Scalable inference for stochastic block models

Chengbin Peng[1,2], Zhihua Zhang[3], Ka-Chun Wong[4],
Xiangliang Zhang[1,*], David E. Keyes[1]
[1] King Abdullah University of Science and Technology,
Thuwal, Saudi Arabia
[2] Ningbo Institute of Industrial Technology, Ningbo, China
[3] Shanghai Jiao Tong University, Shanghai, China
[4] City University of Hong Kong, Hong Kong, China
E-mail: chengbin.peng@kaust.edu.sa, zhang-zh@cs.sjtu.edu.cn,
kc.w@cityu.edu.hk, xiangliang.zhang@kaust.edu.sa,
david.keyes@kaust.edu.sa

December 8, 2017

Abstract

Community detection in graphs is widely used in social and biological networks, and the stochastic block model is a powerful probabilistic tool for describing graphs with community structures. However, in the era of "big data," traditional inference algorithms for such a model are increasingly limited due to their high time complexity and poor scalability. In this paper, we propose a multi-stage maximum likelihood approach to recover the latent parameters of the stochastic block model, in time linear with respect to the number of edges. We also propose a parallel algorithm based on message passing. Our algorithm can overlap communication and computation, providing speedup without compromising accuracy as the number of processors grows. For example, to process a real-world graph with about 1.3 million nodes and 10 million edges, our algorithm requires about 6 seconds on 64 cores of a contemporary commodity Linux cluster. Experiments demonstrate that the algorithm can produce high quality results on both benchmark and real-world graphs. An example of finding more meaningful communities is illustrated consequently in comparison with a popular modularity maximization algorithm.

Keywords: stochastic block model, parallel computing, community detection

*Corresponding author. Phone: +966-12-808-0313. Postal address: Post Box. 2925, King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Kingdom of Saudi Arabia

1 Introduction

Community structures, in which nodes belonging to the same community are more densely connected to each other than externally, prevail in real graphs. Finding those structures can be beneficial in many fields, such as finding protein complexes in biological networks and topical, disciplinary groups in collaborative networks, etc [1, 2, 3, 4, 5, 6, 7].

In this work, we consider non-overlapping community structures. Many community detection algorithms handle such a problem [8, 9, 10, 3, 11, 12, 13, 14]. However, they come along with limitations for large graphs, for example, in handling community heterogeneity [15, 16, 17].

Alternatively, model-based methods can produce more reliable and accurate results when the model assumptions are in accordance with real graphs. Stochastic block models (SBMs) are among the important probabilistic tools describing the connectivity relationship between pairs of nodes [18], and have received considerable attention both in theoretical [19] and application domains [20].

Many algorithms have been proposed to infer the parameters of SBMs, such as Bayesian estimation [21] and nuclear norm minimization [22]. Bayesian estimation defines the prior distributions on latent variables (community labels of nodes) and maximizes the posterior distribution when a graph is given [21]. Nuclear norm minimization solves a relaxed problem to find the community indication matrix.

We can also utilize parallel computers to detect communities on large graphs. Parallel computing has been proposed for decades [23][24]. It can decrease the execution time of applications by executing them on multiple processors [25].

Parallel computers are classified according to the programming model natural to their memory structure, whether shared, or distributed, or hybrid. In a shared memory model [26], each processor can access the entire system memory, while each processor in a distributed memory model [27] can access only its local memory and requires an interprocessor network to communicate data to and from other local memories. In the hybrid model [28], subsets of processors share memory and these subsets are connected by a communication network. The hardware supporting these programming models need not fully resemble them. The convenient global addressing of shared memory can be implemented over distributed memory hardware and the efficient controlled access of distributed memory over shared memory hardware. However, there is a general association of distributed memory protocols with the systems of the largest memory capacities. Therefore, in our algorithm, we choose the message passing model as the most performance-portable to the hardware resources that will be required for graph applications of the future.

In this work, we propose a multi-stage likelihood maximization algorithm based on SBMs, which has three advantages:

- **Speed**: We devise an algorithm based on coordinate descent and use approximations to simplify computations. Our algorithm runs in linear

time with respect to the number of edges.

- **Scalability**: We propose a parallel algorithm based on message-passing, which tends to be the most portable and performance-consistent parallel programming model for a variety of memory structures. We overlap the communication and computation in the algorithm, so that for large graphs, it can achieve significant speedup proportional to the number of processors. To the best of our knowledge, it is the first parallel algorithm for inferring SBMs.

- **Quality**:

  The algorithm can produce high-quality results. In the initialization, it considers each node as one community and then employs a multi-stage iterative strategy to construct larger communities gradually. It outperforms traditional community detection algorithms empirically as measured by normalized mutual information (NMI) [29] with respect to the ground-truth. Experiments on real-world graphs show that our algorithm can produce more meaningful communities with good quality.

2 Related Work

Many community detection algorithms have been proposed in the past decades. Some algorithms rely on a quality function, for example, modularity. Modularity is the fraction of inter-community connections minus the expected fraction when edges are randomly placed [30]. Newman [8] optimizes the modularity in a hierarchical manner. A typical spectral clustering approach [31] can also detect communities. Li and Schuurmans [32] use an iterative rounding strategy for the spectral method to improve the quality of the results. The Louvain method [10] improves the modularity hierarchically. Nevertheless, they may have some disadvantages. For example, modularity-based algorithms may have resolution limit [16], which means they may be unable to find small communities without modifying their objective function [33]. In some cases, other algorithms can give high quality results when maximizing the modularity, but the time complexity could be high. For example, spectral clustering has a typical time complexity no less than $O(\mathbf{NK}^2)$ [34], where $\mathbf{N}$ is the number of nodes and $\mathbf{K}$ is the number of communities. Zhang and Newman [35] developed a multiway spectral algorithm that can divide graphs into many communities and achieves good quality in terms of modularity, but the time complexity is still high. Liu et al. adopted spectral clustering on a reduced graph, and the run time is quadratic to the size of the reduced graph [11]. Block modularity maximization [36] although reduces the run time from cubic to quadratic in terms of the number of clusters, is still expensive and not suitable for large graphs with many communities.

On the other hand, using SBMs is more reliable under particular assumptions. Theoretically, the threshold of exact community recovery can been identified [37]. In practice, many algorithms have been proposed to infer the parameters

of SBMs. Bayesian estimation defines the prior distributions on latent variables (community labels of nodes) and maximizes the posterior distribution when a graph is given [21]. Nuclear norm minimization of a matrix minimizes the sum of its singular values, and their algorithm is verified on graphs containing one thousand nodes [22]. Spectral algorithms also demonstrate the capability of inferencing parameters of SBMs [38]. However due to the high time complexity, the run time of those algorithms typically becomes enormous when the graph size grows to millions or billions, which prohibits the application on big graphs.

Recent years have seen work on parallel implementations for non-SBM community detection algorithm. For example, Riedy et al. proposed an parallel agglomerative approach for community detection that supports both maximizing modularity and minimizing conductance [39]. Bhowmick and Srinivasan used a shared memory interface for parallelizing the Louvain method [40] . Staudt and Meyerhenke developed a parallel framework for multiple algorithms, including label propagation and the Louvain method with refinement [13]. Gregori et al. devised an parallel algorithm for k-clique community detection [41]. Parallel algorithms for spectral clustering are also proposed [42]. Those implementations are mostly on shared-memory systems, which constrain the maximum speedup due to the hardware limit. In addition, none of these implementations can suitably address the problem in SBMs.

In this work, we propose a community shrinking and expanding approach to overcome the limitation of the popular coordinate descent method for stochastic blockmodels. It is also scalable on parallel computers. Our fast and scalable algorithm fills the gap of processing very large graphs using stochastic block models.

3 Stochastic Block Model

In this work, we develop a community detection algorithm based on a stochastic block model (SBM). We define N as the number of nodes and M as the number of undirected and unweighted edges connecting those nodes. Each node belongs to one of K blocks, and we use $Z \in \{0,1\}^{N\times K}$ to represent the block labels. That is, $Z_{ir} = 1$ means node i belongs to block r and each row of Z contains only one nonzero entry. We also define a matrix $B \in [0,1]^{K\times K}$ where $B_{rk}(r \neq k)$ represents the probability of connections between nodes drawn from block r and k, respectively. If $r = k$, B_{rk} represents the probability of connections inside the block.

Using the matrices B and Z, we define a probability matrix $\Theta = ZBZ^T$. Then the adjacency matrix W of a sample network can be drawn from the following model:

$$\Pr(W_{ij}) = \begin{cases} \Theta_{ij} & \text{if } W_{ij} = 1, \\ 1 - \Theta_{ij} & \text{if } W_{ij} = 0, \end{cases} \tag{1}$$

for $i, j \in \{1, 2, \cdots, N\}$ and $i \neq j$, indicating that W_{ij} is a sample from the

Bernoulli distribution with success rate Θ_{ij}.

For example, we can define a probability matrix

$$B = \begin{bmatrix} 0.2 & 0.01 \\ 0.01 & 0.2 \end{bmatrix}_{2\times 2},$$

meaning that nodes inside each community are connected at probability 0.2 and nodes from different communities are connected at probability 0.01. We also define a matrix

$$Z = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ \vdots & \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ \vdots & \\ 0 & 1 \end{bmatrix}_{100\times 2}, \tag{2}$$

indicating the first half and the second half of the nodes are from two different groups. Thus, we can have the probability matrix

$$\Theta = \begin{bmatrix} 0.2 & \dots & 0.2 & 0.01 & \dots & 0.01 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0.2 & \dots & 0.2 & 0.01 & \dots & 0.01 \\ 0.01 & \dots & 0.01 & 0.2 & \dots & 0.2 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0.01 & \dots & 0.01 & 0.2 & \dots & 0.2 \end{bmatrix}_{100\times 100}. \tag{3}$$

In this case, a possible adjacency matrix W can be generated from Θ, by considering each edge indicator W_{ij} (for the edge between node i and j) as a binomial random variable with parameter Θ_{ij}. We can use an image to represent the matrix W, in which non-zero entries (corresponding to edges) are plotted in blue, as shown in Fig. 1. The corresponding matrix W here is of size 100×100 with 1042 non-zero entries.

Typically in practice, the adjacency matrix W is available from the data set. Our primary purpose is to estimate Z.

4 Model Analysis: An Ideal Example

In this section, through a simplified example, we illustrate how our algorithm solves the SBM problem. In particular, we assume the diagonal part of the ground truth B is p, and the off-diagonal part is q. Thus, each pair of nodes from one community is connected with probability p, and those from two different

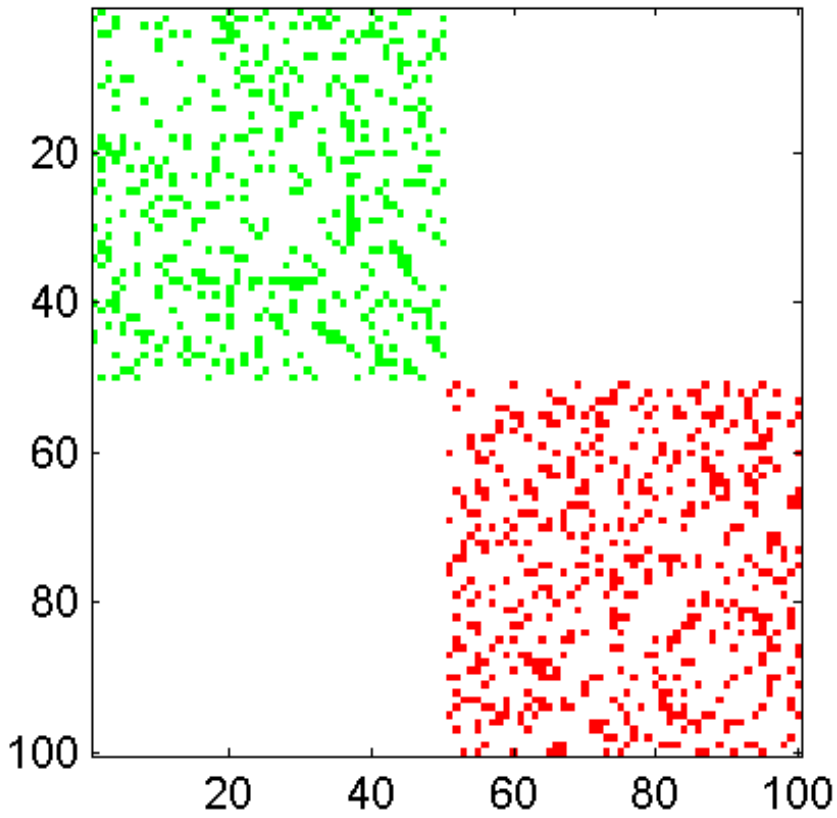


Figure 1: One Adjacency Matrix Generated by the Model

communities are connected with probability q. As mentioned earlier, N is the number of nodes in the graph, and K is the number of communities. We define n as the average number of nodes in each community.

4.1 Self-Assembling Process

We define the community that a subset of nodes indeed belongs to as the *true community* for those nodes, and the community determined by an algorithm at each iteration as the *temporary community*.

We initialize Z by putting nodes almost evenly and randomly into different temporary communities. Thus, we can define $p_R^{(0)} \approx \frac{1}{K^2}$ as the expected fraction of nodes belonging to the same true community over the total number of nodes, in a temporary community at time 0. Due to the randomness, there is very likely

a small bias that some temporary community should have a larger number of nodes from the same true community.

At each iteration, each node will associate with a particular temporary community where it has a tight connection. In this case, the temporary community with more nodes from the same true community will take an advantage because the probability of the intra-community connection is higher than other temporary communities. Due to this bias, the association will result in

$$\begin{aligned} p_R^{(t)} &= \frac{p_R^{(t-1)}}{\frac{1}{K}} \times (\frac{1}{K} - p_R^{(t-1)}) + p_R^{(t-1)} \\ &= 2p_R^{(t-1)} - (p_R^{(t-1)})^2 \times K \end{aligned} \tag{4}$$

for the fraction of nodes in a temporary community and from the same true community. In Eq. (4), $\frac{1}{K} - p_R^{(t-1)}$ is the fraction of nodes that has not yet been correctly clustered, and $\frac{p_R^{(t-1)}}{\frac{1}{K}}$ is the fraction of wrongly clustered nodes that can be fixed in the tth iteration. As the maximum $p_R^{(t)}$ is $\frac{1}{K}$, the condition $2 > p_R^{(t-1)}K$ is always true. Therefore, this equation indicates at the early stage, nodes from the same true community in other temporary communities will tend to assemble to a one temporary community.

In practice, we generate many more temporary communities than the number of true communities at the initialization, to avoid two or more true communities assembling at the same temporary community.

4.2 Minimum Gap between p and q

The gap between p and q represents the difference in edge densities within and across communities. A smaller value of $p-q$ in SBMs causes more challenges in the identification of the true Z. Thus, a parametric inference algorithm is more powerful if it allows a smaller gap between p and q. In this section, we try to find the lower bound of $p - q$.

Let $\Pr(Z_{ik} = 1)$ be the probability that node i of true community k can be correctly assigned via maximum likelihood. We define the destination temporary community as the temporary community that node i will be assigned to after the current iteration. When correctly assigned, the destination temporary community should have more nodes connected to i than any other temporary community. We define X and Y as collections of binary random variables, and $X_j = 1$ means node i has a connection to node j in set X. We use $\sum_j X_j$ and $\sum_j Y_j$ to represent the number of nodes connected to i in the destination temporary community and one other temporary community respectively.

Then we will demonstrate that node i can be correctly identified at least in certain probability p_I, where p_I is the probability for correct identification between the true community and all the other communities. Because there are $K - 1$ communities other than the true one, then, the probability of correct identification from the true community and one of the other communities is

$p_I^{\frac{1}{K-1}}$.

Using an auxiliary variable d to represent the cutting point of $\sum_j X_j$ and $\sum_j Y_j$, we have

$$\begin{aligned}
& p_I^{\frac{1}{K-1}} \\
\geq\ & \Pr(\sum_j X_j > d) \times \Pr(\sum_j Y_j < d) \\
=\ & \big[1 - \Pr(\sum_j X_j < d)\big]\big[1 - \Pr(\sum_j Y_j > d)\big] \\
=\ & \Big[1 - \Pr(\sum_j X_j < pn - (pn - d))\Big] \\
& \times \Big[1 - \Pr(\sum_j Y_j > qn + (-qn + d))\Big] \\
\geq\ & \Big[1 - e^{-\frac{(pn-d)^2}{2pn}}\Big]\Big[1 - e^{-\frac{2(-qn+d)^2}{n}}\Big] \qquad (5) \\
=\ & \gamma^{\frac{1}{K-1}},
\end{aligned}$$

where n is the expected number of nodes within the community and γ is a lower bound of probability p_I. The last inequality is obtained from Chernoff's inequality [43]

$$Pr(\sum_j X_j < pn + \delta) \leq e^{-\frac{\delta^2}{2pn}}$$

and Hoeffding's inequality

$$Pr(\sum_j Y_j > qn + \delta) \leq e^{-\frac{2\delta^2}{n}}$$

for Bernoulli random variables.

By choosing an appropriate d, the lower bound of $p_I^{\frac{1}{K-1}}$ can be well approximated. If we choose d to have the two factors in Eq. (5) equal, we obtain the

following expressions

$$\begin{aligned} pn-d &= 2\sqrt{p}(d-qn), \\ d &= \frac{pn+2\sqrt{p}(qn)}{1+2\sqrt{p}}, \\ pn-d &= pn-\frac{pn+2\sqrt{p}(qn)}{1+2\sqrt{p}} \\ &= \frac{pn+2\sqrt{p}(pn)-pn-2\sqrt{p}(qn)}{1+2\sqrt{p}} \\ &= \frac{2\sqrt{p}(pn-qn)}{1+2\sqrt{p}}. \end{aligned}$$

Hence,

$$\frac{(pn-d)^2}{2pn} = \frac{\frac{4p(pn-qn)^2}{(1+2\sqrt{p})^2}}{2pn} = \frac{2n(p-q)^2}{(1+2\sqrt{p})^2}. \tag{6}$$

Because two factors in Eq. (5) are the same, we have

$$\begin{aligned} & p_I^{\frac{1}{K-1}} \\ \geq & \left[1-e^{-\frac{2n(p-q)^2}{(1+2\sqrt{p})^2}}\right]^2 \qquad (7) \\ = & \gamma^{\frac{1}{K-1}}. \qquad (8) \end{aligned}$$

As the lower bound of the probability p_I, γ is expected to be as large as possible. Then we can have $1-\gamma \leq 1-c_0N^{-10}$ for some constant c_0 to meet the definition of high probability, where N is the total number of nodes. According to Eq. (8), we have

$$\begin{aligned} \frac{2n(p-q)^2}{(1+2\sqrt{p})^2} =& -\log(1-\gamma^{\frac{1}{2(K-1)}}) \\ \geq& -\log[1-(c_0N^{-10})^{\frac{1}{2(K-1)}}] \\ =& (c_0N^{-10})^{\frac{1}{2(K-1)}} + \frac{1}{2}(c_0N^{-10})^{\frac{2}{2(K-1)}} \\ & + \frac{1}{3}(c_0N^{-10})^{\frac{3}{2(K-1)}} + \cdots \\ =& \Omega(N^{-\frac{5}{K-1}}), \end{aligned}$$

where the expansion is based on the Taylor series.

Therefore, as K is no less than two and $n \approx \frac{N}{K}$, we can have the following

approximation

$$\begin{aligned} p-q =& \sqrt{(1+2\sqrt{p})^2\Omega(N^{[-\frac{5}{K-1}-1]}K)} \\ \approx& \Omega\Big(\sqrt{\frac{pK}{N}}\Big), \end{aligned} \tag{9}$$

Thus, the lower bound of $p-q$ by our algorithm can be as small as $\Omega\Big(\sqrt{\frac{pK}{N}}\Big)$. Compared to many other methods summarized in [22], the lower bound by our approach is relatively small.

4.3 Graphs with Heterogeneous Partitions

It is possible that the sizes of the partitions are different, and we analyze the consequent self-assembling process under this assumptions.

First, we define $N_1, N_2, ..., N_{K-1}, N_K$ in ascending order as the size of the K communities. Without loss of generality, we assume the kth partition corresponding to N_k will emerge in the kth temporary community.

Then, when temporary communities are initialized to have the same size, nodes from different communities are assigned to temporary communities in equal probability. To have the self-assembling process still work, we shall have the following expression, so that nodes from community 1 will not assemble to temporary community K.

$$\frac{qN_K}{K} < \frac{pN_1}{K}, \tag{10}$$

where $k \in \{1, 2, ..., K\}$. Under the above condition, nodes from other communities are expected to assemble to correct temporary communities because of the following inequality.

$$\frac{qN_k}{K} < \frac{qN_K}{K} < \frac{pN_1}{K} < \frac{pN_k}{K}. \tag{11}$$

During the self-assembling procedure, similar to Eq. (4), we have the following expression for community k

$$\begin{aligned} p_R^{(t)} =& \quad \frac{p_R^{(t-1)}}{\frac{N_k}{N}} \times (\frac{N_k}{N} - p_R^{(t-1)}) + p_R^{(t-1)} \\ =& \quad 2p_R^{(t-1)} - (p_R^{(t-1)})^2 \times \frac{N}{N_k^{(t-1)}}. \end{aligned}$$

Hence, the convergence rate is still quadratic.

At the end of the iteration, to obtain the correct result, the following expression is expected

$$qN_K < pN_1, \tag{12}$$

which is in accordance with the assembling condition at the initialization in Eq.

(10).

For undirected networks, B is symmetric. When the probability matrix contains a variety of values, rather than only p and q, the following inequality is required as a necessary condition for converging to the correct result.

$$B_{ij}N_i < B_{jj}N_j, \quad \text{for} \quad i > j. \tag{13}$$

5 Methodology

For a specific model, the likelihood is a function of the model parameters, describing the probability of obtaining the observed data with these parameters. The maximum likelihood estimator is the setting of parameters that maximizes the likelihood function.

As defined in Eq. (1), if only W is given, the log-likelihood function is

$$\begin{aligned} L(B, Z|W) &= \sum_{i \neq j} \log \Pr(W_{ij}) \\ &= \sum_{i \neq j} \log[(1 - W_{ij}) + (2W_{ij} - 1)\Theta_{ij}]. \end{aligned}$$

It is very time consuming to maximize such a likelihood function directly through traditional optimization methods (for example, branch-and-bound) for large graphs in which there are at least NK unknown variables.

For the sake of speed and scalability, we propose a fast algorithm that updates B and Z in turn to maximize the objective function $L(B, Z|W)$, and use a multi-stage framework to help the solution be close to the global optimum. We also develop a parallel implementation to make the model more scalable.

5.1 Estimation Algorithm

Given W, the maximum likelihood estimates of (B, Z) are defined as

$$\begin{aligned} &\underset{B,Z}{\operatorname{argmax}} \Big\{ L(B, Z|W) \\ &= \sum_{i \neq j} \log[(1-W_{ij}) + (2W_{ij}-1)\Theta_{ij}] \\ &= \sum_{i \neq j} \log[(1-W_{ij}) + (2W_{ij}-1)(ZBZ^T)_{ij}] \Big\}, \end{aligned} \tag{14}$$

subject to $0 \leq B_{ij} \leq 1$, $Z_{ij} \in \{0, 1\}$, $\sum_j Z_{ij} = 1$. We solve the above optimization problem by alternatively updating B and Z and using a community shrinking and expanding strategy to improve accuracy.

We first describe the alternating updating procedure. When Z is fixed and B is considered as unknown, without loss of generality, let $\beta = B_{rk}$. If other

entries are fixed, $L(B, Z|W) = s\log(\beta) + \hat{s}\log(1-\beta) + C$, where C is a constant, $s = \sum_{ij} W_{ij}(Z_{:r}Z_{:k}^T)_{ij}$ and $\hat{s} = \sum_{ij}(1 - W_{ij})(Z_{:r}Z_{:k}^T)_{ij}$.

Taking the derivative of L with respect to β,

$$\frac{\partial L}{\partial \beta} = \frac{s}{\beta} - \frac{\hat{s}}{1-\beta} \tag{15}$$

and setting the derivative to be zero, we have

$$\beta = \frac{s}{s + \hat{s}}. \tag{16}$$

As the inter-block connection probabilities are small, we use a representative scalar value to replace the off-diagonal entries of B, which can be computed by counting all the inter-community edges.

Thus, the total time complexity of updating B is $\mathcal{O}(N) + \mathcal{O}(K) + \mathcal{O}(M) = \mathcal{O}(M)$.

Theorem 5.1 *For fixed Z, the objective function $L(B, Z|W)$ achieves its global maximum if entries of B are updated according to Eq. (16).*

Proof When Z is fixed, because each entry of B can optimize the objective function independently, after updating all the entries by Eq. (16), the resulting B is a stationary point of the objective function and each entry satisfies the constraints.

By taking the second derivative of the objective function, we have

$$\left.\frac{\partial^2 L}{\partial \beta^2}\right|_{\beta=\frac{s}{s+\hat{s}}} = \left. -\frac{s}{\beta^2} - \frac{\hat{s}}{(1-\beta)^2}\right|_{\beta=\frac{s}{s+\hat{s}}} < 0. \tag{17}$$

Therefore, when Z is given, the objective function at β (determined by Eq. (16)) is a global maximum. As each entry of B is irrelevant to each other, we can update B by Eq. (16) sequentially.

When B is fixed, we use the block coordinate descent method to update Z row by row. When updating the first Z in ZBZ^T in Eq. (14), the algorithm keeps the second Z as its previous estimate $Z^{(t-1)}$. Then, the likelihood function can be locally maximized by setting all the elements in the row to 0 but $Z_{ir_{\max}} = 1$, where $r_{\max}$ is chosen by

$$\begin{aligned} r_{\max} = \operatorname*{argmax}_r \sum_{j\neq i} \log \big[&(1 - W_{ij}) \\ &+ (2W_{ij} - 1)(B[Z^{(t-1)}]^T)_{rj}\big]. \end{aligned} \tag{18}$$

The time complexity for updating one node by Eq. (18) is $\mathcal{O}(NK)$ and for all the nodes it is $\mathcal{O}(N^2K)$.

To reduce the time complexity, we propose a faster method. Let $N^{(c)}$ be a column vector containing the number of nodes in each community, which is

updated once the row $Z_{i:}$ is changed. Let $N^{(d)}$ be a column vector containing the number of nodes connected to node i in each community. Thus, we have a new update rule:

$$r_{\max} = \operatorname*{argmax}_r \sum_{k=1}^{K} \big[N_k^{(d)} \log(B_{rk}) + (N_k^{(c)} - N_k^{(d)}) \log(1 - B_{rk})\big]. \tag{19}$$

The time complexity is $\mathcal{O}(M_i) + \mathcal{O}(K^2)$, where M_i is the degree of node i used in computing $N^{(d)}$. If we use a scalar to represent the inter-block connection probabilities as before, we can replace the summation in Eq. (19) by two multiplications. We also note that a node will take only a block label from its neighbors, and the time to enumerate over different choices of r becomes $\mathcal{O}(M_i)$ for node i in Eq. (19). Thus, the time complexity of computing labels for all the node is reduced to $\sum_{i=1}^{N} \mathcal{O}(M_i) = \mathcal{O}(M)$.

For fixed B, we define $G(B, Z^{(2)}, Z^{(1)}) = \sum_{i \neq j} \log[(1 - W_{ij}) + (2W_{ij} - 1)(Z^{(2)} B {Z^{(1)}}^T)_{ij}]$.

Therefore, the objective function can be rewritten as $L(B, Z|W) := G(B, Z, Z)$.

If entries in Z are continuous variables, then the above optimization can find a global maximum.

Theorem 5.2 *Optimization* $\min_{Z^{(t)}} -G(B, Z^{(t)}, Z^{(t-1)})$ *under the constraints* $\sum_r Z_{ir} = 1$ *and* $0 \leq Z_{ir} \leq 1$ *is a convex optimization problem when* B *and* $Z^{(t-1)}$ *are given.*

Proof First we reshape the matrix Z into a column vector z where Z_{ij} corresponds to $z_{K(i-1)+j}$.

Let $H \in \mathbb{R}^{NK \times NK}$ be the Hessian matrix of the objective function $G(B, Z^{(t)}, Z^{(t-1)})$ with respect to $z^{(t)}$. Let $M = \log[(1 - W) + (2W - 1) \odot Z^{(t)} B {Z^{(t-1)}}^T]$, which is the matrix in $G(B, Z^{(t)}, Z^{(t-1)})$ before the final summation, and we only need to consider the off-diagonal part of the matrix in this prove. Here $\odot$ denotes the element-wise product of two matrices.

If $[Z_{i:}^{(t-1)}]^T$ are constants, the second derivative of the (i, j)th term $(i \neq j)$ in M with respect to variables other than $[Z_{i:}^{(t)}]^T$ is 0, while with respect to the variables in $[Z_{i:}^{(t)}]^T$, it is as follows

$$
\begin{aligned}
&\frac{\partial^2 - M_{ij}}{(\partial[Z_{i:}^{(t)}]^T)^2} \\
=&\frac{\partial^2 - \log[(1-W_{ij}) + (2W_{ij}-1)Z_{i:}^{(t)}B[Z_{j:}^{(t-1)}]^T]}{(\partial[Z_{i:}^{(t)}]^T)^2} \\
=&\frac{\partial - \frac{(2W_{ij}-1)B[Z_{j:}^{(t-1)}]^T}{(1-W_{ij})+(2W_{ij}-1)Z_{i:}^{(t)}B[Z_{j:}^{(t-1)}]^T}}{\partial[Z_{i:}^{(t)}]^T} \\
=&\frac{[(2W_{ij}-1)]^2 B[Z_{j:}^{(t-1)}]^T[Z_{j:}^{(t-1)}]B^T}{[(1-W_{ij}) + (2W_{ij}-1)Z_{i:}^{(t)}B[Z_{j:}^{(t-1)}]^T]^2},
\end{aligned}
$$

where the square operation is element-wise. It is positive semi-definite because it is a covariance matrix multiplied with a positive scalar.

Therefore, for any column vector x, we have

$$
\begin{aligned}
-x^T H x =& -x^T \frac{\partial^2 \sum_{i\neq j} M_{ij}}{\partial z^2} x \\
=& \sum_{i\neq j} x^T \frac{\partial^2 - M_{ij}}{\partial z^2} x \\
\geq& 0.
\end{aligned} \tag{20}
$$

Hence, the objective function is convex. As obviously the constraints are convex as well, this problem is a convex optimization.

However, in our model, Z is a Boolean matrix. Thus, it will probably converge to a local optimum, in which nodes from several true communities may become one single temporary community. The local optimum problem can be avoided by limiting each temporary community to contain only one true community in the initialization. The next subsection describes the detail.

5.1.1 Community Shrinking and Expanding

A community shrinking and expanding approach works as follows. First, randomly initialize the nodes into αK temporary communities where α should be large enough so that the community size is tiny. Second, run the inference algorithm while gradually merge communities and reduce the number of communities to K. When K is unknown, the algorithm proceeds until no more merging is possible.

This approach reduces the "collision" probability in the initialization significantly, where a "collision" is the situation that two true communities both have most of the nodes in one temporary community. The fraction of the "non-

collision" probability after shrinking to the probability without shrinking is:

$$\prod_{k=0}^{K-1} \frac{1-\frac{k}{\alpha K}}{1-\frac{k}{K}} = \prod_{k=0}^{K-1} [1 + \frac{(1-\frac{1}{\alpha})k}{K-k}] \geq (1-\frac{1}{\alpha})\frac{K^K}{K!}.$$

The lower bound is not sensitive to the choice of α when α is large enough. For example, the lower bounds at $\alpha = 10$ and $\alpha = 100$ are $\frac{0.9K^K}{K!}$ and $\frac{0.99K^K}{K!}$, respectively, both of which are very significant. In practice, we can choose $\alpha K = N$.

Initializing an extra number of communities may introduce some tiny communities in the final result. The community expanding is devised by considering the likelihood of the inter-community edges. When they are more likely to be in certain communities, the corresponding nodes are merged. In detail, for any two communities r and k, the number of edges between them is

$$c_{rk} = \sum_{i \neq j} Z_{ir} W_{ij} Z_{jk},$$

which is out of $n_{rk} = \sum_i Z_{ir} \times \sum_i Z_{ik}$, the maximum possible connections. Therefore, the log likelihood of the inter-community edges c_{rk} belonging to the true community r can be represented by L_p, while the likelihood of not belonging to the community is L_q.

$$L_p = c_{rk} \log(p) + (n_{rk} - c_{rk}) \log(1-p), \tag{21}$$
$$L_q = c_{rk} \log(q) + (n_{rk} - c_{rk}) \log(1-q), \tag{22}$$

where $p = (c_{rr} + c_{rk} + c_{kr} + c_{kk})/(n_{rr} + n_{rk} + n_{kr} + n_{kk})$ indicating the internal density after merging, and $q = B_{rk}$. If $L_p > L_q$, we merge r and k into one community, otherwise leave them separated.

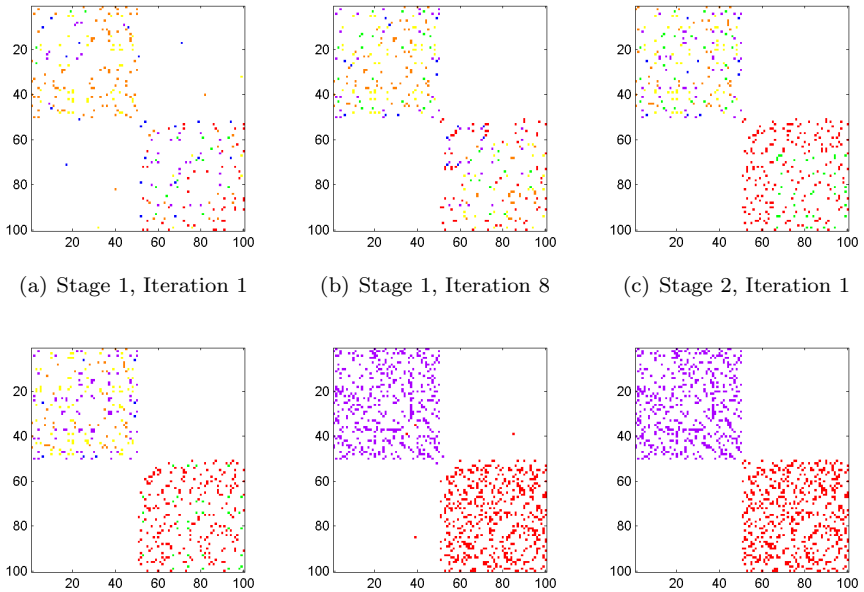
The time complexity for each merging is $\mathcal{O}(K^2)$. However, if we only consider the pairs of communities that have at least one edge between them, the time complexity becomes $\mathcal{O}(M)$. Community merging runs at the end of each stage after updating Z and B. Algorithm 1 is the pseudocode of the serial algorithm.

In Fig. 2, we use the graph in Fig. 1 as an example to illustrate the process. When two nodes at both ends of an edge belong to the same community, we assign the edge with a specific color corresponding to the community. Otherwise, the edge is in background color (white) and invisible.

At the beginning, we shrink the temporary communities as initialization, so at Iteration 1 of Stage 1, there are many small communities (temporary communities) and the edge colors are very diverse. At the end of Stage 1, due to the self assembling process, some temporary communities become a bit larger. After each stage, we expand the communities by merging temporary communities that very likely belong to the same community. Therefore, after the transition of each stage, we can see much larger communities which are slightly refined in the stage. Eventually, the algorithm converges and the two true communities are identified.

Algorithm 1 Serial Algorithm (ML-SBM)

set community number to αK
initialize B, $Z^{(t)}$ into many tiny communities
$Z^{(t-1)} = Z^{(t)}$
repeat
  repeat
    for $i = 1 : N$ **do**
      compute $N^{(d)}$ for node i
      update $Z_{i:}^{(t)}$ according to Eq. (19)
      $Z_{i:}^{(t-1)} = Z_{i:}^{(t)}$
      update $N^{(c)}$
    end for
    update B according to Eq. (16)
  until $Z^{(t)}$ does not change anymore
  merge communities according to Eq. (21) and (22)
until no more merging is possible



(a) Stage 1, Iteration 1 (b) Stage 1, Iteration 8 (c) Stage 2, Iteration 1

(d) Stage 3, Iteration 3 (e) Stage 4, Iteration 1 (f) Stage 4, Iteration 3

Figure 2: Changes of Community Labels in Each Iteration

5.1.2 Convergence Rate

In this part, we consider an ideal stochastic block model. We assume that there are K true communities, each containing the same number of nodes. We

initialize Z by putting nodes almost evenly and randomly into different temporary communities, with naturally a small bias that some temporary community should have a larger number of nodes from the same true community. Namely, we can define $p_R^{(0)} \approx \frac{1}{K^2}$ as the expected fraction of nodes belonging to the same true community over the total number of nodes, in a temporary community at iteration zero.

Theorem 5.3 *For an ideal stochastic block model, ML-SBM with random initialization converges quadratically as*

$$\lim_{t\to\infty} \frac{|p_R^{(t)} - \frac{1}{K}|}{|p_R^{(t-1)} - \frac{1}{K}|^2} = K.$$

Proof When the sizes of communities are equal, each community has $\frac{1}{K}$ nodes. Let $p_R^{(t-1)} = \frac{1}{K} + \theta$ where θ is a bias item. According to the analysis of the self-assembling process, by Eq. (4), we have the following expression:

$$\begin{aligned}
&\lim_{t\to\infty} \frac{|p_R^{(t)} - \frac{1}{K}|}{|p_R^{(t-1)} - \frac{1}{K}|^2} \\
&= \frac{|2(\frac{1}{K} + \theta) - (\frac{1}{K} + \theta)^2 \times K - \frac{1}{K}|}{|\theta|^2} \\
&= K,
\end{aligned}$$

which means that the convergence is quadratic.

5.2 Parallelism

The parallelism is designed as follows. Let S_I be the set of integers from 1 to N, partitioned into non-overlapping and approximately equal-size subsets, each of which is represented by S_{Ib} for the bth processor. For each processor, we also use local variables s_b, $\hat{s}_b$ and β_b, which have the similar definitions as s, $\hat{s}$ and β, but taking only the ith row ($i \in S_{Ib}$) of W into account.

We choose a message-passing model which is applicable on a variety of memory structures, whether shared, or distributed, or hybrid. We use non-blocking MPI communications to improve the communication-computation overlap. This has two-fold benefits: if the hardware allows, it utilizes the network bandwidth during computation, and it maintains a convergence rate for Z similar to the serial version by transmitting up-to-date community labels. In order to reduce the overall communication data, changes of $Z_{i:}$ are sent to processor b only if i has neighboring nodes in S_{Ib}. Because the communication bandwidth will be higher as the message size is larger, we buffer the change information of Z until computing f rows of Z finishes. Here f is chosen to be inversely proportional to the number of MPI ranks, and proportional to N, maintaining a consistent convergence rate.

The algorithm uses similar partitioning and communication strategies when computing B. Algorithm 2 presents the pseudocode of the parallel algorithm.

Algorithm 2 Parallel Algorithm (Par-SBM)

set community number to αK
initialize B, $Z^{(t)}$ into many tiny communities
$Z^{(t-1)} = Z^{(t)}$
repeat
  repeat
    do in parallel
    for $i \in S_{Ib}$ **do**
      compute $N^{(d)}$ for node i
      update $Z_{i:}^{(t)}$ according to Eq. (19)
      $Z_{i:}^{(t-1)} = Z_{i:}^{(t)}$
      synchronize changes of $Z_{i:}$ at frequency f
    end for
    update s_b and $\hat{s}_b$ for each β_b entries
    synchronize $s = \sum_b s_b$ and $\hat{s} = \sum_b \hat{s}_b$
    update B according to Eq. (16)
    update $N^{(c)}$
  until $Z^{(t)}$ does not change anymore
  merge communities according to Eq. (21) and (22)
until no more merging is possible

In parallel computing, the rows of W are distributed on different processors. If there are totally g processors (equal to the number of MPI ranks) and data are evenly distributed, the computation time of each processor for Eq. (16) and Eq. (19) is $\mathcal{O}(\frac{M}{g})$. When the computations for B are partitioned on each processor, reducing the number of communities also requires $\mathcal{O}(\frac{M}{g})$ for each processor. The total message length for Z and B during the iterations is $\mathcal{O}(g(N+K))$. If the bandwidth is constant, the speedup of the parallel algorithm is

$$\frac{T(M)}{T(\frac{M}{g}) + \mathcal{O}(N+K)}, \tag{23}$$

where T is a linear function, such that the numerator and denominator indicate the run times of the serial version and parallel version, respectively. Therefore, the speedup will increase towards g if the number of edges increases given the same N and K.

6 Experimental Results

6.1 The Serial Algorithm

In this section, we compare our algorithm (ML) with other serial algorithms in MATLAB. The competitors include the Louvain method (LV) [10], a Bayesian inference algorithm (BI) [21], spectral clustering (SC) [31], label propagation (LP) [9], and modularity optimization (MM) [8]. We use the LFR benchmark generator [44] to create graphs and run all of them for comparison. Each generated graph is of size 1000, average degree 30, maximum degree 50, exponent of degree distribution -2, exponent of community size distribution -2, minimum community size 20, and maximum size 100. μ is the proportion of inter-community connections to the intra-community ones.

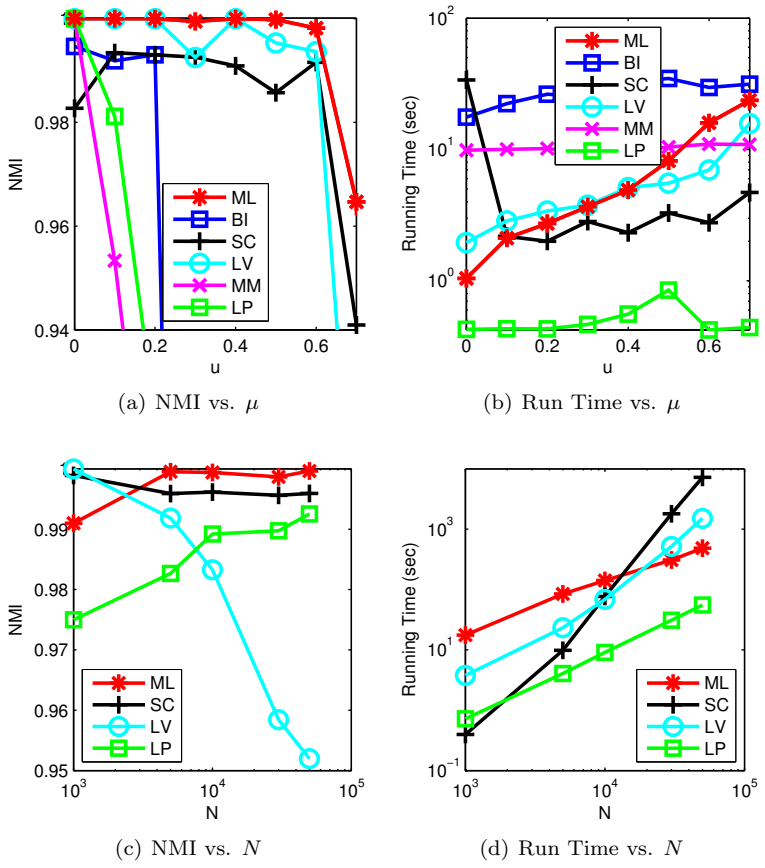
The accuracy is evaluated by normalized mutual information (NMI) [29], which compares the similarity between the computed community structure and the ground-truth one. The larger the NMI is, the more similar the two schemes are. If two schemes are identical, NMI is 1. The results are averaged on five runs.

From Figs. 3(a) and 3(b) we can see that our algorithm is relatively fast and has the best accuracy. When $\mu = 0$, most of the algorithms can find the correct result, and our algorithm is the second fastest one. As μ increases to 0.7, our algorithm needs longer time to iterate, but its accuracy is still the best. Only SC and LV can achieve a similar accuracy over a similar amount of time.

However, for large graphs, SC is not scalable, and LV is not accurate. Our algorithm is the method of choice for both scalability and accuracy. Figs. 3(c) and 3(d) show the results on graphs similar as the previous experiment but the average degree is 10 and $\mu = 0.1$. On large graphs, compared with other algorithms, ours consistently produces more accurate results with respect to the ground-truth setting, and the run time growth rate is smaller than others as the graph size grows. The slow growth of the run time of our algorithm is in accordance with the theoretical time complexity which is linear in the number of edges.

6.2 The Parallel Algorithm

In this section, we exploit distributed memory parallelism for larger graphs. When the input graph has more than one million nodes, typical algorithms for stochastic block models are not able to infer the parameters within a reasonable amount of time (for example, 24 hours). Therefore, we compare our algorithm (PS+number of processors) to a popular community detection algorithm, the Louvain method (LV) [10] and a fast graph partitioning algorithm Metis (MT) [45]. The Louvain method implemented in C can also generate level one structures as byproducts containing the finest communities (LVF). Those communities are merged into larger ones in LV. Parallel algorithms are often extended from serial versions [12], but the results are typically invariant. We use several metrics to compare the results, such as NMI, modularity [46], and



(a) NMI vs. μ (b) Run Time vs. μ

(c) NMI vs. N (d) Run Time vs. N

Figure 3: Comparison to popular algorithms in MATLAB. (a) and (b) represent the variation of average accuracy and run time over different choices of μ. (c) and (d) represent the variation over different choices of graph size N.

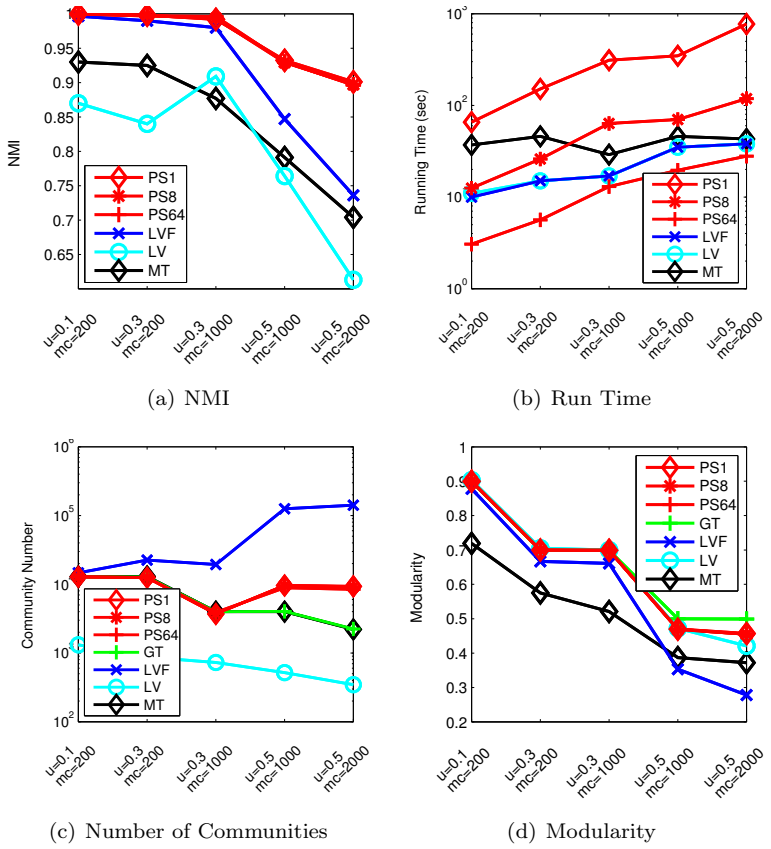
conductance (the smaller, the better) [47]. The results on those quantities are averaged on five runs.

First, we run our algorithm on benchmark graphs (LFR-1e6) generated by LFR benchmark [44]. The parameters are similar to the serial case except the graph size is increased to one million, and we change two parameters (the mixing parameter μ and maximum community size mc) to have more variations. Fig. 4 shows the results, in which GT is the ground truth given by the generator. Generally, the increase of μ and mc will decrease the community detection accuracy, but the results generated by our algorithm are always most close to the ground truth according to NMI. The results with our algorithm are also very close to the ground truth in numbers of communities (except for MT which is predefined) and modularity, and are usually better than others in modularity.

In Fig. 4(b), as the problem becomes more difficult from left to right, our algorithm spends more time to maintain the quality, while the other algorithms finish within the same amount of time as before while sacrificing the quality. In addition, the performance of our algorithm is stable as the run results using 8 processors (P8) and 64 processors (P64) are fairly close.

Graph Name	Node Number	Edge Number
LFR-1e6	1,000,000	$\sim$ 5,000,000
cit-Patents	3,774,768	16,518,948
dblp-Coauthor	1,314,050	10,724,828

Table 1: Properties of the test graphs

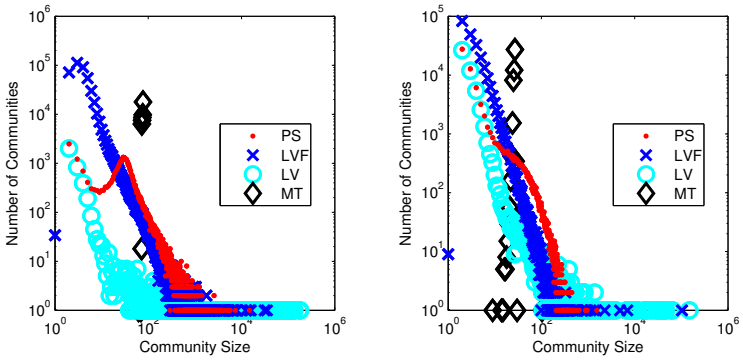


(a) NMI (b) Run Time

(c) Number of Communities (d) Modularity

Figure 4: Comparison on benchmark data

We also compare algorithms on real-world graphs: cit-Patents [48] and dblp-



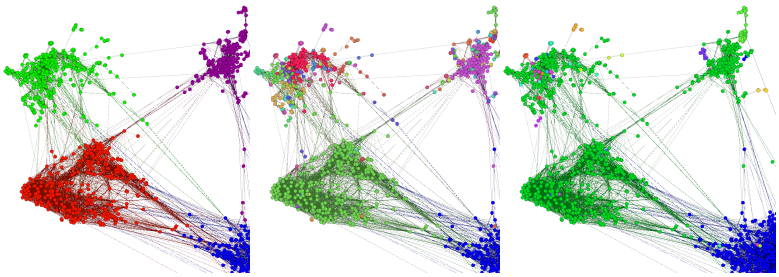
(a) cit-Patents (b) dblp-Coauthor

Figure 5: Number of communities versus community size

Graph	Algo	Cond	Mod	Time
cit-Patents	PS1	0.418	0.631	1514
	PS8	0.416	0.638	284
	PS64	**0.415**	**0.639**	51
	LVF	0.594	0.560	103
	MT	0.488	0.487	243
dblp-Coauthor	PS1	0.114	0.631	186
	PS8	0.113	**0.632**	33
	PS64	**0.112**	**0.632**	6
	LVF	0.397	0.561	22
	MT	0.621	0.253	73

Table 2: Comparison on real-world data

Coauthor [49]. Nodes in cit-Patents are patents in U.S. patent dataset and edges represent the citation relationships between the two patents. dblp-Coauthor is a coauthor graph extracted from publications in DBLP database. It is also worth noting that LVF has higher NMI and more communities than LV in benchmark test. It indicates that LV suffers from the resolution limit problem [16] that prefers unrealistically large communities. A similar problem happens on LV for real-world data as it generates many communities containing more than 10^4 nodes (Fig. 5). Especially, for example, for dblp-Coauthor network, the biggest community by LV contains about 10^5 nodes (10% of the whole graph). It is not realistic for collaboration networks and is orders of magnitude larger than a reasonable community size for human interaction [47], so LV is omitted from the remaining comparisons. Alternatively, choosing low level structures such



(a) by Par-SBM (b) by LVF (c) by LV2

Figure 6: Subgraph of the dblp-Coauthor network, where colors represent the community labels determined by different algorithms

as LVF can be a remedy [50], although it still contains a few extremely large communities.

On the other hand, MT tends to find even-size communities, which is not desirable for community detection either. Table 2 shows the quality of results by different algorithms, where run time is in seconds.

6.3 Microscopic Example on dblp-Coauthor Network

In this section, we analyze the community detection results of a run by our Par-SBM (PS) and the Louvain method [10] using random initialization. For the Louvain method, we pick the Level One structure with finest communities (LVF) and the Level Two structure denoted by LV2. Communities become larger in higher level structures by the Louvain method. For example, we consider community containing author "Michael_Wooldridge," and use S_{PS}, S_{LVF}, and S_{LV2} to represent the nodes in the community found by Par-SBM and the Louvain method, respectively. The community size by our algorithm is comparable to the one by LVF, but much smaller than the one by LV2, as $|S_{PS}| = 255$ and $|S_{LVF}| = 166$, while $|S_{LV2}| = 26294$.

All the algorithms are able to identify blue nodes in Fig. 6 as a separate community, while our Par-SBM provides the clearest detail. S_{LV2} contains one hundred times more nodes than S_{PS}. Among all the communities found by Par-SBM, the red, green, and purple communities in Fig. 6(a) are the top-three contributors to S_{LV2}, owning 2.96% of total nodes in S_{LV2}. The three communities can be represented by three nodes respectively: purple by Dr. Michael Wooldridge (multi-agent systems), green by Dr. Bruce W. Porter (knowledge systems), and red by Dr. Christopher W. Geib (Reports of the AAAI Conference Workshops). Nodes in red form a strong community because the authors (although from different areas of AI) are fully connected by the workshop reports annually. However, some nodes in the aforementioned three communities are inappropriately separated by LVF as shown in Fig. 6(b).

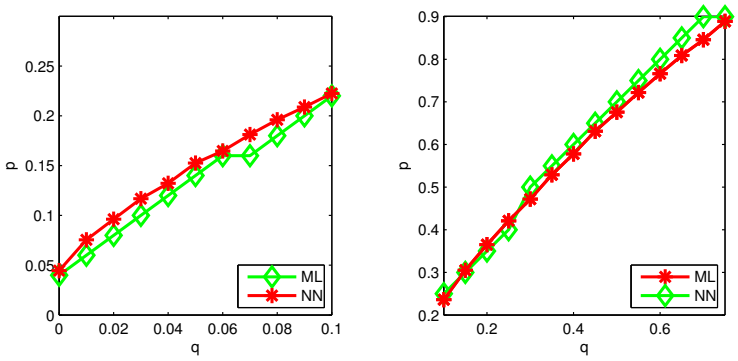
In addition, Par-SBM can find relevant nodes more than LVF does. For example, Par-SBM uniquely includes "H._H._Pham" and "Ali_M._Aseere" into the purple community in Fig. 6(a). The corresponding authors have "agent" in the title of all their publications.

6.4 Comparing to Nuclear Norm Minimization

In 2012, a nuclear norm minimization algorithm (NN) [22] is proposed to solve such SBM problem. It naturally comes with time complexity $\mathcal{O}(N^3)$ due to the SVD decomposition, but it can find the global optimum theoretically and has the advantage of clustering sparse graphs. As their source code is not available, we run our serial algorithm with exactly the same setting as it is in [22] and compare the accuracy.

The test graph contains 1000 nodes with 5 communities; over different q, we take the smallest p that succeeds, and average over 20 trials; success means the misclassification for pairs of nodes is less than 0.1%.

Fig. 7 compares the accuracy of our algorithm (ML) with NN, which is the best reported in [22]. This figure shows that our algorithm is very competitive to the nuclear norm minimization (NN), and the minimum possible gap between p and q for the two algorithms are similar. This fact indicates that our algorithm can find a local optimum that is very close to a global one with much less time complexity.



(a) Identifiability at $0 \leq q \leq 0.1$ (b) Identifiability at $q \geq 0.1$

Figure 7: Comparison to the State-of-the-art Algorithm by Nuclear Norm Minimization. The Curves Represent the Average Smallest p Required by the Algorithms to Successfully Recover the Communities.

6.5 Strong Scaling on Large Graphs

We use four different graphs. The first graph (LFR-1e6) is generated by LFR benchmark [44] using the same setting as in Fig. 3(c) and 3(d) except that the graph size is much larger. The second graph (Bitcoin) is a Bitcoin transaction network [51], where each node corresponds to one Bitcoin address, and each edge indicates that there is at least one transaction between two addresses. The third graph (soc-LiveJournal1 [47]) is obtained from Stanford Large Network Dataset Collection. The fourth graph (Wikipedia) contains most of the pages and the interlinks on Wikipedia in 2009.

The graph properties are listed in Column "Node Number" and "Edge Number" in Table 3. These graph data are preprocessed and partitioned on each processor so that different processors will have a similar number of nodes and edges.

Graph Name	Node Number	Edge Number
Bitcoin	13,086,528	44,032,115
soc-LiveJournal1	4,847,571	68,993,773
Wikipedia	5,716,808	130,160,392

Table 3: Properties of the Test Graphs

We run our algorithm on these graphs using different number of processors, but the accuracy is quite stable as expected. For example, when running on LFR-1e6 with ground-truth cluster information, the resulted NMI by our algorithm is in the range of 0.997 ± 0.002, indicating that the detected clusters are very similar to the ground truth, regardless the number of processors.

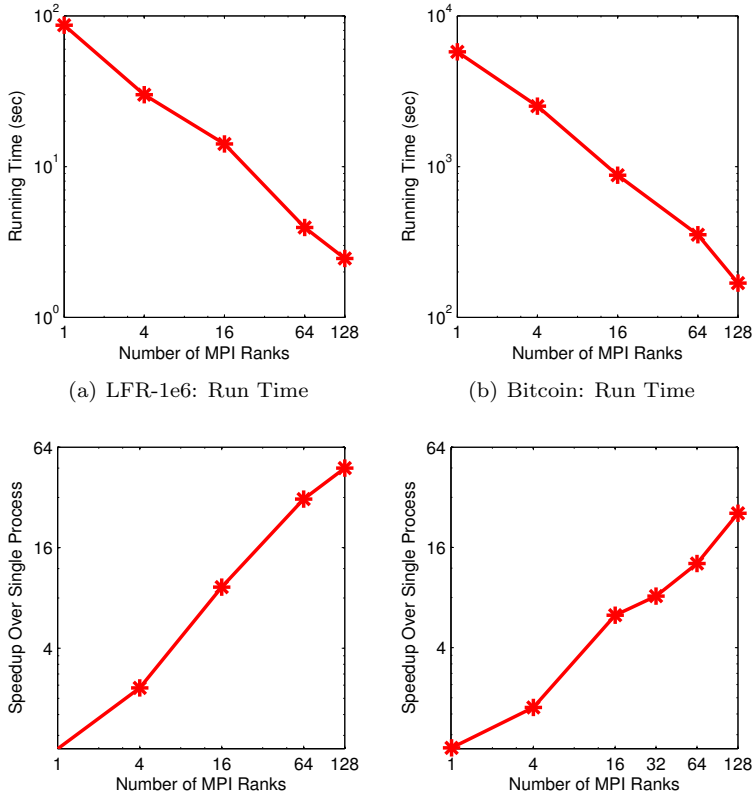
Fig. 8 demonstrates the run time and the speedup. Our algorithm on all the graphs can achieve a significant speedup as the number of processors grows. The Wikipedia graph has the highest average degree in all the test graphs, and Fig. 8(c) shows that our algorithm can achieve the biggest speedup of 48x on 128 cores. This phenomenon is in accordance with Eq. (23).

We also compare the speedup performance with a recently proposed parallel community detection algorithm [39, 52] running on shared-memory systems. Their community detection algorithm achieves 5.1x speedup on 16 cores using Intel Xeon X5570 CPU and 9.4x speedup using 32 cores on XMT2 [52] for soc-LiveJournal1 data.

Using the same data set and the same Intel Xeon CPU, our algorithm achieves an 8.2x and 12.8x speedup on 16 and 32 cores respectively, as shown in Fig. 8(d). Furthermore, 25.7x speedup can be achieved on 128 cores.

7 Concluding Remarks

In this paper, we have proposed a fast algorithm for clustering nodes in large graphs using stochastic block models. We have adopted alternative updates and



(a) LFR-1e6: Run Time

(b) Bitcoin: Run Time

(c) Wikipedia: Speedup

(d) LiveJournal: Speedup

Figure 8: Performance Evaluation for Graphs

coordinate descent methods to infer the latent parameters, and used community shrinking and expanding to improve accuracy. Our algorithm has linear time complexity in the number of edges, and can scale to multi-processor systems.

Compared with other community detection algorithms, our SBM-based method can produce high quality results on benchmark graphs and find interesting communities in the real-world graphs.

This work can boost the application of SBMs on big data and bring new insights by overcoming the accuracy or run time shortages of traditional algorithms. The code is available at https://github.com/pdacorn/par-sbm.

8 Acknowledgements

This work is supported by King Abdullah University of Science and Technology (KAUST) Sensor Innovation, China Postdoctoral Science Foundation

(NO.2017M612047), National Key Technology R&D Program (NO. 2015BAF14B01), and Excellent Youth Foundation of Zhejiang Scientific Committee (NO. L-R13F020004).

References

[1] Mitchell M. Complex systems: Network thinking. *Artificial Intelligence*, 2006, 170(18):1194–1212.

[2] Wang P, He W, Zhao J. A tale of three social networks: User activity comparisons across facebook, twitter, and foursquare. *IEEE Internet Computing*, Mar 2014, 18(2):10–15.

[3] Fortunato S, Castellano C. Community structure in graphs. In *Computational Complexity*, pp. 490–512. Springer, 2012.

[4] Yang P, Zhao P, Zheng V W, Li X L. An aggressive graph-based selective sampling algorithm for classification. In *Data Mining (ICDM), 2015 IEEE International Conference on*, Nov 2015, pp. 509–518.

[5] Kemp C, Tenenbaum J B, Griffiths T L, Yamada T, Ueda N. Learning systems of concepts with an infinite relational model. In *AAAI*, volume 3, 2006, p. 5.

[6] Ansótegui C, Giráldez-Cru J, Levy J. The community structure of SAT formulas. In *Theory and Applications of Satisfiability Testing–SAT 2012*, pp. 410–423. Springer, 2012.

[7] Wang H, Zhang P, Tsang I, Chen L, Zhang C. Defragging subgraph features for graph classification. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, 2015, pp. 1687–1690.

[8] Newman M E. Fast algorithm for detecting community structure in networks. *Physical Review E*, 2004, 69(6):066133.

[9] Raghavan U N, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 2007, 76(3):036106.

[10] Blondel V D, Guillaume J L, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008, 2008(10):P10008.

[11] Liu J, Wang C, Danilevsky M, Han J. Large-scale spectral clustering on graphs. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 2013, pp. 1486–1492.

[12] Wickramaarachchi C, Frincu M, Small P, Prasanna V. Fast parallel algorithm for unfolding of communities in large graphs. In *18th IEEE High Performance Extreme Computing Conference (HPEC 14)*, 2014, pp. 1–6.

[13] Staudt C, Meyerhenke H. Engineering parallel algorithms for community detection in massive networks. *IEEE Transactions on Parallel and Distributed Systems*, 2015, PP(99):1–1.

[14] Li G, Pan Z, Xiao B, Huang L. Community discovery and importance analysis in social network. *Intelligent Data Analysis*, 2014, 18(3):495–510.

[15] Nadler B, Galun M. Fundamental limitations of spectral clustering. In *Advances in Neural Information Processing Systems*, 2006, pp. 1017–1024.

[16] Fortunato S, Barthelemy M. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 2007, 104(1):36–41.

[17] Xiang J, Hu K. Limitation of multi-resolution methods in community detection. *Physica A: Statistical Mechanics and its Applications*, 2012, 391(20):4995–5003.

[18] Holland P W, Laskey K B, Leinhardt S. Stochastic blockmodels: First steps. *Social Networks*, 1983, 5(2):109–137.

[19] Celisse A, Daudin J J, Pierre L. Consistency of maximum-likelihood and variational estimators in the stochastic block model. *Electronic Journal of Statistics*, 2012, 6:1847–1899.

[20] Daudin J J, Picard F, Robin S. A mixture model for random graphs. *Statistics and Computing*, 2008, 18(2):173–183.

[21] Hofman J M, Wiggins C H. Bayesian approach to network modularity. *Physical Review Letters*, 2008, 100(25):258701.

[22] Chen Y, Sanghavi S, Xu H. Clustering sparse graphs. In *Advances in Neural Information Processing Systems 25*, 2012, pp. 2213–2221.

[23] Grama A, Gupta A, Karypis G, Kumar V. *Introduction to Parallel Computing, Second Edition*. Pearson Education, 2003.

[24] Asanovic K, Bodik R, Catanzaro B C, Gebis J J, Husbands P, Keutzer K, Patterson D A, Plishker W L, Shalf J, Williams S W et al. The landscape of parallel computing research: A view from Berkeley. Technical report, Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 2006.

[25] Diaz J, Munoz-Caro C, Nino A. A survey of parallel programming models and tools in the multi and many-core era. *IEEE Transactions on Parallel and Distributed Systems*, 2012, 23(8):1369–1386.

[26] Dagum L, Menon R. Openmp: an industry standard api for shared-memory programming. *IEEE Computational Science & Engineering*, 1998, 5(1):46–55.

[27] Gropp W, Lusk E, Skjellum A. *Using MPI: portable parallel programming with the message-passing interface*, volume 1. MIT Press, 1999.

[28] Rabenseifner R, Hager G, Jost G. Hybrid mpi/openmp parallel programming on clusters of multi-core smp nodes. In *17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, 2009, pp. 427–436.

[29] Danon L, Diaz-Guilera A, Duch J, Arenas A. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005, 2005(09):P09008.

[30] Newman M E. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 2006, 103(23):8577–8582.

[31] Von Luxburg U. A tutorial on spectral clustering. *Statistics and Computing*, 2007, 17(4):395–416.

[32] Li W, Schuurmans D. Modular community detection in networks. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[33] Traag V A, Van Dooren P, Nesterov Y. Narrow scope for resolution-limit-free community detection. *Physical Review E*, 2011, 84(1):016114.

[34] White S, Smyth P. A spectral clustering approach to finding communities in graphs. In *Proceedings of the Fifth SIAM International Conference on Data Mining*, volume 119, 2005, p. 274.

[35] Zhang X, Newman M. Multiway spectral community detection in networks. *Physical Review E*, 2015, 92(5):052808.

[36] Biesan S, Anthony A, desJardins M. Block modeling in large social networks with many clusters. In *AAAI Fall Symposium Series*, 2012.

[37] Abbe E, Bandeira A S, Hall G. Exact recovery in the stochastic block model. *IEEE Transactions on Information Theory*, 2016, 62(1):471–487.

[38] Chin P, Rao A, Vu V. Stochastic block model and community detection in the sparse graphs: A spectral algorithm with optimal rate of recovery. *arXiv preprint arXiv:1501.05021*, 2015, 2(4).

[39] Riedy E J, Meyerhenke H, Ediger D, Bader D A. Parallel community detection for massive graphs. In *Parallel Processing and Applied Mathematics*, pp. 286–296. Springer, 2012.

[40] Bhowmick S, Srinivasan S. A template for parallelizing the louvain method for modularity maximization. In *Dynamics On and Of Complex Networks, Volume 2*, pp. 111–124. Springer, 2013.

[41] Gregori E, Lenzini L, Mainardi S. Parallel k-clique community detection on large-scale networks. *IEEE Transactions on Parallel and Distributed Systems*, 2013, 24(8):1651–1660.

[42] Kang U, Meeder B, Papalexakis E E, Faloutsos C. Heigen: Spectral analysis for billion-scale graphs. *IEEE Transactions on Knowledge and Data Engineering*, 2014, 26(2):350–362.

[43] Chung F, Lu L. *Complex Graphs and Networks (Cbms Regional Conference Series in Mathematics)*. American Mathematical Society, Boston, MA, USA, 2006.

[44] Lancichinetti A, Fortunato S. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 2009, 80(1):016118.

[45] Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 1998, 20(1):359–392.

[46] Clauset A, Newman M E, Moore C. Finding community structure in very large networks. *Physical Review E*, 2004, 70(6):066111.

[47] Leskovec J, Lang K J, Dasgupta A, Mahoney M W. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 2009, 6(1):29–123.

[48] Leskovec J, Kleinberg J, Faloutsos C. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the 11th ACM International Conference on Knowledge Discovery in Data Mining*, 2005, pp. 177–187.

[49] Ley M. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *Proceedings of the International Symposium on String Processing and Information Retrieval*, 2002, pp. 1–10.

[50] Good B H, Montjoye Y A, Clauset A. Performance of modularity maximization in practical contexts. *Physical Review E*, 2010, 81(4):046106.

[51] Kondor D, Psfai M, Csabai I, Vattay G. Do the rich get richer? an empirical analysis of the bitcoin transaction network. *PLoS ONE*, 02 2014, 9(2):e86197.

[52] Riedy J, Bader D A, Meyerhenke H. Scalable multi-threaded community detection in social networks. In *the 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW)*, 2012, pp. 1619–1628.