The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Towards Efficient Detection of Overlapping Communities in Massive Networks

Bing-Jie Sun,[1,2] Huawei Shen,[1] Jinhua Gao,[1,2] Wentao Ouyang,[1] Xueqi Cheng[1]
[1]CAS Key Laboratory of Network Data Science and Technology, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
{sunbingjie, gaojinhua}@software.ict.ac.cn, {shenhuawei, ouyangwt, cxq}@ict.ac.cn

Abstract

Community detection is essential to analyzing and exploring natural networks such as social networks, biological networks, and citation networks. However, few methods could be used as off-the-shelf tools to detect communities in real world networks for two reasons. On the one hand, most existing methods for community detection cannot handle massive networks that contain millions or even hundreds of millions of nodes. On the other hand, communities in real world networks are generally highly overlapped, requiring that community detection method could capture the mixed community membership. In this paper, we aim to offer an off-the-shelf method to detect overlapping communities in massive real world networks. For this purpose, we take the widely-used Poisson model for overlapping community detection as starting point and design two speedup strategies to achieve high efficiency. Extensive tests on synthetic and large scale real networks demonstrate that the proposed strategies speedup the community detection method based on Poisson model by 1 to 2 orders of magnitudes, while achieving comparable accuracy at community detection.

Introduction

Many real world networks, such as biological networks, social networks, and citation networks, are found to divide naturally into communities, i.e., groups of nodes with relatively denser connections within groups but sparser connections between them (Girvan and Newman 2002). One fundamental problem in network analysis is how to detect such communities observed in networks.

Community detection has attracted much interest among scientists from various fields in the last decade (Fortunato 2010; Palla et al. 2005; Shen et al. 2009; Karrer and Newman 2011; Ren et al. 2009; Newman and Leicht 2007; Newman 2013; Sun et al. 2017; Sun, Shen, and Cheng 2014; Rosvall and Bergstrom 2008; Newman and Peixoto 2015). However, few methods could be used as off-the-shelf tools to detect communities in real world networks, which is attributed to two challenges underlying the problem of community detection. On the one hand, many real world networks have millions and even hundreds of millions of nodes. Most existing methods cannot handle networks at this scale (Newman and Girvan 2004; Cheng et al. 2010; Newman 2013; 2006). On the other hand, communities in real-world networks are highly overlapped, requiring that community detection method could capture the heterogeneity of community membership (Airoldi et al. 2008; Palla et al. 2005; Kim and Leskovec 2012; Newman and Leicht 2007; Karrer and Newman 2011). Therefore, we lack a method that could efficiently detect overlapping communities in massive networks, especially when the number of communities is large.

In the last few years, researchers attempt to combat the aforementioned problem by improving the computational efficiency of several well-performed and principled methods for overlapping community detection. For example, Gopalan and Blei proposed an efficient method to uncover overlapping communities in massive networks (Gopalan and Blei 2013). Their method is actually a variant of mixed membership stochastic blockmodel (Airoldi et al. 2008), achieving high efficiency by stochastic variational inference (Latouche, Birmele, and Ambroise 2012), i.e., approximately estimating the posterior community membership of nodes by analyzing a sub-sampling of network rather than the whole network. Yang et al. proposed BIGCLAM method which optimizes a variant of non-negative matrix factorization objective function to discover communities through maximizing the likelihood of network (Yang and Leskovec 2013). The speedup lies in transforming the loop over all nodes into the loop of neighbor nodes when calculating the gradient of a node. Sun et al. proposed to use fast non-overlapping community detection method to initialize overlapping community detection method, achieving high efficiency by avoiding local optimum of non-convex optimization (Sun, Shen, and Cheng 2014). However, these methods are still inappropriate to detect overlapping communities in massive networks when the number of communities is large. The key defect of these methods lies in that the basic models for these methods consider all nodes pairs with or without links, with an implicit squared computation complexity.

In this paper, we aim to achieve an efficient method to detect overlapping communities in massive networks. For this purpose, we take a well-performed and principled model, i.e., Poisson model, for overlapping community detection as starting point, and design speedup strategies to achieve high efficiency. The obtained method distinguishes itself

from previous speedup methods in two aspects: (1) Different from mixed membership stochastic blockmodel (Airoldi et al. 2008) and non-negative matrix factorization method (Lee and Seung 1999; Choi 2008; Ding et al. 2006), the complexity of Poisson model is linear with respect to the number of edges rather than the square of the number of nodes. This offers a good starting point for designing efficient methods for community detection; (2) Existing methods mostly focus on the speedup of posterior inference without a fine-grained scrutinization of the process of algorithm. Consequently, these methods fail to handle the scenario that the number of communities are large.

In this paper, we design two speedup strategies by scrutinizing the factors that affects the complexity of Poisson model, i.e., $O(mKt)$, where m is the number of edges in network, K is the number of communities to be discovered, and t is the number of iteration steps. Firstly, we reduce the complexity of the Poisson model using an early-stopping criterion, i.e., we stop the update of the posterior of each edge if the community membership of its two end nodes both keep unchanged. Secondly, we leverage the sparsity of community membership of nodes, i.e., each node only belongs to a handful of communities, and adaptively reduce the number of communities for each node during the process of learning the community membership of nodes. The combination of the two strategies further reduce the number of iterations before convergence. Taken together, we achieve an efficient method to detect overlapping communities in massive networks.

Extensive tests on synthetic benchmark networks demonstrate that our method improves the efficiency of community detection in 1 to 2 orders of magnitude. Finally, to demonstrate the efficiency of our method, we apply it to several real world massive networks with as many as millions of nodes and hundreds of millions of edges, verifying its efficiency in detecting overlapping community structure.

Preliminary

Traditional generative models for community detection discover the communities by maximizing the likelihood of generating the network. In these models, the community membership of a node is represented by a distributed vector. The probability of generating an edge between two nodes is generally modeled as $P(A_{ij}|\theta_i, \theta_j) = f(g(\theta_i, \theta_j))$, reflecting whether there exists an edge between node i and node j. θ_i is a vector which represents the community membership of node i. $g(\cdot)$ is a link function which transforms θ_i and θ_j into the required parameters for the probability distribution $f(\cdot)$ to generate edge (i, j). Different choices of $g(\cdot)$ and $f(\cdot)$ result in different models.

Mixed membership stochastic blockmodel, MMSB for abbreviation, considers the generation of node pairs from the block perspective, and adopts *Bernoulli* distribution to model the linking state between two nodes, with the parameter characterized by the block interaction and the community memberships of those node pairs (Airoldi et al. 2008), i.e.,

$$P(A_{ij} = 1|\theta_i, \theta_j) = \sum_{r=1}^{K} \theta_{ir}\theta_{jr}\beta_r, \quad (1)$$

where β_r is the probability that two nodes are connected given that they both belong to community r, and K is the number of expected communities. In MMSB, the representation of each node is constrained by $\sum_r \theta_{ir} = 1$, which guarantees the link function $g(\theta_i, \theta_j) = \sum_{r=1}^{K} \theta_{ir}\theta_{jr}\beta_r$ offers a proper probability for the *Bernoulli* distribution. However, MMSB has to model the linking states between all n^2 node pairs, resulting in a $O(n^2)$ complexity, limiting its applicability to large-scale networks. Gopalan and Blei proposed to adopt the stochastic optimization algorithm to improve the efficiency. In each iteration, they subsampled a subset of node pairs, and updated the community memberships of the nodes based on those sampled pairs.

Another line of works aims to model the edge generation based on interactions between nodes. Yang et al. (Yang and Leskovec 2013) proposed to model the linking probability of two nodes using their community memberships directly, i.e.,

$$P(A_{ij} = 1|\theta_i, \theta_j) = 1 - exp(-\theta_i\theta_j^T). \quad (2)$$

Bernoulli distribution was also adopted for edge generation. However, in each iteration updating the gradient of each node takes $O(n)$ time,

$$\bigtriangledown l(\theta_i) = \sum_{j \in N(i)} \theta_j \frac{exp(-\theta_i\theta_j^T)}{1 - exp(-\theta_i\theta_j^T)} - \sum_{j \notin N(i)} \theta_j, \quad (3)$$

making it impractical for large-scale networks. $N(i)$ denotes the neighbor nodes of node i. Yang et al. proposed BIGCLAM algorithm which replaced the time consuming part, i.e. $\sum_{j \notin N(i)} \theta_j$, based on the observation that

$$\sum_{j \notin N(i)} \theta_j = \sum_j \theta_j - \theta_i - \sum_{j \in N(i)} \theta_j. \quad (4)$$

Thus the complexity for updating the gradient of a single node decreases from $O(n)$ to $O(|N(i)|)$.

Poisson model is another popular generative model for community detection (Karrer and Newman 2011; Ball, Karrer, and Newman 2011). Unlike previous models which assume a *Bernoulli* distribution for edge generation, it assumes that the number of edges between each node pair follows a *Poisson* distribution with mean $g(\theta_i, \theta_j) = \theta_i\theta_j^T$, i.e.,

$$P(A_{ij}|\theta_i, \theta_j) = \frac{(\sum_r \theta_{ir}\theta_{jr})^{A_{ij}}}{A_{ij}!} exp(-\sum_r \theta_{ir}\theta_{jr}). \quad (5)$$

Ball et al. fit the model to an observed network and adopt the expectation-maximization (EM) algorithm for optimization. Taking the log of Eq. (5), they maximize the log-likelihood using EM algorithm. The E-step is optimizing $q_{ij}(r)$ given the values of θ_{ir} and θ_{jr}

$$q_{ij}(r) = \frac{\theta_{ir}\theta_{jr}}{\sum_r \theta_{ir}\theta_{jr}}, \quad (6)$$

where $q_{ij}(r)$ is the probability that the edge between node i and j locates in community r, which satisfies $\sum_r q_{ij}(r) = 1$. The M-step optimizes θ_{ir} given $q_{ij}(r)$

$$\theta_{ir} = \frac{\sum_j A_{ij} q_{ij}(r)}{\sqrt{\sum_{ij} A_{ij} q_{ij}(r)}}. \quad (7)$$

Ball et al. pointed out that the space requirement is dominated by the parameter $q_{ij}(r)$. They proposed to use an intermediate variable, k_{ir}, which denotes the average number of edges connected to node i in community r. The definition for k_{ir} is

$$k_{ir} = \sum_j A_{ij} q_{ij}(r). \quad (8)$$

Original θ_{ir} can be retrieved by

$$\theta_{ir} = \frac{k_{ir}}{\sqrt{\kappa_r}}, \quad (9)$$

and $q_{ij}(r)$ can be calculated as

$$q_{ij}(r) = \frac{k_{ir} k_{jr}}{D_{ij} \times \kappa_r}, \quad (10)$$

where $\kappa_r = \sum_i k_{ir}$ and $D_{ij} = \sum_r \frac{k_{ir} k_{jr}}{\kappa_r}$.

The above Poisson model performs well in finding overlapping communities in moderate network. However, it still cannot be applied to large-scale networks. In this paper we analyze the optimization process and propose a two-level acceleration algorithm, which improves the speed of Poisson model by 1 to 2 orders of magnitudes, while demonstrating a comparable performance in community detection task.

Acceleration of Poisson Model

As shown in the previous section, the key part for *Poisson* model is the updating of k_{ir} for each node, which loops over all the edges. Thus the number of edges to be dealt with in each iteration is crucial to the algorithm's efficiency. In Ball's implementation, they proposed to discard the edges whose both end nodes' representations converge to be one-hot. However, most nodes in real-world networks are overlapping, resulting in more than one non-zero entry in representations. Observing that the representations of most nodes are converged after a few iterations, we propose edge level acceleration which discards the edges when its end nodes' representations both converged, without restricting them to converge into one-hot vector. Edge level acceleration can remarkably reduces the number of edges to be handled during each iteration.

Moreover, the representations of most nodes are very sparse, since most nodes belong to only a few communities. We propose dimension level acceleration to take advantage of the sparsity of representations. For each dimension k_{ir} in node i's representation, once it is updated to be 0, it will stay 0 for the rest of optimization. Thus we only need to keep track of non-zero dimensions in each node's representation.

Algorithm 1: PARAMUPDATE: Updating the membership of connected nodes

Input: Edge number m, Parameter κ, $EL[m][2]$ stores the edge list, $Converge[i]$ converge state of node i, $NZ[i]$ stores the non-zero dimensions of node i

Output: All the membership of nodes k_i^{new}.

```
1  for Each edge e = ⟨i, j⟩ in the edgelist do
2      if Edge e = ⟨i, j⟩ has not been removed then
3          if Converge[i]==True and
             Converge[j]==True then
4              Remove this node pair from optimization
               process.
5          else
6              if Converge[j]==False then
7                  k_j^new=DIMENOPT(j,i,NZ,κ);
8              if Converge[i]==False then
9                  k_i^new=DIMENOPT(i,j,NZ,κ);
10 return k_i^new for all nodes;
```

Edge Level Acceleration

According to Eq. (10), we can tell that once k_{ir} becomes 0, $q_{ij}(r)$ will stay 0 for any node j that connected to node i, resulting in k_{ir} staying 0 according to Eq.(8). Thus if the representation of a node becomes one-hot, it will stay being one-hot ever after. In Ball's implementation, if the representations of two end nodes of an edge become one-hot, they proposed to discard this edge in the rest of optimization as it will not change the community memberships of two end nodes. However, most nodes are actually overlapping, indicating their representations are unlikely to be one-hot during optimization. Nonetheless, the representations of most nodes tend to be settled after only a few iterations. Therefore we propose to discard edges whose both end nodes' representations are converged, as shown in line $3-4$ in Algorithm 1. Once both nodes' representations are converged, this edge is not considered for the rest of optimization. Otherwise, we continue to update k_i for unconverged node i, as done in line $6-9$. In experiment part this strategy proves to remarkably reduce the number of edges during the optimization process, resulting in a large decrease in the number of iterations needed for the convergence of the algorithm..

Dimension Level Acceleration

When updating $k_{i\cdot}$ for unconverged node i, we have to deal with a K-dimension vector. As pointed out before, k_{ir} will stay 0 after it is updated to 0. Since most nodes belong to only a few communities, their representations tend to be very sparse. We propose to keep track of only non-zero dimensions in each node's representation to further reduce computation cost, as done in Algorithm 2. The calculation of D_{ij} and updating of k_i only take places on the non-zero entries of node i's representation, which are recorded by $NZ[i]$ vector. Dimension level acceleration enables our algorithm to detect

Algorithm 2: DIMENOPT: Dimension level acceleration

Input: Node i and node j, Parameter κ, $NZ[i]$ stores the non-zero dimensions of node i
Output: Updated membership of node k_i^{new} and k_j^{new}

for *Each non-zero dimension* r *in* $NZ[i]$ **do**
  Calculate the D_{ij} using $D_{ij} = \sum_r \frac{k_{ir}^{old} k_{jr}^{old}}{\kappa_r}$.
for *Each non-zero dimension* r *in* $NZ[i]$ **do**
  Update the k_{ir}^{new} using $k_{ir}^{new} = k_{ir}^{new} + A_{ij} \frac{k_{ir}^{old} k_{jr}^{old}}{D_{ij} \times \kappa_r}$.
return k_i^{new};

Algorithm 3: LCNODE: Marking the dimension labels of converged Nodes

Input: Node number n, number of communities K, Parameter κ, k_i^{old} and k_i^{new}, $Converge[i]$ stores converge state of node i, $NZ[i]$ stores the non-zero dimensions of node i
Output: Updating the converge state of node $Converge[i]$, the non-zero dimensions of each node $NZ[i]$ and κ

1 $error = 0$;
2 **for** *Each node* i **do**
3 **if** $Converge[i]==False$ **then**
4 **for** *Each* r *in* $NZ[i]$ **do**
5 $error+ = |k_{ir}^{old} - k_{ir}^{new}|$;
6 **if** $k_{ir}^{new} < \delta$ **then**
7 $k_{ir}^{new} = 0$;
8 Remove r from $NZ[i]$;
9 $\kappa_r = \kappa_r + k_{ir}^{new} - k_{ir}^{old}$;
10 $k_{ir}^{old} = k_{ir}^{new}$;
11 $k_{ir}^{new} = 0$;
12 **if** $error < \delta'$ **then**
13 $Converge[i]=Ture$;
14 return $Converge[i]$ and $NZ[i]$ for all nodes;

a larger number of communities for a given network, and experiment results demonstrate that the performance boost of dimension level acceleration increases with the growing number of expected communities K.

The **LCNODE** algorithm collects the convergence state of each node for edge level acceleration, and updates the positions of non-zero entries in each node's representation for dimension level acceleration. After each iteration, LCNODE algorithm checks the change in each dimension of the representation of each node. If the value is lower than a predefined threshold δ, it will be updated to 0 for the convenience of dimension level acceleration, and the non-zero positions will be updated as done in line $6-8$. The over change in the representation of each node is accumulated in line 5. If the overall change is lower than a predefined threshold δ', this node will be marked as converged as in line $12-13$, which serves as conditions for edge deletion in edge level acceleration. The overall optimization algorithm is presented in Algorithm 4. The overall framework is an iterative framework. **PARAMUPDATE()** calculates k^{new} based on k^{old} from last iteration, and **LCNODE()** denotes the changes in the states of all the nodes for next iteration.

Algorithm 4: MAIN: Fast Poisson model algorithm

Input: Edge number m, $EL[m][2]$ stores observed edge list, number of communities K
Output: The community assignment θ

Random initialize k^{old} for all nodes;
Initial κ using k^{old};
Initial k^{new} to be 0;
Initial $Converge[i]$ for all node i to be -1;
Initial NZ with all dimensions non-zero;
while $|k^{old} - k^{new}|$ *not converged* **do**
  k^{new}=**PARAMUPDATE**(m,κ,EL,$Converge$,NZ);
  LCNODE(n,K,κ,k^{old},k^{new},$Converge$,NZ);
return θ for all nodes using Eq. (9);

Experiments

We adopt both synthetic networks generated by LFR "benchmark" tool (Lancichinetti and Fortunato 2009) and a range of real-world networks (Yang and Leskovec 2015) to evaluate the performance of our proposed method. Synthetic networks with ground truth community structures allow us to validate the effectiveness of the proposed method, while real-world networks at different scales enable us to compare the efficiency of proposed method with baseline methods. Moreover, we provide empirical studies on the acceleration performance of proposed two-level acceleration strategies.

Baseline methods

As summarized in the preliminary section, we adopt three fast generative community detection models as our baseline methods. The main idea for acceleration is to reduce the parameters to be calculated during the optimization process.

- Ball's method: Ball et al. speeded up its optimization through removing those edges whose end nodes' community membership are one-hot (Ball, Karrer, and Newman 2011);
- Gopalan's method: Gopalan's method reduces the time cost of the optimization process though subsampling a subset of nodes and updating parameters based on the sampled subset (Gopalan and Blei 2013);
- BIGCLAM: BIGCLAM accelerates its optimization process by replacing the loop over all nodes with the loop over its neighbor nodes when calculating the gradient for each node (Yang and Leskovec 2013).

Experiments on Synthetic networks

Our synthetic networks are generated by LFR "benchmark" tool. Three key parameters have to be set when generating

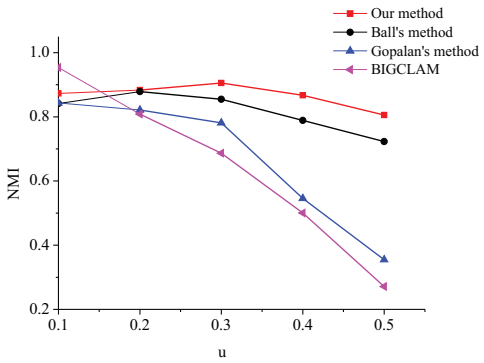

Figure 1: The NMI performance of our method compared with the baseline methods in synthetic networks. To alleviate the effect of randomness caused by stochastic optimization, we run each algorithm 10 times and select the best result as final result.

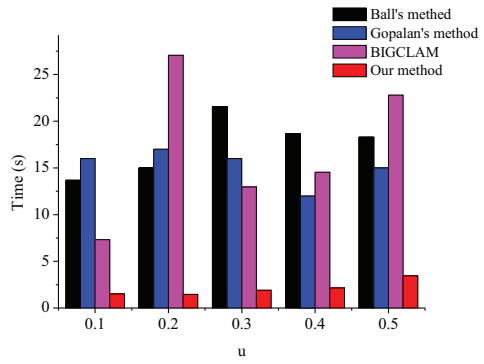

Figure 2: The time consumption of our method compared with the baseline methods in synthetic networks. Each histogram illustrates the average consumed time of 10 repeated runs of the algorithms.

benchmark networks, i.e., the power exponent of the degree distribution α, the power exponent of the community size distribution β, and the mixing parameter μ. The mixing parameter controls the fraction of a node's links that connect to nodes in other communities. We vary the value of mixing parameter μ from 0.1 to 0.5 (the networks tend to be random graph when $\mu > 0.5$) and α and β are set to be 2 and 1 respectively. We set the maximum degree as $k_{max} = \frac{n}{100}$ and the average degree $k = 0.4 \times k_{max}$, where n is the number of nodes. For community size, we set the minimum and maximum community size to be $[\frac{n}{50}, \frac{n}{250}]$. The number of nodes is set to be $5,000$ for our experiments.

For evaluation, as ground truth community labels of nodes are known, we adopt normalized mutual information (NMI) (Lancichinetti, Fortunato, and Radicchi 2008), which measures the extent to which the true and discovered community labels are consistent with each other, to measure the performance of our method and baseline methods.

The result is shown in Fig. 1. Our method consistently

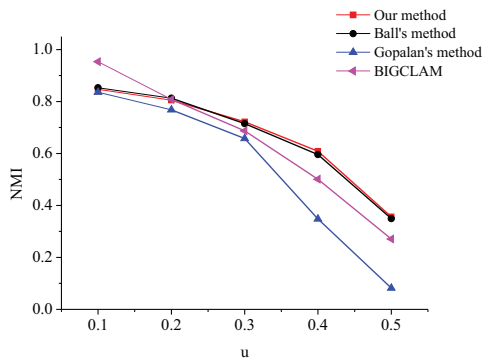

Figure 3: The overlapping community detection performance of our method compared with the baseline methods in synthetic networks. Each point is the best result of 10 repeated runs of the algorithms.

outperforms other baselines with μ varying from 0.1 to 0.5, which demonstrates the effectiveness of our proposed method. We also presented the time consumption of different methods in Fig. 2. The time consumption of our method is remarkably less than that of baseline methods when the structure of the networks are relative clear, i.e., μ is small. Along with the increase of mixing parameter μ, community structures become more confused and the community assignment of an edge is hard to decide. Consequently, the edges being removed during the optimization process is reduced, resulting in an increase of time cost for our method.

The nodes in above generated networks belong to only one community, while in real scenarios most nodes are overlapping. To further evaluate the performance of our method in detecting overlapping communities, we generate benchmark networks with overlapping nodes. In addition to above parameter settings, we require 10% of the nodes to belong to 2 communities. We also generate networks with μ varying from 0.1 to 0.5. Overlapping NMI (Lancichinetti and Fortunato 2009) is adopted to measure the performance of different methods.

The results are shown in Fig. 3. Our method still achieves best performance in overlapping settings. The time cost is presented in Fig. 4, which shows a similar tendency with that under deterministic settings.

Experiments on real networks

To evaluate the performance of our method in massive real world networks, we adopt Amazon dataset, DBLP dataset, Youtube dataset, LiveJournal dataset and Orkut dataset provided in the SNAP project for our experiments (Yang and Leskovec 2015). The statistics of these datasets are shown in Tab. 1. The network of the Amazon dataset characterizes the co-purchased products on the Amazon website, with each node representing a product, and each link indicating that those two products are frequently co-purchased. The DBLP network is a collaboration network. The nodes are authors, and two authors are connected if they have published at least one paper together. The networks in Youtube dataset, Live-

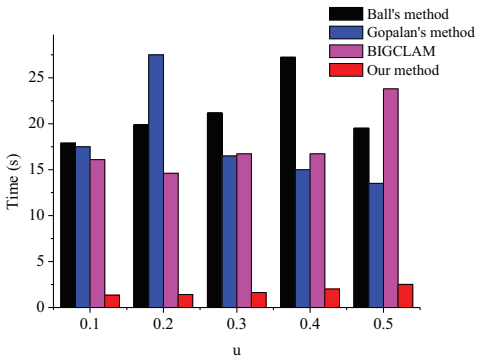

Figure 4: The time consumption of our method compared with the baseline methods in overlapping synthetic networks. Each plot is the average consumed time of the relative methods over 10 repeated runs.

Table 1: Statistics of real world networks. N: number of nodes, E: number of edges, L: number of labels, S: average nodes under the labels. M denotes a million and k denotes one thousand.

Dataset	N	E	L	S
Amazon	334,863	925,872	49k	99.86
DBLP	317,080	1,049,866	2.5k	429.79
YouTube	1,134,890	2,987,624	30k	9.75
LiveJournal	3,997,962	34,681,189	310k	40.06
Orkut	3,072,441	117,185,083	8.5M	34.86

Journal dataset and Orkut dataset are all social networks, with nodes representing users and edges representing friend relationships among users.

For those real-world networks, the ground truth community structures are unknown (Peel, Larremore, and Clauset 2016). Instead, each node is provided with a label. We adopt purity to evaluate the performances of different methods. Purity, as a measure to evaluate the results of community detection methods, is defined as

$$P = \frac{1}{K}\sum_{i=1}^{K}\max_{1\le j\le L}\frac{|C_{ij}|}{|C_i|}, \tag{11}$$

where K is the number of detected communities, L is the number of distinct labels and $|C_{ij}|$ is the number of label j in community i. Higher purity score indicates better performance.

As the number of expected communities, i.e. K, remarkably influences the time consumption, we set $K = 100$ for three small networks, i.e. Amazon network, DBLP network and Youtube network, and for the rest two large networks, K is set to be 10. The time consumption of different methods is represented in Tab. 2. Our proposed method improves the speed of Ball's method by 1 to 2 orders of magnitude, and runs consistently faster than Gopalan's method and BIGCLAM. The purity scores are presented in Tab. 3. It's easily seen that our method achieves comparable performances in community detection while consuming much less time compared with baseline methods.

Table 2: Time consumption of different methods on real-world large scale networks.

Dataset	Ball	Gopalan	BIGCLAM	Our
Amazon	396.94	819	365	**33.42**
DBLP	532.44	374	548	**45.93**
YouTube	4858.79	1246	5640	**527.23**
LiveJournal	16283.3	4170	*	**1023.04**
Orkut	61901.5	19873	*	**7538.14**

* BIGCLAM fails to find reasonable communities in these two networks.

Table 3: Purity performances on two real networks.

Dataset	Ball	Gopalan	BIGCLAM	Our
Amazon	0.166	0.165	0.51	0.167
DBLP	0.035	0.029	0.088	0.035
YouTube	0.006	0.005	0.0098	0.006
LiveJournal	0.043	0.042	*	0.042
Orkut	0.020	0.022	*	0.024

* BIGCLAM fails to find reasonable communities in these two networks.

Analysis on two acceleration strategies

In this section, we empirically explore the acceleration performance of two level acceleration strategies. Amazon network and DBLP network are utilized for experiments. We apply proposed two level acceleration strategies, i.e., edge level acceleration and dimension level acceleration, to Ball's method respectively, and evaluate the time consumption compared with original Ball's method. The result is shown in Fig. 5. We can see that the major reduction in time cost is achieved by edge level acceleration, while the contribution of dimension level acceleration becomes more significant with the increase of K.

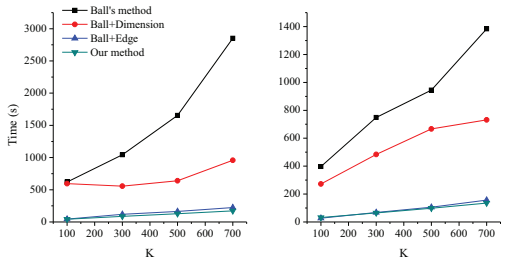

Figure 5: The performance of the two level acceleration strategies. We compare Ball's method and the two level acceleration strategies applied to Ball's method respectively with our method. The community number K varies from 100 to 700. Each histogram represents the average time consumption over 5 repeating rounds of the network under the community number K.

We further validate proposed acceleration strategies by recording the number of iterations and the number of edges to be dealt in each iteration. This experiment is conducted on Amazon network and DBLP network, with the number of expected communities being set to be 30 and 100 respectively. The results are shown in Fig. 6. Compared to the Ball's method, the number of edges to be dealt in each iteration drops exponentially, demonstrating the effectiveness of edge level acceleration strategy. Moreover, the exponential decrease in edges results in a much less number of iterations needed for convergence, which further improves the speed of the algorithm.

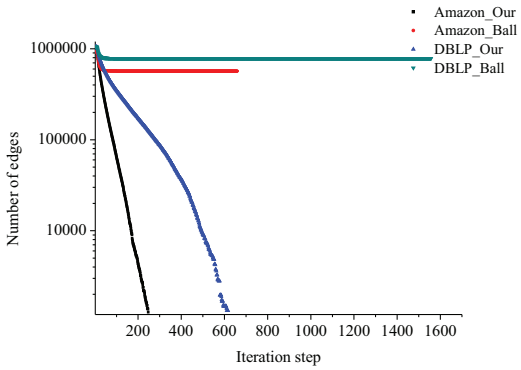

Figure 6: The performance of our method and Ball's method in the number of edges to be dealt during each iteration. We use a log-linear plot to demonstrate the change of edges. The horizontal axis is the iteration step and the vertical axis is the number of edges.

Related work

In this section, we briefly review related works, including community detection methods based on network partition and latent factor models for overlapping community detection.

Community detection based on network partition

Traditional methods for community detection are mostly based on network partition, taking each component of a partition as a community. Girvan and Newman proposed to find network partition by iteratively deleting the edges with the highest edge betweenness (Girvan and Newman 2002). Rosvall and Bergstrom detected communities by looking for a partition that minimizes the expected description length of a random walk (Rosvall and Bergstrom 2008). For community detection methods based on network partition, the key is how to evaluate the quality of a network partition. The most widely-used criterion is modularity (Newman and Girvan 2004), which is defined as the difference between the fraction of edges within communities and the expected fraction of edges within communities when all edges are randomly placed. Many methods are proposed to find communities through modularity optimization, such as simulated annealing, greedy algorithm, and spectral optimization (Newman and Girvan 2004; Blondel et al. 2008; Newman 2013).

Latent factor models for overlapping community detection

Latent factor models are used as principled and effective methods for community detection. Typical methods include: (1) mixed membership block models (Airoldi et al. 2008; Ball, Karrer, and Newman 2011; Gopalan and Blei 2013), and (2) non-negative matrix factorization methods and its variants (Ding, He, and Simon 2005; Choi 2008; Ding et al. 2006; Yuan and Oja 2005; Ren et al. 2009; Yang and Leskovec 2013; Jin et al. 2013). Mixed membership block models follow a probabilistic framework, where the link probability between nodes is fully determined by their representation. The representation of each node is defined as a multinomial distribution over communities. Such a normalization constraint on node representation indicates that a higher probability of membership to one community implies a lower probability of membership to other communities. Consequently, these models are inappropriate for modeling the mixed membership of nodes (Kim and Leskovec 2012). Non-negative matrix factorization methods take each node as a dimension of network, and regard the task of community detection as finding a low-dimensional representation of network. However, they don't guarantee that the obtained representation corresponds to communities, thus requiring some heuristic constraint to improve their interpretability.

Conclusion

In this paper, we propose to accelerate a well-performed and principled model, i.e., Poisson model, to offer an off-the-shelf method for overlapping community detection in massive networks. The acceleration strategy works in two levels: the edge level and the dimension level. Edge level acceleration keeps removing edges with both end nodes' representations being converged, and dimension level acceleration takes advantage of the sparsity in representation by keeping track of only non-zero entries in a node's representation. Experiment results demonstrate that the proposed strategies speedup the community detection method based on Poisson model by 1 to 2 orders of magnitudes, while achieving comparable accuracy at community detection.

Acknowledgments

This work was funded by the National Basic Research Program of China (973 Program) under grant numbers $2013CB329606$ and $2014CB340401$, and the National Natural Science Foundation of China under grant numbers 61472400, 61425016, 61433014 and 61602439. Huawei Shen is also funded by K.C.Wong Education Foundation, and the Youth Innovation Promotion Association CAS.

References

Airoldi, E. M.; Blei, D. M.; Fienberg, S. E.; and Xing, E. P. 2008. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research* 9(Sep):1981–2014.

Ball, B.; Karrer, B.; and Newman, M. E. 2011. Efficient and principled method for detecting communities in networks. *Physical Review E* 84(3):036103.

Blondel, V. D.; Guillaume, J.-L.; Lambiotte, R.; and Lefebvre, E. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008(10):P10008.

Cheng, X.-Q.; Ren, F.-X.; Shen, H.-W.; Zhang, Z.-K.; and Zhou, T. 2010. Bridgeness: a local index on edge significance in maintaining global connectivity. *Journal of Statistical Mechanics: Theory and Experiment* 2010(10):P10011.

Choi, S. 2008. Algorithms for orthogonal nonnegative matrix factorization. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, 1828–1832. IEEE.

Ding, C.; Li, T.; Peng, W.; and Park, H. 2006. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 126–135. ACM.

Ding, C.; He, X.; and Simon, H. D. 2005. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, 606–610. SIAM.

Fortunato, S. 2010. Community detection in graphs. *Physics reports* 486(3):75–174.

Girvan, M., and Newman, M. E. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences* 99(12):7821–7826.

Gopalan, P. K., and Blei, D. M. 2013. Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences* 110(36):14534–14539.

Jin, D.; He, D.; Hu, Q.; Baquero, C.; and Yang, B. 2013. Extending a configuration model to find communities in complex networks. *Journal of Statistical Mechanics: Theory and Experiment* 2013(09):P09013.

Karrer, B., and Newman, M. E. 2011. Stochastic blockmodels and community structure in networks. *Physical Review E* 83(1):016107.

Kim, M., and Leskovec, J. 2012. Latent multi-group membership graph model. *arXiv preprint arXiv:1205.4546*.

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

Lancichinetti, A.; Fortunato, S.; and Radicchi, F. 2008. Benchmark graphs for testing community detection algorithms. *Physical review E* 78(4):046110.

Latouche, P.; Birmele, E.; and Ambroise, C. 2012. Variational bayesian inference and complexity control for stochastic block models. *Statistical Modelling* 12(1):93–115.

Lee, D. D., and Seung, H. S. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–791.

Newman, M. E., and Girvan, M. 2004. Finding and evaluating community structure in networks. *Physical review E* 69(2):026113.

Newman, M. E., and Leicht, E. A. 2007. Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences* 104(23):9564–9569.

Newman, M. E., and Peixoto, T. P. 2015. Generalized communities in networks. *Physical review letters* 115(8):088701.

Newman, M. E. 2006. Finding community structure in networks using the eigenvectors of matrices. *Physical review E* 74(3):036104.

Newman, M. E. 2013. Spectral methods for community detection and graph partitioning. *Physical Review E* 88(4):042822.

Palla, G.; Derényi, I.; Farkas, I.; and Vicsek, T. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* 435(7043):814–818.

Peel, L.; Larremore, D. B.; and Clauset, A. 2016. The ground truth about metadata and community detection in networks. *arXiv preprint arXiv:1608.05878*.

Ren, W.; Yan, G.; Liao, X.; and Xiao, L. 2009. Simple probabilistic algorithm for detecting community structure. *Physical Review E* 79(3):036111.

Rosvall, M., and Bergstrom, C. T. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences* 105(4):1118–1123.

Shen, H.; Cheng, X.; Cai, K.; and Hu, M.-B. 2009. Detect overlapping and hierarchical community structure in networks. *Physica A: Statistical Mechanics and its Applications* 388(8):1706–1712.

Sun, B.-J.; Shen, H.; Gao, J.; Ouyang, W.; and Cheng, X. 2017. A non-negative symmetric encoder-decoder approach for community detection. In *The 26th ACM International Conference on Information and Knowledge Management*, 597–606. ACM.

Sun, B.-J.; Shen, H.-W.; and Cheng, X.-Q. 2014. Detecting overlapping communities in massive networks. *EPL (Europhysics Letters)* 108(6):68001.

Yang, J., and Leskovec, J. 2013. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, 587–596. ACM.

Yang, J., and Leskovec, J. 2015. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* 42(1):181–213.

Yuan, Z., and Oja, E. 2005. Projective nonnegative matrix factorization for image compression and feature extraction. In *Scandinavian Conference on Image Analysis*, 333–342. Springer.