A Scalable Community Detection Algorithm for Large Graphs Using Stochastic Block Models

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

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.

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 or disciplinary groups in collaborative networks [Fortunato and Castellano, 2012; Kemp *et al.*, 2006; Ansótegui *et al.*, 2012].

In this work, we consider non-overlapping community structures. Many community detection algorithms handle such a problem [Newman, 2004; Raghavan *et al.*, 2007; Blondel *et al.*, 2008; Fortunato and Castellano, 2012; Liu *et al.*, 2013; Wickramaarachchi *et al.*, 2014; Staudt and Meyerhenke, 2015]. However, they come along with limitations for large graphs, for example, in handling community heterogeneity [Nadler and Galun, 2006; Fortunato and Barthelemy, 2007; Xiang and Hu, 2012]. Alternatively, model-based methods can produce more reliable and accurate results when the model assumptions are in accordance with the real graphs. Stochastic block models (SBMs) are among the important probabilistic tools describing the connectivity relationship between pairs of nodes [Holland *et al.*, 1983], and have received considerable attention both in theoretical [Celisse *et al.*, 2012] and application domains [Daudin *et al.*, 2008].

Many algorithms have been proposed to infer the parameters of SBMs, such as Bayesian estimation [Hofman and Wiggins, 2008] and nuclear norm minimization [Chen *et al.*, 2012]. Bayesian estimation defines the prior distributions on latent variables (community labels of nodes) and maximizes the posterior distribution when a graph is given [Hofman and Wiggins, 2008]. Nuclear norm minimization of a matrix minimizes the sum of its singular values, and their algorithm is verified on graphs containing one thousand nodes.

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 a 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) [Danon *et al.*, 2005] with respect to the

ground-truth. Experiments on real-world graphs show that our algorithm can produce more meaningful communities with good quality.

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

3 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.

3.1 Estimation Algorithm

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

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

subject to $0 \leq B_{ij} \leq 1$, $Z_{ij} \in \{0, 1\}$, $\sum_j Z_{ij} = 1$. Roughly speaking, 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,

$$\frac{\partial L}{\partial \beta} = \frac{s}{\beta} - \frac{\hat{s}}{1 - \beta} \tag{3}$$

and setting the derivative to be zero, we have

$$\beta = \frac{s}{s + \hat{s}}. \tag{4}$$

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 1 *For fixed Z, the objective function $L(B, Z|W)$ achieves its global maximum if entries of B are updated according to Eq. (4).*

Proof When Z is fixed, because each entry of B can optimize the objective function independently, after updating all the entries by Eq. (4), 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

$$\left.\frac{\partial^2 L}{\partial \beta^2}\right|_{\beta=\frac{s}{s+\hat{s}}} = -\frac{s}{\beta^2} - \left.\frac{\hat{s}}{(1-\beta)^2}\right|_{\beta=\frac{s}{s+\hat{s}}} < 0. \tag{5}$$

Therefore, when Z is given, the objective function at β (determined by Eq. (4)) is a global maximum. As each entry of B is irrelevant to each other, we can update B by Eq. (4) 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. (2), 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

$$r_{\max} = \operatorname*{argmax}_r \sum_{j \neq i} \log \big[(1 - W_{ij}) + (2W_{ij} - 1)(B[Z^{(t-1)}]^T)_{rj}\big]. \quad (6)$$

The time complexity for updating one node by Eq. (6) 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:

$$r_{\max} = \operatorname*{argmax}_r \sum_{k=1}^{K} \big[N_k^{(d)} \log(B_{rk}) + (N_k^{(c)} - N_k^{(d)}) \log(1 - B_{rk})\big]. \quad (7)$$

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. (7) 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. (7). 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 2 $\min_{Z^{(t)}} -G(B, Z^{(t)}, Z^{(t-1)})$ *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.*

We omit the proof because it is direct.

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. Here we refer to 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 is called the *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.

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 ratio 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.

$$L_p = c_{rk} \log(p) + (n_{rk} - c_{rk}) \log(1 - p), \quad (8)$$

$$L_q = c_{rk} \log(q) + (n_{rk} - c_{rk}) \log(1 - q), \quad (9)$$

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.

Convergence Rate

We consider an ideal stochastic block model, which comprises K true communities with the same properties (community size, internal density, etc.). We define $p_R^{(0)} \approx \frac{1}{K^2}$ as the expected ratio of nodes belonging to the same true community over the total number of nodes over the total number of nodes, in a temporary community at iteration zero.

Theorem 3 *For an ideal stochastic block model, ML-SBM with random initialization converges quadratically as*

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

The result can be proved by utilizing the concept of temporary communities.

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. (7)
      $Z_{i:}^{(t-1)} = Z_{i:}^{(t)}$
      update $N^{(c)}$
    end for
    update B according to Eq. (4)
  until $Z^{(t)}$ does not change anymore
  merge communities according to Eq. (8) and (9)
until no more merging is possible

3.2 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 communicating strategies when computing B. Algorithm 2 presents the pseudocode of the parallel algorithm.

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. (4) and Eq. (7) 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

$$\frac{T(M)}{T(\frac{M}{g}) + \mathcal{O}(N+K)}, \tag{10}$$

where T is a linear function, such that the numerator and denominator indicate the running 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.

Algorithm 2 Parallel Algorithm (Par-SBM)

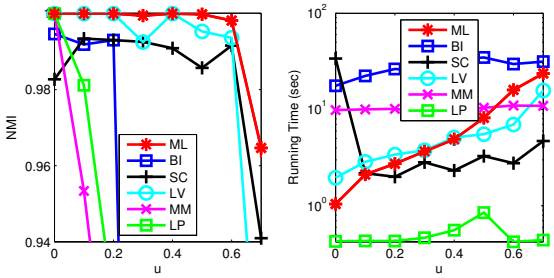
set community number to αK
initialize B, $Z^{(t)}$ into many tiny communities
$Z^{(t-1)} = Z^{(t)}$
repeat
  repeat
    do in parallel
    for $i \in S_{Ib}$ **do**
      compute $N^{(d)}$ for node i
      update $Z_{i:}^{(t)}$ according to Eq. (7)
      $Z_{i:}^{(t-1)} = Z_{i:}^{(t)}$
      synchronize 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. (4)
    update $N^{(c)}$
  until $Z^{(t)}$ does not change anymore
  merge communities according to Eq. (8) and (9)
until no more merging is possible

4 Experimental Results

4.1 The Serial Algorithm

In this section, we compare our algorithm (ML) with other serial algorithms in MATLAB. The competitors include the Louvain method (LV) [Blondel *et al.*, 2008], a Bayesian inference algorithm (BI) [Hofman and Wiggins, 2008], spectral clustering (SC) [Von Luxburg, 2007], label propagation (LP) [Raghavan *et al.*, 2007], and modularity optimization (MM) [Newman, 2004]. We use the LFR benchmark generator [Lancichinetti and Fortunato, 2009] 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) [Danon *et al.*, 2005], 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. 1(a) and 1(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 a 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.



(a) NMI vs. μ (b) Running Time vs. μ

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

Figure 1: Comparison to popular algorithms in MATLAB. (a) and (b) represent the variation of average accuracy and running time over different choices of μ. (c) and (d) represent the variation over different choices of graph size N.

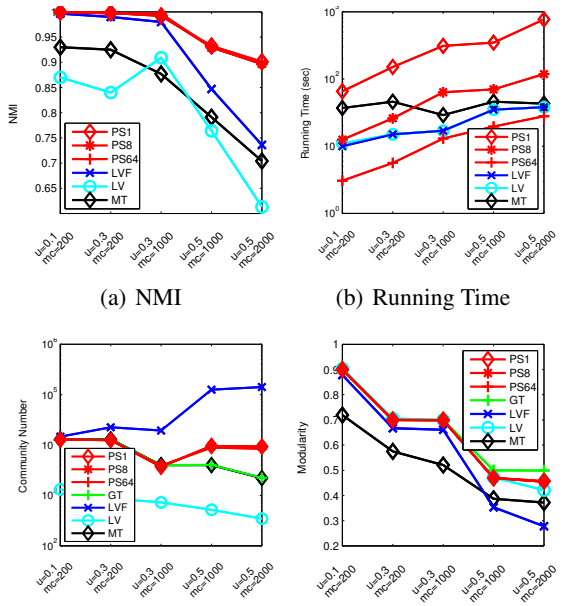
However, for large graphs, SC is not scalable, and LV is not accurate. Our algorithm is the method of choice for both scalability and accuracy. Figs. 1(c) and 1(d) show the results on graphs similar as the previous experiment but the average degree is 10 and $\mu = 0.1$. On large graphs, compared with other algorithms, ours consistently produces more accurate results with respect to the ground-truth setting, and the running time growth rate is smaller than others as the graph size grows. The slow growth of the running time of our algorithm is in accordance with the theoretical time complexity which is linear in the number of edges.

4.2 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) [Blondel *et al.*, 2008] and a fast graph partitioning algorithm Metis (MT) [Karypis and Kumar, 1998]. 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 [Wickramaarachchi *et al.*, 2014], but the results are typically invariant. We use several metrics to compare the results, such as NMI, modularity [Clauset *et al.*, 2004], and conductance (the smaller, the better) [Leskovec *et al.*, 2009]. The results on those quantities are averaged on five runs.

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



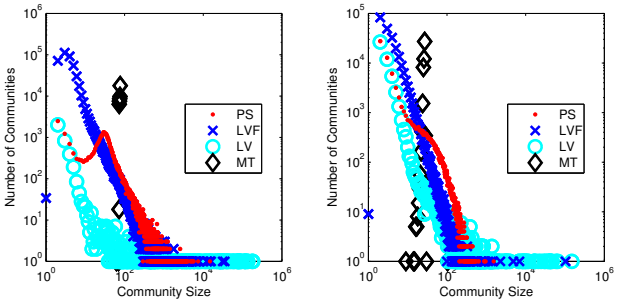
(a) NMI (b) Running Time

(c) Number of Communities (d) Modularity

Figure 2: Comparison on benchmark data

First, we run our algorithm on benchmark graphs (LFR-1e6) generated by LFR benchmark [Lancichinetti and Fortunato, 2009]. 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. 2 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. 2(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 while sacrificing the quality. In addition, the performance of our algorithm is stable as the running results using 8 processors (P8) and 64 processors (P64) are fairly close.

We also compare algorithms on real-world graphs: cit-Patents [Leskovec *et al.*, 2005] and dblp-Coauthor [Ley, 2002]. Nodes in cit-Patents are patents in U.S. patent dataset



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

Figure 3: Number of communities versus community size

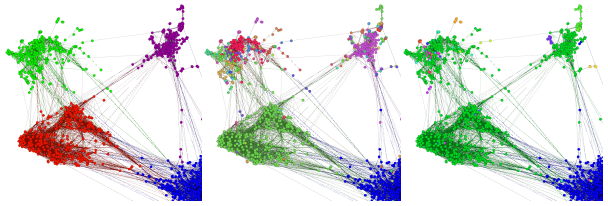
Graph	Algo	Cond	Mod	Time (sec)
cit-Patents	PS1	0.418	0.631	1514
	PS8	0.416	0.638	284
	PS64	**0.415**	**0.639**	51
	LVF	0.594	0.560	103
	MT	0.488	0.487	243
dblp-Coauthor	PS1	0.114	0.631	186
	PS8	0.113	**0.632**	33
	PS64	**0.112**	**0.632**	6
	LVF	0.397	0.561	22
	MT	0.621	0.253	73

Table 2: Comparison on real-world data

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 [Fortunato and Barthelemy, 2007] 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. 3). 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 [Leskovec *et al.*, 2009], so LV is omitted from the remaining comparisons. Alternatively, choosing low level structures such as LVF can be a remedy [Good *et al.*, 2010], 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.

4.3 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 [Blondel *et al.*, 2008] 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. Taking the Conference Chair as an 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 $|S_{LV2}| = 26294$.



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

Figure 4: Subgraph of the dblp-Coauthor network, where colors represent the community labels determined by different algorithms

All the algorithms are able to identify blue nodes in Fig. 4 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. 4(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 forms 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. 4(b).

5 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 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 running time shortages of traditional algorithms. The code is

available at https://github.com/pdacorn/par-sbm.

References

[Ansótegui *et al.*, 2012] Carlos Ansótegui, Jesús Giráldez-Cru, and Jordi Levy. The community structure of SAT formulas. In *Theory and Applications of Satisfiability Testing–SAT 2012*, pages 410–423. Springer, 2012.

[Blondel *et al.*, 2008] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[Celisse *et al.*, 2012] Alain Celisse, Jean-Jacques Daudin, and Laurent Pierre. Consistency of maximum-likelihood and variational estimators in the stochastic block model. *Electronic Journal of Statistics*, 6:1847–1899, 2012.

[Chen *et al.*, 2012] Yudong Chen, Sujay Sanghavi, and Huan Xu. Clustering sparse graphs. In *Advances in Neural Information Processing Systems 25*, pages 2213–2221, 2012.

[Clauset *et al.*, 2004] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, 2004.

[Danon *et al.*, 2005] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, 2005.

[Daudin *et al.*, 2008] Jean Jacques Daudin, Franck Picard, and Stéphane Robin. A mixture model for random graphs. *Statistics and Computing*, 18(2):173–183, 2008.

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

[Fortunato and Castellano, 2012] Santo Fortunato and Claudio Castellano. Community structure in graphs. In *Computational Complexity*, pages 490–512. Springer, 2012.

[Good *et al.*, 2010] Benjamin H Good, Yves-Alexandre de Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106, 2010.

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

[Holland *et al.*, 1983] Paul W. Holland, Kathryn B. Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

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

[Kemp *et al.*, 2006] Charles Kemp, Joshua B Tenenbaum, Thomas L Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, volume 3, page 5, 2006.

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

[Leskovec *et al.*, 2005] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the 11th ACM International Conference on Knowledge Discovery in Data Mining*, pages 177–187. ACM, 2005.

[Leskovec *et al.*, 2009] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

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

[Liu *et al.*, 2013] Jialu Liu, Chi Wang, Marina Danilevsky, and Jiawei Han. Large-scale spectral clustering on graphs. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 1486–1492. AAAI Press, 2013.

[Nadler and Galun, 2006] Boaz Nadler and Meirav Galun. Fundamental limitations of spectral clustering. In *Advances in Neural Information Processing Systems*, pages 1017–1024, 2006.

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

[Raghavan *et al.*, 2007] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007.

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

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

[Wickramaarachchi *et al.*, 2014] Charith Wickramaarachchi, Marc Frincu, Patrick Small, and Viktor Prasanna. Fast parallel algorithm for unfolding of communities in large graphs. In *18th IEEE High Performance Extreme Computing Conference (HPEC 14)*, 2014.

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