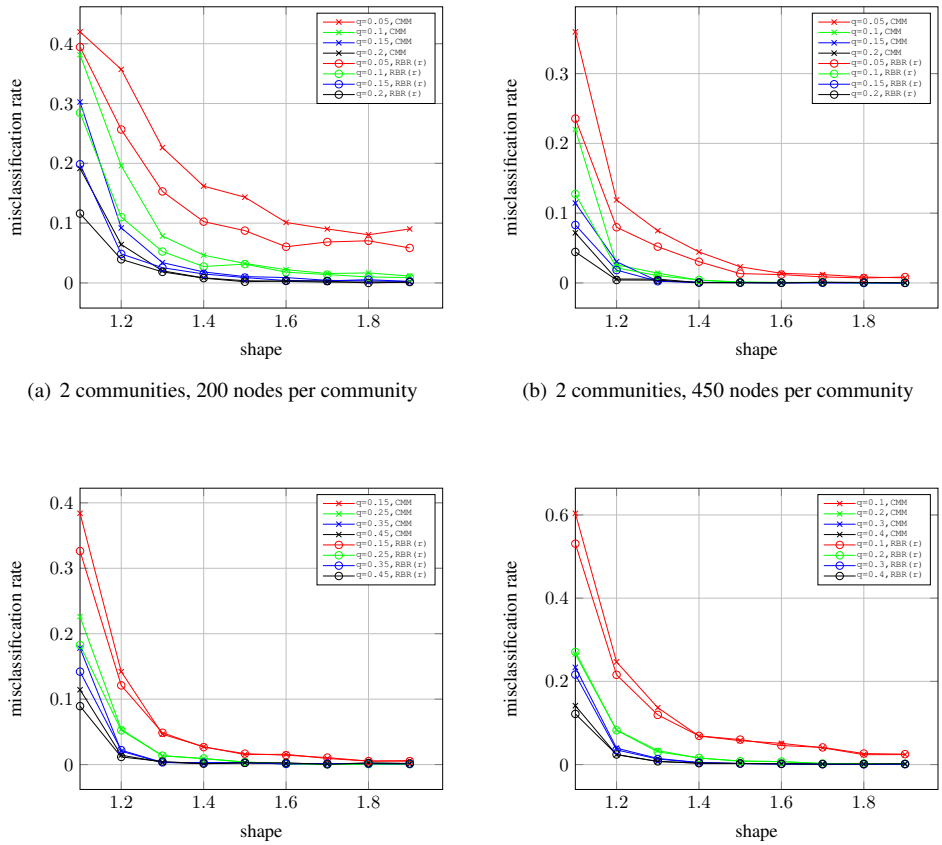
The performance of our method is shown in Figure 1. The y-axis is the misclassification rate while the x-axis is the shape parameter α by which we generate the random graphs. The circle solid line is our RBR algorithm with rounding. The line with "x" marker stands for the CMM algorithm. The q in the figure denotes the intra-cluster connection probability. The inter-cluster connection probability is set to be $0.3q$. It is clear that in all these cases, our approach outperforms the CMM algorithm, especially when in class connection probability is small, i.e., the class structure is not very strong. Table 1 and 2 are the results under $q = 0.1, \alpha = 1.4$ and $q = 0.1, \alpha = 1.8$, respectively. We can see that CMM and our method have similar performances. Both of them are better than SCORE and OCCAM. If the shape parameter is increased from 1.4 to 1.8, all of the four methods perform better, but our method still has the smallest misclassification rate.

TABLE 1
Misclassification rate and runtime, $q = 0.1, \alpha = 1.4$

	$n = 200, k = 2$		$n = 450, k = 2$		$n = 200, k = 3$		$n = 200, k = 4$	
solver	err	Time	err	Time	err	Time	err	Time
CMM	1.06%	5.10	0.04%	22.05	0.30%	10.43	0.25%	18.37
RBR(r)	0.90%	0.01	0.04%	0.03	0.27%	0.03	0.23%	0.07
RBR(K)	1.05%	1.35	0.04%	4.33	0.30%	3.00	0.25%	4.37
OCCAM	1.50%	0.02	20.02%	0.05	19.13%	0.04	22.35%	0.08
SCORE	1.50%	0.02	18.02%	0.03	7.43%	0.05	14.90%	0.09

5.3. Results on Small-Scaled Real World Data. In this section, we test the empirical performances of CMM, SCORE, OCCAM, RBR(r) and RBR(K) on the US political blog network dataset from [20], the Simmons College network and the Caltech network from

FIG. 1. *Numerical results on synthetic data.*



(a) 2 communities, 200 nodes per community

(b) 2 communities, 450 nodes per community

(c) 3 communities, 200 nodes per community

(d) 4 communities, 200 nodes per community

TABLE 2
Misclassification rate and runtime, $q = 0.1, \alpha = 1.8$

	$n = 200, k = 2$		$n = 450, k = 2$		$n = 200, k = 3$		$n = 200, k = 4$	
solver	err	Time	err	Time	err	Time	err	Time
CMM	0.25%	5.20	0.00%	22.77	0.10%	10.44	0.10%	18.66
RBR(r)	0.25%	0.01	0.00%	0.04	0.10%	0.03	0.07%	0.06
RBR(K)	0.15%	1.41	0.00%	4.76	0.10%	3.24	0.10%	5.08
OCCAM	0.45%	0.02	0.00%	0.04	0.13%	0.04	2.88%	0.06
SCORE	0.45%	0.02	0.02%	0.03	0.27%	0.04	2.12%	0.05

Facebook dataset. The properties of these networks are shown in Table 3, where nc is the number of real communities.

We run our RBR algorithms for 10 times as a batch starting from different randomly initial points. The one with the highest modularity is chosen as the result in this batch. This procedure is repeated 300 times. The histograms of the misclassification rates of RBR(r) are shown in Figure 2. A summary of the misclassification rates and the cpu time of all algorithms is presented in Table 4, where the results for RBR(r) and RBR(K) are the averaged values.

On the polblogs network, the smallest misclassification rate of RBR(r) method is 4.4%

TABLE 3
Small-scale real-world networks with ground truth

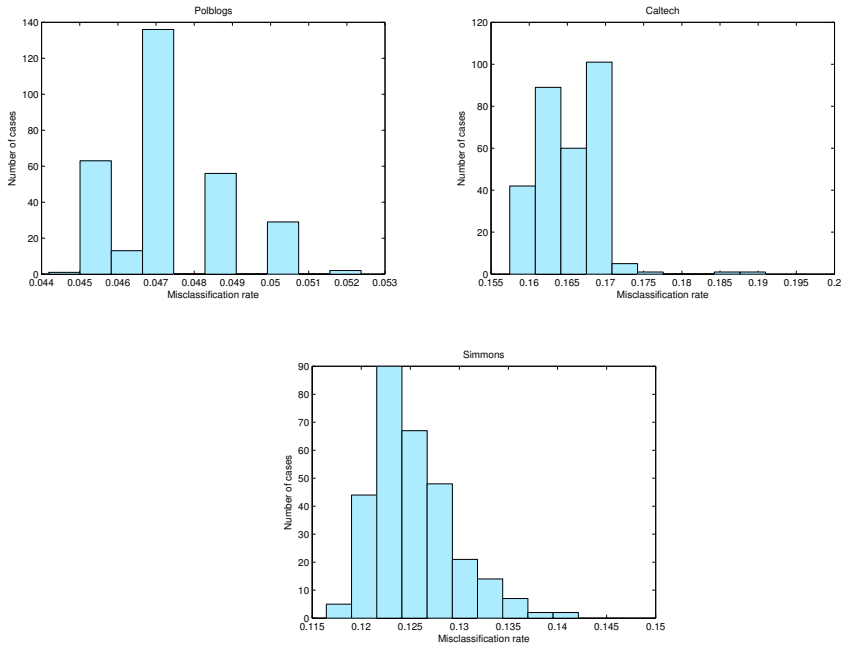
Name	nodes	edges	nc	feature of community stucture
Polblogs	1222	16714	2	political leaning
Simmons	1168	24449	4	graduation year between 2006 and 2009
Caltech	597	12823	8	total 8 dorm numbers

TABLE 4
Overall comparison between the five methods

	Polblogs		Simmons		Caltech	
	err	Time	err	Time	err	Time
CMM	4.99%	50.12	13.10%	53.74	21.94%	14.22
SCORE	4.75%	0.05	23.92%	0.11	27.18%	0.17
OCCAM	5.32%	0.10	24.36%	0.16	35.53%	0.10
RBR(K)	5.09%	1.42	12.56%	10.53	16.51%	1.08
RBR(r)	4.75%	0.04	14.00%	0.08	21.03%	0.06

while its averaged value is 4.75%. For the Caltech network, we use dorms as the ground truth cluster labels according to [31]. The nodes are partitioned into 8 groups. The averaged misclassfication rate of RBR(K) and RBR(r) are 16.54% and 21.03%, respectively, whereas CMM, SCORE and OCCAM have higher error rate of 21.94%, 27.18% and 35.53%, respectively. In particular, the smallest error by RBR(K) is 15.55%. For the Simmons College network, the graduation year ranging from 2006 to 2009 is treated as cluster labels for its strong correlation with the community structure observed in [31]. RBR(r), RBR(K), CMM, SCORE and OCCAM misclassified 14.00%, 12.56%, 13.10%, 23.92% and 24.36% of the nodes on average, respectively. The smallest error of RBR(K) reaches 11.5%.

FIG. 2. *Histograms of misclassification rates*



Finally, the confusion matrices of CMM, SCORE and RBR(r) on these networks are presented in Figures 3, 4 and 5, respectively. The results for OCCAM and RBR(K) are not shown due to their similarity with SCORE and RBR(r), respectively. We can see that RBR(r) is indeed competitive on identifying good communities.

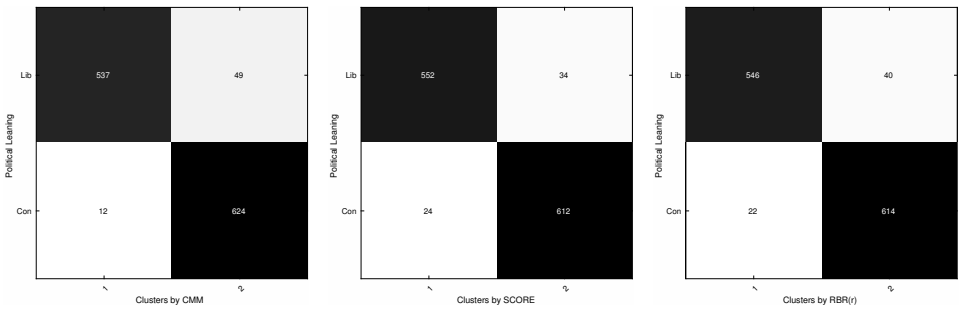


FIG. 3. *The confusion matrices of CMM, SCORE and RBR(r) on Polblogs network.*

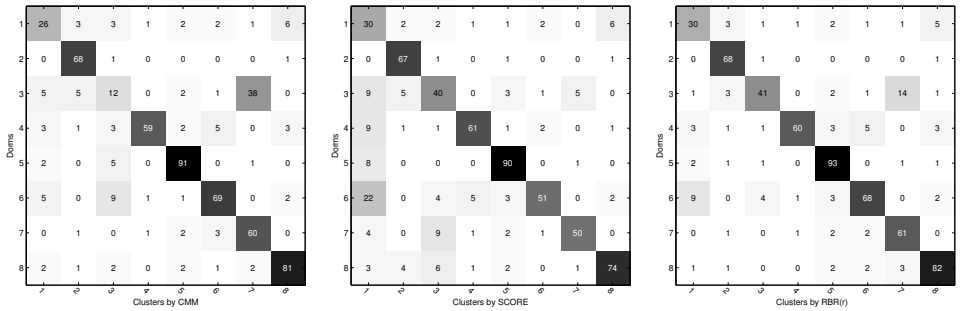


FIG. 4. *The confusion matrices of CMM, SCORE and RBR(r) on Caltech network.*

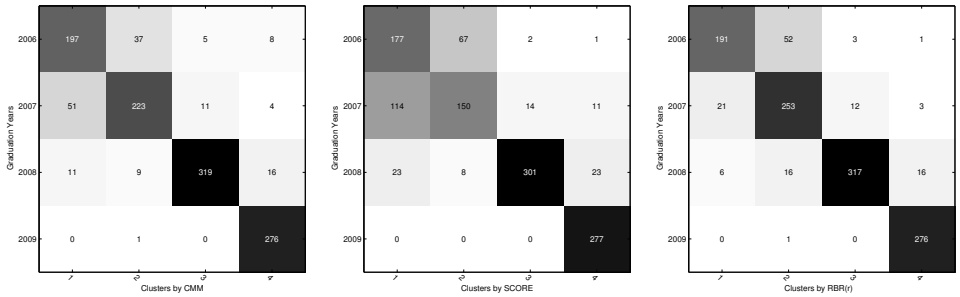


FIG. 5. *The confusion matrices of CMM, SCORE and RBR(r) on Simmons College network.*

5.4. Results on Large-Scaled Data. For large-sized network, we choose ten undirected networks from Stanford Large Network Dataset Collection [20] and UF Sparse Matrix Collection [9]. A detailed description of these networks can be found in Table 5. The number of nodes in the dataset varies from 36 thousand to 50 million. Due to the size of these networks, we run the RBR(r) algorithm on these networks for only once.

For these large networks, we do not have the true partition. Therefore we employ three metrics to evaluate the quality of the clustering.

TABLE 5
Large-scale networks

Name	nodes	edges	Descriptions
amazon	334,863	925,872	Amazon product network
DBLP	317,080	1,049,866	DBLP collaboration network
email-Enron	36,692	183,831	Email communication network from Enron
loc-Gowalla	196,591	950,327	Gowalla location based online social network
loc-Brightkite	58,228	214,078	Brightkite location based online social network
youtube	1,134,890	2,987,624	Youtube online social network
LiveJournal	3,997,962	34,681,189	LiveJournal online social network
Delaunay_n24	16,777,216	50,331,601	Delaunay triangulations of random points in the plane
road_usa	23,947,347	28,854,312	Road network of USA
europe_osm	50,912,018	54,054,660	Road network of Europe

- **CC**: *Cluster Coefficient(CC)* is defined as

$$CC = \frac{1}{nC}\sum_{i=1}^{nC}\left(\frac{1}{|C_i|}\sum_{v\in C_i}\frac{2|\{e_{ts}\in E\ :\ v_t, v_s\in N(v)\cap C_i\}|}{d(v)(d(v)-1)}\right) \tag{34}$$

  where nC is total number of detected communities; $|C_i|$ is the cardinality of the community C_i; $N(v)$ is the set that contains all neighbors of node v. Thus, high CC means that the connections in each cluster are dense. That is, it assumes that for any v within class C, the neighborhood of v should have the structure of a complete graph. In some sparse graphs such as road nets, the CC value may be small.
- **S**: *Strength(S)* shows that a community is valid if most nodes have the same label as their neighbors. Denote $d(v)^{in}$ and $d(v)^{out}$ as the degrees inside and outside the community C containing v. If for all v in C, we have $d(v)^{in} > d(v)^{out}$, then C is a strong community; if $\sum_{v\in C} d(v)^{in} > \sum_{v\in C} d(v)^{out}$, then C is a weak community; otherwise C not valid as a community. Consequently, S is defined as

$$S = \frac{1}{nC}\sum_{i=1}^{nC}\mathrm{score}(C_i), \tag{35}$$

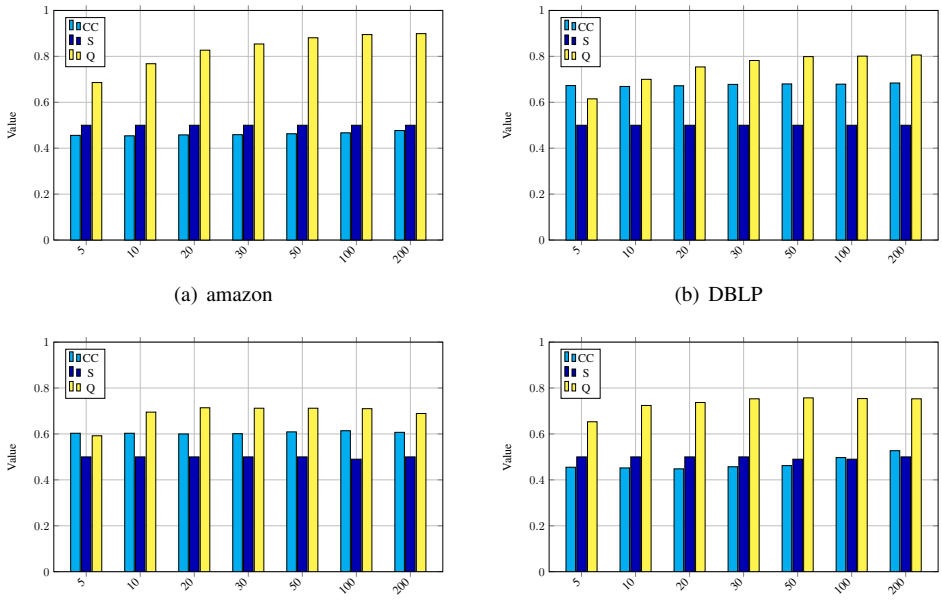
  where

$$\mathrm{score}(C_i) = \begin{cases} 1, & C_i \text{ is strong,} \\ 0.5, & C_i \text{ is weak,} \\ 0, & C_i \text{ is invalid.} \end{cases}$$

  A cluster with a high S implies that the neighbouring nodes tend to have the same label, which is true for most graphs with community structures.
- **Q**: *Modularity(Q)* is a widely used measure in community detection. In our RBR algorithm, the modularity is the absolute value of the objective function value modulo a constant factor $2|E|$.

We first evaluate the performance of RBR(r) with respect to different values of k, i.e., the expected number of communities. Specifically, for each $k \in \{5, 10, 20, 30, 50, 100, 200\}$, we perform a fixed number of iterations and report the corresponding values of the metrics (CC, S, Q). The paramter p is set to 5 in all cases. The results are shown in Figure 6. The modualarity Q increases as k becomes larger but it becomes almost the same when $k \geqslant 50$. From an optimization perspective, a larger k means a larger feasible domain. Therefore, a

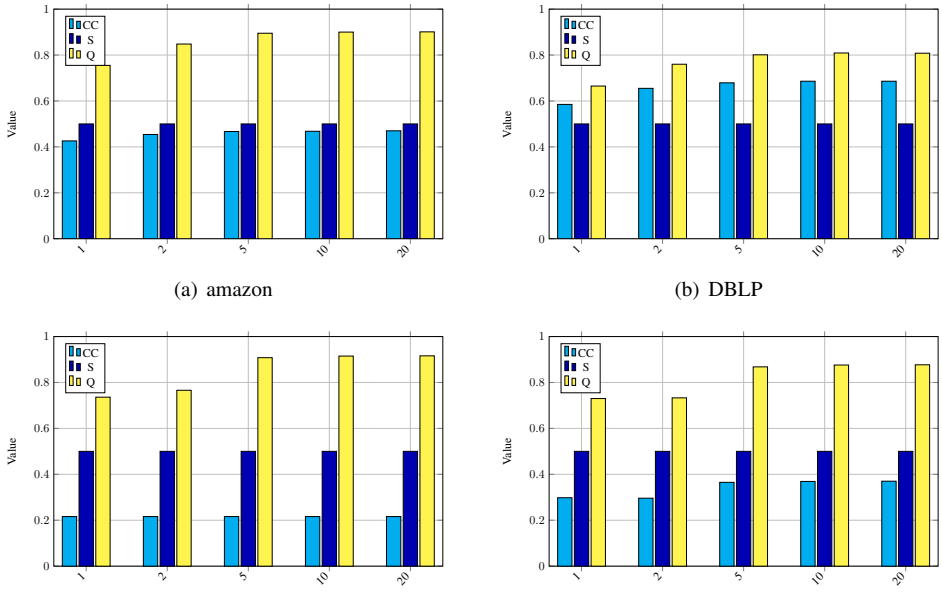
FIG. 6. *Metrics with respect to different values of k*



(a) amazon (b) DBLP

(c) youtube (d) LiveJournal

FIG. 7. *Metric with respect to different values of p*



(a) amazon (b) DBLP

(c) road_usa (d) Delaunay_n24

better value should be expected. On the other hand, the values of CC and S are almost the same for all k.

We next compare the performance of RBR(r) with respect to different values of p in the

ℓ_0 constraint over each subproblem. By taking advantage of the sparsity, RBR is able to solve problems of very large scale since the storage of U is $\mathcal{O}(np)$. For each $p \in \{1, 2, 5, 10, 20\}$, we set $k = 100$ and perform a fixed number of iterations. The results are shown in Figure 7. Ideally, RBR tends to search solutions in a larger space for the subproblem if a larger p is applied. From the figure, we can observe that RBR performs better when a larger p is used. The metrics become similar when $p \geqslant 5$ but they are better than the results with values $p = 1$ and $p = 2$.

Finally we compare our parallel RBR with LOUVAIN, which runs extremely fast and usually provides good results. It is worth mentioning that LOUVAIN can begin with different initial solutions. By default it starts from a trivial classification that each node belongs to a community that only contains itself. It merges clusters in order to obtain a higher modularity. LOUVAIN terminates if it is unable to increase the modularity or the improvement is not significant. In this sense, there is no guarantee on the number of communities it finally provides. On the other hand, one can limit the maximal number of communities by letting LOUVAIN start from a partition with no more than k communities. Since LOUVAIN always reduces the number of communities by merging, it is possible to obtain a low rank solution. However, the quality of the solution may be highly dependent on the initial values. Without utilizing much prior knowledge of these networks, we set the initial value of both algorithms to be a random partition with at most k labels.

A summary of the computational results are shown in Table 6. For each algorithm, three sets of parameters are tested. For example, RBR(100, 1) denotes that the desired number of communities k is set to 100 and p is set to 1; LV(500) means that LOUVAIN begins with a randomly assigned partition with at most 500 communities. LV(0), however, means that we launch LOUVAIN with a trivial partition, in which case it may not produce low rank solutions. In the table, k_0 stands for the actual number of communities that the algorithm finally produces. If k is given, then k_0 will not exceed k. The CPU time of each algorithm is also reported.

From the perspective of solution qualities, LV(0) outperforms other algorithms and settings. Its CC, S, Q are high on most networks. Especially for the last three large networks, the modularity Q is over 0.99. However, it is hard to control k_0 in LV(0). It is acceptable on networks such as DBLP and Delaunay_n24. It may produce over one thousand communities and k_0 is over 28000 in the youtube network. Although LV(500) and LV(2000) can return partitions with a small k_0, their modularity is not as good as LV(0).

We can see that RBR(100,5) almost always returns a good moduality. For instance, it is better than LV(0) on email-Enron. Note that RBR(20,5) is also very competitive due to its low rank solutions without sacrificing the modularity too much. These facts indicate that RBR is better than LOUVAIN if a small number of communities are needed. Additionally, it can be observed that RBR(100,1) can produce similar results as LV(500) and LV(2000) do. In fact, the procedure of these algorithms are similar in the sense of finding a new label for a given node and increasing the modularity Q. The difference is that LOUVAIN only considers the labels of the node's neighbours while RBR searches for the new label among all k possibilities. If RBR is modified properly, it can also utilize the network structure.

LOUVAIN is faster in terms of the running time. Each step of LOUVAIN only considers at most $d_{\max}$ nodes, where $d_{\max}$ is the maximum of the vertex degrees, while RBR has to sort all k possible labels and choose at most p best candidates. However, RBR(20,5) is still competitive on most graphs in terms of both speed and the capability of identifying good communities. It is the fastest one on networks such as LiveJournal and Delaunay_n24. Of course, it is possible for RBR to consider fewer labels according to the structure of the network in order to reduce the computational cost of sorting.

We should point out that LOUVAIN is run in a single threaded mode since its multi-

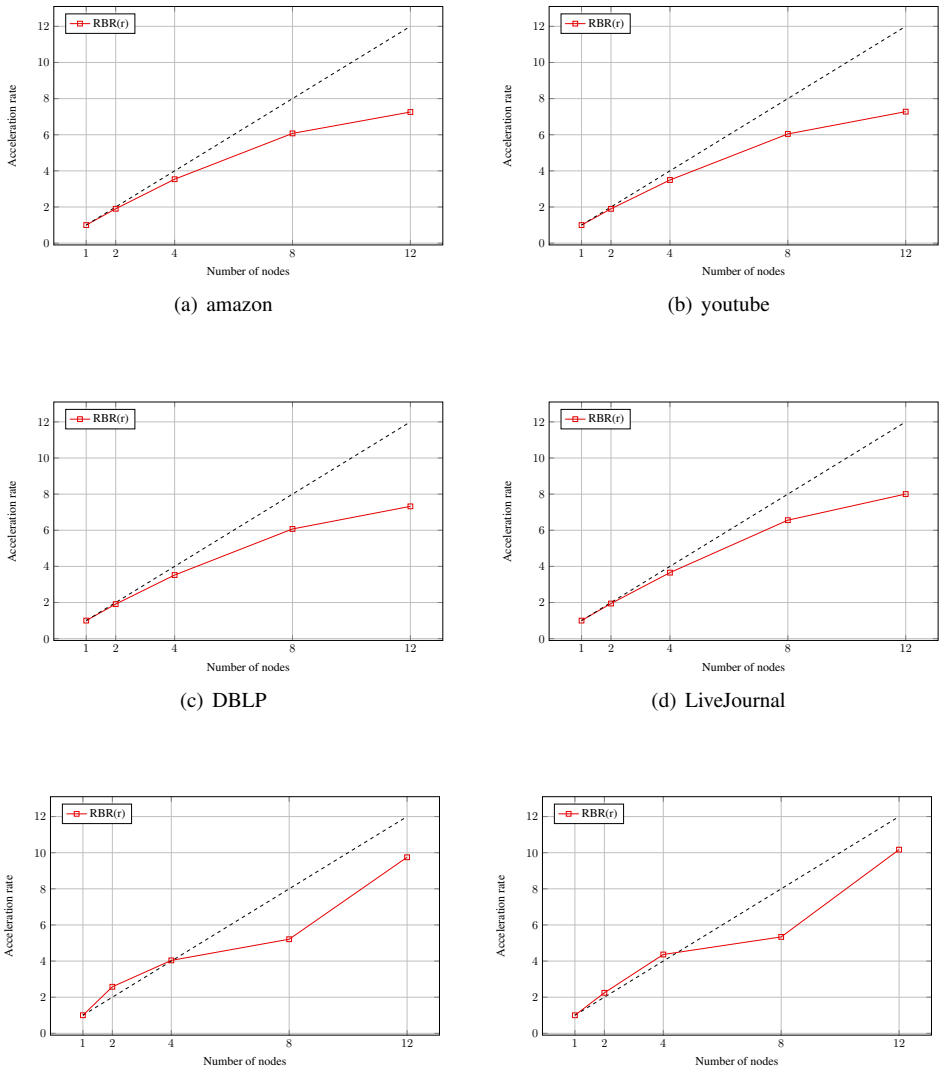
TABLE 6
Comparison between RBR and LOUVAIN algorithm under different settings

method	amazon					youtube				
	k_0	CC	S	Q	Time	k_0	CC	S	Q	Time
RBR(100,1)	100	0.426	0.500	0.755	3.5	100	0.589	0.480	0.623	12.5
RBR(20,5)	20	0.458	0.500	0.827	3.0	20	0.600	0.500	0.714	10.2
RBR(100,5)	100	0.467	0.500	0.895	5.0	98	0.614	0.490	0.710	22.6
LV(0)	230	0.531	0.598	0.926	1.7	28514	0.930	0.159	0.723	8.7
LV(500)	165	0.427	0.500	0.757	1.7	95	0.593	0.495	0.646	7.0
LV(2000)	183	0.435	0.500	0.775	1.7	93	0.587	0.495	0.658	6.0
method	DBLP					LiveJournal				
	k_0	CC	S	Q	Time	k_0	CC	S	Q	Time
RBR(100,1)	100	0.585	0.500	0.665	3.1	100	0.425	0.500	0.668	39.2
RBR(20,5)	20	0.672	0.500	0.754	2.9	20	0.448	0.500	0.737	43.4
RBR(100,5)	100	0.679	0.500	0.801	5.6	98	0.497	0.490	0.754	64.5
LV(0)	187	0.742	0.618	0.820	2.9	2088	0.729	0.877	0.749	181.5
LV(500)	146	0.583	0.500	0.673	2.1	101	0.433	0.500	0.665	81.9
LV(2000)	155	0.578	0.500	0.681	2.3	113	0.447	0.500	0.687	172.7
method	email-Enron					loc-Brightkite				
	k_0	CC	S	Q	Time	k_0	CC	S	Q	Time
RBR(100,1)	100	0.589	0.400	0.559	0.3	100	0.485	0.360	0.596	0.5
RBR(20,5)	20	0.708	0.500	0.605	0.3	20	0.516	0.500	0.667	0.5
RBR(100,5)	100	0.782	0.585	0.605	0.6	100	0.552	0.510	0.675	0.8
LV(0)	1245	0.935	0.974	0.597	0.2	732	0.829	0.939	0.687	0.3
LV(500)	45	0.705	0.600	0.581	0.2	47	0.513	0.500	0.637	0.3
LV(2000)	331	0.894	0.899	0.604	0.2	79	0.653	0.671	0.661	0.4
method	loc-Gowalla					Delaunay_n24				
	k_0	CC	S	Q	Time	k_0	CC	S	Q	Time
RBR(100,1)	100	0.439	0.470	0.637	1.7	100	0.298	0.500	0.730	190.7
RBR(20,5)	20	0.459	0.500	0.694	1.7	20	0.372	0.500	0.837	172.9
RBR(100,5)	98	0.497	0.475	0.696	4.3	100	0.365	0.500	0.868	252.5
LV(0)	850	0.592	0.761	0.705	1.2	354	0.430	0.500	0.990	189.1
LV(500)	60	0.442	0.500	0.658	1.3	500	0.287	0.500	0.717	792.7
LV(2000)	65	0.460	0.500	0.672	1.1	981	0.286	0.500	0.716	1390.5
method	europe_osm					road_usa				
	k_0	CC	S	Q	Time	k_0	CC	S	Q	Time
RBR(100,1)	100	0.042	0.500	0.790	689.9	100	0.216	0.500	0.736	274.3
RBR(20,5)	20	0.042	0.500	0.822	574.5	20	0.216	0.500	0.841	239.6
RBR(100,5)	100	0.042	0.500	0.894	717.4	100	0.216	0.500	0.908	297.4
LV(0)	3057	0.038	0.501	0.999	481.8	1569	0.225	0.510	0.998	303.2
LV(500)	500	0.041	0.500	0.663	243.7	500	0.216	0.500	0.663	153.6
LV(2000)	1999	0.041	0.500	0.663	337.3	1367	0.216	0.500	0.662	206.1

threaded version is not available. On the other hand, RBR(r) is an asynchronous parallel method. Although its theoretical property is not clear, it works fine in all of our experiments. Thus, RBR(r) is useful on multi-core machines. The speedup of RBR(r) is shown in Figure 8 when utilizing multiple computing threads. We can see that our RBR algorithm has almost a linear acceleration rate thanks to asynchronous parallelization. The average speedup is close to 8 using 12 threads. In particular, it reaches 10 times speedup on the loc-Brightkite network.

6. Conclusion. This paper is concerned with the community detection problem. We develop a sparse and low-rank relaxation of the modularity maximization model followed by either the K-means or weighted K-means clusterings or a direct rounding procedure. A fast algorithm called RBR with asynchronous parallellization is designed to efficiently solve the proposed problem. We provide some non-asymptotically bounds on the misclassification rate with high probability under some standard random network assumptions. Numerical experiments are fully consistent with the theoretical bounds. The new proposed method provides a competitive alternative state-of-the-art methods in terms of both misclassification rates and numerical efficiency, in addition to the assured theoretical bounds.

Acknowledgment: The authors thank Yizhou Wang for testing an early version of the algorithm on Small-Scaled Real World Data.



(a) amazon (b) youtube

(c) DBLP (d) LiveJournal

(e) email-Enron (f) loc-Brightkite

FIG. 8. *Acceleration rate of RBR(r) on large real data*

REFERENCES

[1] V. D. BLONDEL, J.-L. GUILLAUME, R. LAMBIOTTE, AND E. LEFEBVRE, *Fast unfolding of communities in large networks*, Journal of Statistical Mechanics: Theory and Experiment, 10 (2008), p. 10008.

[2] T TONY CAI, XIAODONG LI, ET AL., *Robust and computationally feasible community detection in the presence of arbitrary outlier nodes*, The Annals of Statistics, 43 (2015), pp. 1027–1059.

[3] KAMALIKA CHAUDHURI, FAN CHUNG GRAHAM, AND ALEXANDER TSIATAS, *Spectral clustering of graphs with general degrees in the extended planted partition model.*, in COLT, vol. 23, 2012, pp. 35–1.

[4] Y. CHEN, X. LI, AND J. XU, *Convexified modularity maximization for degree-corrected stochastic block models*, arXiv preprint arXiv:1512.08425, (2015).

[5] YUDONG CHEN, SUJAY SANGHAVI, AND HUAN XU, *Clustering sparse graphs*, in Advances in neural information processing systems, 2012, pp. 2204–2212.

[6] YUDONG CHEN AND JIAMING XU, *Statistical-computational phase transitions in planted models: The high-dimensional setting.*, in ICML, 2014, pp. 244–252.

[7] AMIN COJA-OGHLAN AND ANDRÉ LANKA, *Finding planted partitions in random graphs with general degree distributions*, SIAM Journal on Discrete Mathematics, 23 (2009), pp. 1682–1714.
[8] ANIRBAN DASGUPTA, JOHN E HOPCROFT, AND FRANK MCSHERRY, *Spectral analysis of random graphs with skewed degree distributions*, in Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on, IEEE, 2004, pp. 602–610.
[9] T. A. DAVIS AND Y. HU, *The university of florida sparse matrix collection*. http://www.cise.ufl.edu/research/sparse/matrices, 2011.
[10] CHRIS DING, XIAOFENG HE, AND HORST D SIMON, *On the equivalence of nonnegative matrix factorization and spectral clustering*, in Proceedings of the 2005 SIAM International Conference on Data Mining, SIAM, 2005, pp. 606–610.
[11] CHRIS DING, TAO LI, WEI PENG, AND HAESUN PARK, *Orthogonal nonnegative matrix t-factorizations for clustering*, in Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2006, pp. 126–135.
[12] OLIVIER GUÉDON AND ROMAN VERSHYNIN, *Community detection in sparse networks via grothendiecks inequality*, Probability Theory and Related Fields, 165 (2016), pp. 1025–1049.
[13] LENNART GULIKERS, MARC LELARGE, AND LAURENT MASSOULIÉ, *A spectral method for community detection in moderately-sparse degree-corrected stochastic block models*, arXiv preprint arXiv:1506.08621, (2015).
[14] PAUL W HOLLAND, KATHRYN BLACKMOND LASKEY, AND SAMUEL LEINHARDT, *Stochastic blockmodels: First steps*, Social networks, 5 (1983), pp. 109–137.
[15] ROGER A HORN AND CHARLES R JOHNSON, *Matrix analysis*, Cambridge university press, 2012.
[16] JIASHUN JIN ET AL., *Fast community detection by score*, The Annals of Statistics, 43 (2015), pp. 57–89.
[17] BRIAN KARRER AND MARK EJ NEWMAN, *Stochastic blockmodels and community structure in networks*, Physical Review E, 83 (2011), p. 016107.
[18] JINGU KIM AND HAESUN PARK, *Sparse nonnegative matrix factorization for clustering*, tech. report, Georgia Institute of Technology, 2008.
[19] DA KUANG, CHRIS DING, AND HAESUN PARK, *Symmetric nonnegative matrix factorization for graph clustering*, in Proceedings of the 2012 SIAM international conference on data mining, SIAM, 2012, pp. 106–117.
[20] JURE LESKOVEC AND ANDREJ KREVL, *SNAP Datasets: Stanford large network dataset collection*. http://snap.stanford.edu/data, jun 2014.
[21] FRANK MCSHERRY, *Spectral partitioning of random graphs*, in Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on, IEEE, 2001, pp. 529–537.
[22] MARK EJ NEWMAN, *Modularity and community structure in networks*, Proceedings of the national academy of sciences, 103 (2006), pp. 8577–8582.
[23] SAMET OYMAK AND BABAK HASSIBI, *Finding dense clusters via" low rank+ sparse" decomposition*, arXiv preprint arXiv:1104.5186, (2011).
[24] XINGHAO PAN, MAXIMILIAN LAM, STEPHEN TU, DIMITRIS PAPAILIOPOULOS, CE ZHANG, MICHAEL I JORDAN, KANNAN RAMCHANDRAN, CHRIS RE, AND BENJAMIN RECHT, *Cyclades: Conflict-free asynchronous machine learning*, arXiv preprint arXiv:1605.09721, (2016).
[25] SUBHADEEP PAUL AND YUGUO CHEN, *Orthogonal symmetric non-negative matrix factorization under the stochastic block model*, arXiv preprint arXiv:1605.05349, (2016).
[26] IOANNIS PSORAKIS, STEPHEN ROBERTS, MARK EBDEN, AND BEN SHELDON, *Overlapping community detection using bayesian non-negative matrix factorization*, Physical Review E, 83 (2011), p. 066114.
[27] TAI QIN AND KARL ROHE, *Regularized spectral clustering under the degree-corrected stochastic blockmodel*, in Advances in Neural Information Processing Systems, 2013, pp. 3120–3128.
[28] BENJAMIN RECHT, CHRISTOPHER RE, STEPHEN WRIGHT, AND FENG NIU, *Hogwild: A lock-free approach to parallelizing stochastic gradient descent*, in Advances in Neural Information Processing Systems, 2011, pp. 693–701.
[29] KARL ROHE, SOURAV CHATTERJEE, AND BIN YU, *Spectral clustering and the high-dimensional stochastic blockmodel*, The Annals of Statistics, (2011), pp. 1878–1915.
[30] DANIEL L SUSSMAN, MINH TANG, DONNIELL E FISHKIND, AND CAREY E PRIEBE, *A consistent adjacency spectral embedding for stochastic blockmodel graphs*, Journal of the American Statistical Association, 107 (2012), pp. 1119–1128.
[31] A. L. TRAUD, E. D. KELSIC, P. J. MUCHA, AND M. A. PORTER, *Comparing Community Structure to Characteristics in Online Collegiate Social Networks*, ArXiv e-prints, (2008).
[32] FEI WANG, TAO LI, XIN WANG, SHENGHUO ZHU, AND CHRIS DING, *Community discovery using nonnegative matrix factorization*, Data Mining and Knowledge Discovery, 22 (2011), pp. 493–521.
[33] MENG WANG, CHAOKUN WANG, JEFFREY XU YU, AND JUN ZHANG, *Community detection in social networks: an in-depth benchmarking study with a procedure-oriented framework*, Proceedings of the VLDB Endowment, 8 (2015), pp. 998–1009.
[34] ZHIRONG YANG, TELE HAO, ONUR DIKMEN, XI CHEN, AND ERKKI OJA, *Clustering by nonnegative*

matrix factorization using graph random walk, in Advances in Neural Information Processing Systems, 2012, pp. 1079–1087.

[35] YUAN ZHANG, ELIZAVETA LEVINA, AND JI ZHU, *Detecting overlapping communities in networks using spectral methods*, arXiv preprint arXiv:1412.3432, (2014).