NODE EMBEDDING FOR NETWORK COMMUNITY DISCOVERY

*Christy Lin, Prakash Ishwar**

ECE, Boston University, Boston, MA, USA

Weicong Ding

Technicolor Research, Los Altos, CA, USA

ABSTRACT

Neural node embedding has been recently developed as a powerful representation for supervised tasks with graph data. We leverage this recent advance and propose a novel approach for unsupervised community discovery in graphs. Through extensive experimental studies on simulated and real-world data, we demonstrate consistent improvement of the proposed approach over the current state-of-the-arts. Specifically, our approach empirically attains the information theoretic limits under the benchmark Stochastic Block Models and exhibits better stability and accuracy over the best known algorithms in the community recovery limits.

***Index Terms*—** Community Detection, Stochastic Block Model, Neural Embedding

1. INTRODUCTION

Learning a representation for nodes in a graph, also known as node embedding, has been an important tool for extracting features that can be used in machine learning problems involving graph data [1, 2, 3, 4]. Perhaps the most widely adopted node embedding is the one based on the eigendecomposition of the adjacency matrix or the graph Laplacian [2, 5, 6]. Recent advances in word embeddings for natural language processing such us [7] has inspired the development of analogous embeddings for nodes in graphs [8, 3]. These so-called "neural" node embeddings have been applied to a number of supervised learning problems such us link prediction and node classification and demonstrated state-of-the-art performance [8, 3, 4].

In contrast to applications to supervised learning problems in graphs, in this work we leverage the neural embedding framework to develop an algorithm for the *unsupervised* community discovery problem in graphs [9, 10, 11, 12]. The key idea is straightforward: learn node embeddings such that vectors of similar nodes are close to each other in the latent embedding space. Then, the problem of discovering a community in graph can be solved by finding clusters in the embedding space.

We focus on non-overlapping communities and validate the performance of the new approach through a comprehensive set of experiments on both synthetic and real-world data. Results demonstrate that the performance of the new method is consistently superior to those of spectral methods across a wide range of graph sparsity levels. In fact, we find that the proposed algorithm can empirically attain the *information-theoretic phase transition* thresholds for exact and weak recovery of communities under the Stochastic Block Model (SBM) [13, 14, 11, 15]. SBM is a canonical probabilistic model for random graphs with latent structure and has been widely used for empirical validation and theoretical analysis of community detection algorithms [16, 17, 10, 9]. In particular, when compared to the best known algorithms based on Acyclic Belief Propagation (ABP) that can provably detect communities at the information-theoretic limits [15, 11, 14], our approach has consistently better accuracy. In addition, we find that ABP is very sensitive to random initialization and exhibits high variability. In contrast, our approach is stable to both random initialization and a wide range of algorithm parameter settings.

Related works: The community detection problem has been extensively studied [9, 10, 12]. SBM was first proposed in [16, 18, 19] as a canonical model for analysis and various community detection algorithms based on it have been proposed, e.g., [20, 5, 21, 22]. Only very recently have information-theoretic limits for community recovery under the general SBM model been established [13, 11, 15].

Graph neural embedding was motivated in the famous "word2vec" algorithm for natural language processing [7]. This was extended to graphs in [3] by viewing nodes as "words" and forming "sentences" via random paths on the graph. Different ways of creating "sentences" of nodes was further explored in [8]. The embeddings have been used for supervised tasks in [3, 8] and semi-supervised tasks in [4].

2. NODE EMBEDDING FOR COMMUNITY DISCOVERY

Let $\mathcal{G}$ be a graph with n nodes and K latent communities. We focus on non-overlapping communities and denote by $\pi_i \in \{1, \ldots, K\}$ the latent community assignment for node i. Given $\mathcal{G}$, the goal is to infer the community assignment $\hat{\pi}_i$.

Our approach is to learn, in an unsupervised fashion, a low-dimensional vector representation for each node that captures its local neighborhood structure. These vectors are referred to as *node embeddings*. The premise is that if done correctly, nodes from the same community will be close to each other in the embedding space. Then, communities can

*CL and PI was supported by the U.S. NSF under Grant 1527618.

be found via clustering of the node embeddings.

We proceed as in the skip-gram framework recently developed in natural language processing [7, 3]. A document is an ordered sequence of words from a vocabulary. A w-skip-bigram is an ordered pair of words (i, j) that occur within a distance of w words from each other within a sentence in the document. A document is then viewed as a multiset $\mathcal{D}_+$ of all its w-skip-bigrams (i, j) which are generated in an IID fashion according to a *joint* probability $p((i, j))$ which is related to the word embedding vectors $\mathbf{u}_i, \mathbf{u}_j \in \mathbb{R}^d$, of words i and j respectively. Now consider a multiset $\mathcal{D}_-$ of w-skip-bigrams (i, j) which are generated in an IID fashion according to the *product* probability $p((i)) \cdot p((j))$ where the $p((i))$'s are the unigram (single word) probabilities. The w-skip-bigrams in $\mathcal{D}_+$ are labeled positive ($D = +1$) and those in $\mathcal{D}_-$ negative ($D = -1$). In the negative sampling framework [7, 3], the *posterior* probability that an observed w-skip-bigram will be labeled as positive is modeled as $p(D = +1|(i, j)) = \frac{1}{1+e^{-\mathbf{u}_j^\top \mathbf{u}_i}}$. The word embedding vectors $\{\mathbf{u}_i\}$ are then selected to maximize the posterior likelihood of observing all the positive and negative samples, i.e.,

$$\arg\max_{\mathbf{u}_i} \prod_{(i,j)\in\mathcal{D}_+} p(D = +1|(i, j)) \prod_{(i,j)\in\mathcal{D}_-} p(D = -1|(i, j))$$

$$= \arg\min_{\mathbf{u}_i} \left[\sum_{\mathcal{D}_+} \ln(1 + e^{-\mathbf{u}_j^\top \mathbf{u}_i}) + \sum_{\mathcal{D}_-} \ln(1 + e^{+\mathbf{u}_j^\top \mathbf{u}_i}) \right] \quad (1)$$

To apply this word embedding framework to node embedding, the key idea is to view nodes as words and a document as a collection of sentences that correspond to paths of nodes in the graph. We simulate r random walks (node-sentences) on $\mathcal{G}$ of fixed length ℓ starting from each node. The set $\mathcal{D}_+$ is the multiset of all node pairs (i, j) for each node i and all nodes j that are within $\pm w$ steps of node i in all the simulated paths *whenever* i appears. The set $\mathcal{D}_-$ is constructed by drawing for each node i a set of m nodes $j_1, \ldots, j_m$ in an IID manner from *all* the nodes according to the (estimated) unigram node (word) distribution across the document of node paths. Once the paths are generated, we optimize Eq. 1 using stochastic gradient descent [7]. Once the embedding vectors $\mathbf{u}_i$'s are learned, we apply K-means clustering to get the community memberships for each node. These steps are summarized in Algorithm 1. We note that since stochastic gradient descent which is used to optimize Eq. 1 can be parallelized, the proposed algorithm scales nicely to large graphs.

Algorithm 1 vec: Community Disc.via Node Embedding

Input: Graph $\mathcal{G}$, Number of communities K; Paths per node r, Length of path ℓ, Embedding dimension d, Contextual window size w
Output: Estimated Community memberships $\hat{\pi}_1, \ldots, \hat{\pi}_n$
for Each node v and $t \in \{1 \ldots r\}$ **do**
  $\mathbf{s}_{v,t} \leftarrow$ A random path of length ℓ starting from node i
end for
$\{\hat{\mathbf{u}}_i\}_{i=1}^n \leftarrow$ solve (1) with paths $\{\mathbf{s}_{v,t}\}$ and window size w.
$\hat{\pi}_1, \ldots, \hat{\pi}_n \leftarrow K$-means$(\{\hat{\mathbf{u}}_{i=1}^n\}_i, K)$

3. EXPERIMENTAL STUDY

We present a comprehensive set of experimental results in this section for both simulated and real-world graphs.[1] We compare the proposed "vec" algorithm against: (1) Spectral Clustering (SC) that is widely adopted in practice [6, 21, 20, 5] and (2) Acyclic Belief Propagation (ABP) which can achieve the information-theoretic limits in SBMs [14, 11, 15].

Metrics: We use the commonly used **Normalized Mutual Information** (NMI) [9] and **Correct Classification Rate** (CCR) metrics to measure the clustering accuracy when ground-truth community assignment is available. For real-world datasets, we also calculate the **Modularity** [23] which is an oft-used measure of community quality.

Implementation details: Unless otherwise mentioned, we set $r = 10$, $\ell = 60$, $w = 8$, and $d = 50$ for the proposed algorithm. We will discuss the choice of these algorithm parameters later. For SC we use a state-of-the-art implementation that can handle large scale sparse graphs [6]. To assure the best performance of ABP, we assume the ground-truth SBM model parameters are known, and adopted the parameters suggested in [15] which are functions of the ground-truth.

3.1. Stochastic Block Models

Generative procedure: In an SBM, a random graph with K latent communities is generated thus: (1) Each node i is randomly assigned to one community $\pi_i \in \{1, ...K\}$ with probability $\mathbf{p} = (p_1, \ldots, p_K)$; (2) For each unordered pair of nodes $\{i, j\}$, an edge is formed with probability $\mathbf{Q}_n(\pi_i, \pi_j) \in [0, 1]$. $\mathbf{Q}_n$ are the self- and cross-community connection probabilities and are typically assumed to vanish as $n \to \infty$ to capture the sparse connectivity (average node degree $\ll n$) of most real-world networks [24].

Weak and exact recovery: We consider two definitions of recovery studied in SBMs. Let accuracy α be the fraction of nodes for which the estimated communities $\hat{\pi}$ agrees with π (for the best node relabeling). Then, (1) *Weak* recovery is solvable if an algorithm can achieve accuracy $\alpha > \epsilon + \max_k p_k$, for some $\epsilon > 0$, with probability $1 - o_n(1)$, (2) *Exact* recovery is solvable if an algorithm can achieve accuracy $\alpha = 1$ with probability $1 - o_n(1)$.

Simulation setting: We simulate graphs with balanced communities $\mathbf{p} = (1/K, \ldots, 1/K)$. For $\mathbf{Q}_n$, we consider the standard *planted partition model* where $Q_n(1, 1) = \ldots = Q_n(K, K)$ and $Q_n(k, k')$ for all $k \neq k'$ are the same. We consider two scaling regimes for Q_n, (i) constant expected node degree: $Q_n(k, k) = \frac{c}{n}$, $Q_n(k, k') = \frac{c(1-\lambda)}{n}$ and (ii) logarith-

[1] Our implementation is available at `https://github.com/cy93lin/SBM_node_embedding`

mic scaling: $Q_n(k,k) = \frac{c' \ln(n)}{n}$, $Q_n(k,k') = \frac{c' \ln(n)(1-\lambda)}{n}$. Intuitively, c and c' determine sparsity while λ determines the separation between communities. Let $\mu := 1+(K-1)(1-\lambda)$. The most recent results in [13, 11, 15] can be summarized as:
Condition 1: For constant scaling, weak recovery is guaranteed if $\frac{\lambda^2 c}{K\mu} > 1$. For $K \leq 4$, the condition is also necessary.
Condition 2: For logarithmic scaling, exact recovery is solvable if and only if $\sqrt{c'} - \sqrt{c'(1-\lambda)} > \sqrt{K}$.

We choose different combinations of c, c', K, λ to explore recovery behavior around the weak and exact recovery thresholds. We set $\lambda = 0.9$ in both cases as it is typical in real-world datasets (see Sec. 3.2). For each combination of model parameters, we simulate 5 random graphs and report the mean and standard deviation of NMI and CCR.

Weak recovery phase transition: To understand behavior around the weak recovery limit, we simulated SBM graphs with $K = 2$, $n = 10000$, and $\lambda = 0.9$ at various sparsity levels c in the constant scaling regime. Weak recovery is possible if, and only if, $c > c_{\text{weak}} \approx 2.8$. The results are summarized in Fig. 1.

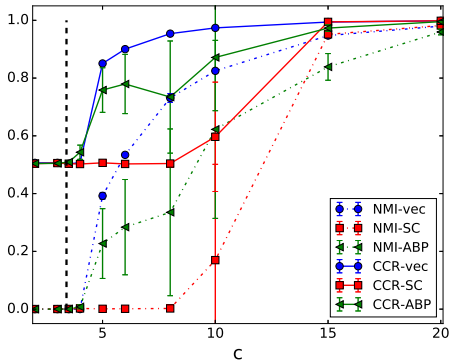

Fig. 1: NMI (dashed) and CCR (solid) versus sparsity level c for vec, SC, and ABP on SBM graphs with constant degree scaling. Here, $\mathbf{p}$ is uniform, $K = 2, \lambda = 0.9$, and $n = 10000$. The vertical dashed line is $c_{\text{weak}} = 2.8$. This figure is best viewed in color.

Figure 1 reveals that the proposed vec algorithm exhibits *weak recovery phase transition* behavior: for $c > c_{\text{weak}}$, CCR > 0.5 and when $c < c_{\text{weak}}$, CCR ≈ 0.5 (random guess). The behavior of ABP which provably achieves the weak recovery limit [15] is also shown in Fig. 1. Compared to ABP, vec has consistently superior mean clustering accuracy over the entire range of c values. In addition, we note that the variance of NMI and CCR for ABP is significantly larger than vec. This is discussed later in this section. SC, however, does not achieve weak recovery for sparse c (*cf.* Fig. 1) which is consistent with theory [20]. To reinforce our observations we also simulated SBM graphs with increasing graph size n and $K = 2, \lambda = 0.9, c = 5.0$ held fixed in the constant degree scaling regime. Since $c = 5.0 > c_{\text{weak}}$, weak recovery is possible asymptotically. As shown in Fig. 2, vec can empirically achieve weak recovery for both small and large graphs, and consistently outperforms ABP and SC.

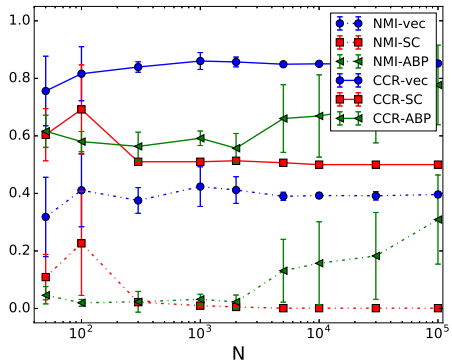

Fig. 2: NMI (dashed) and CCR (solid) versus n for vec, SC, and ABP on SBM graphs with constant degree scaling. Here, $\mathbf{p}$ is uniform, $K = 2, \lambda = 0.9, c = 5.0 > c_{\text{weak}}$.

Crossing below the weak recovery limit for $K > 4$: Here we explore the behavior of vec below the weak recovery limit for $K > 4$. As in Fig. 1, we simulated SBM graphs in the constant degree scaling regime for various sparsity levels c fixing $K = 5, \lambda = 0.9$, and $n = 1000$. In this setting, $c > c_{\text{weak}} = 8.7$ is *sufficient but not necessary* for weak recovery.

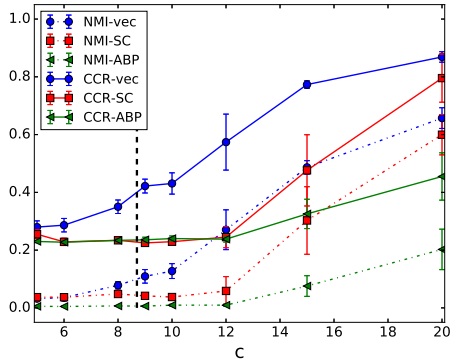

Fig. 3: NMI (dashed) and CCR (solid) versus sparsity level c for vec, SC, and ABP on SBM graphs with constant degree scaling. Here, $\mathbf{p}$ is uniform, $K = 5, \lambda = 0.9$, and $n = 1000$. The vertical dashed line is $c_{\text{weak}} = 8.6$.

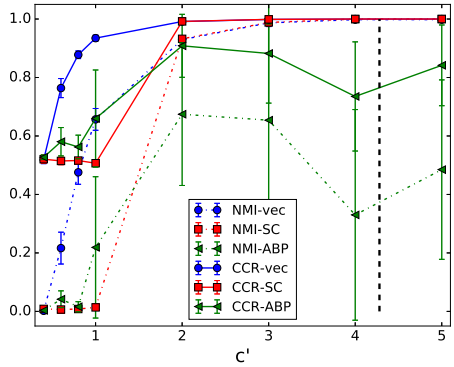

Fig. 4: NMI (dashed) and CCR (solid) versus sparsity level c' for vec, SC, and ABP on SBM graphs with logarithmic scaling. Here, $\mathbf{p}$ is uniform, $K = 2, \lambda = 0.9$, and $n = 10000$. The vertical dashed-line is at $c'_{\text{exact}} = 4.3$.

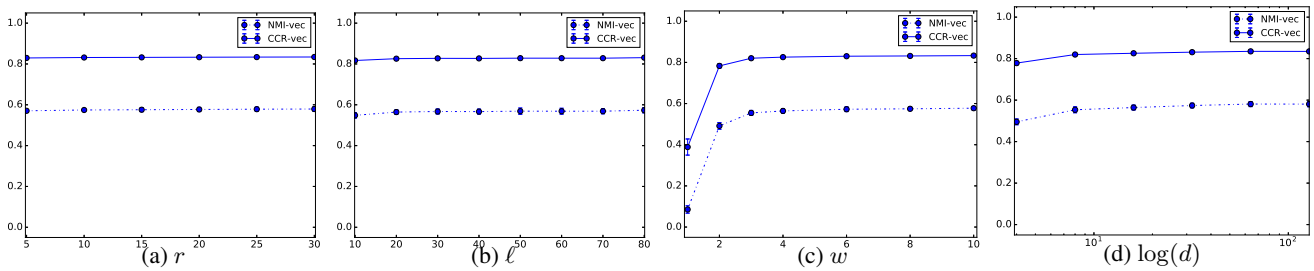

Fig. 5: NMI (dashed) and CCR (solid) of vec for different algorithm parameters: (a) the number of random paths simulated from each node p, (b) the length of each random path ℓ, (c) the size of the local window w, and (d) the embedding dimension d. In each subplot, only one parameter is varied keeping others fixed. When fixed, the default parameter values are $r = 10, \ell = 60, w = 8, d = 50$.

The results are summarized in Fig. 3.

As in Fig. 3, the vec algorithm can cross the weak recovery limit: for some $c \leq c_{\text{weak}}$, CCR $> \frac{1}{K}$ and NMI > 0 with a significant margin. Here too we observe that vec consistently outperforms ABP and SC with a large margin.

Exact recovery limit: We now turn to explore the behavior of vec near the exact recovery limit. Figure 4 plots NMI and CCR as a function of increasing sparsity level c' for SBM graphs under *logarithmic* node degree scaling fixing $K = 2$, $N = 10000$, and $\lambda = 0.9$. In this setting, exact recovery is solvable if, and only if, $c' > c'_{\text{exact}} = 4.3$. As can be seen in Fig. 4, the CCR and NMI accuracy values of vec converge to 1.0 as $c' \geq c'_{\text{exact}}$. Therefore, vec empirically achieves the exact recovery limit. We note that SC can match the performance of vec when c' is large, but cannot correctly detect communities for very sparse graphs ($c' \leq 1$). Note also that vec significantly outperforms ABP in this scaling scheme. Due to limited space we omit plots of accuracy versus n.

Parameter sensitivity of proposed approach: The performance of vec depends on the number of random paths per node r, the length of each path ℓ, the local window size w, and the embedding dimension d. We simulated SBM graphs under logarithmic scaling with $K = 5, N = 10,000, c' = 2, \lambda = 0.9$ and applied vec with different choices for r, ℓ, w, and d (*cf.* Fig. 5). While the performance of vec is robust to r, ℓ, and d, a relatively large local window size $w \geq 3$ is essential to vec (*cf.* Fig. 5(c)). This suggests that exploiting a larger graph neighborhood is critical for the success of vec.

Randomness in performance of vec and ABP: We also studied the effect of random initialization in vec and ABP. We simulated two SBM graphs as described in Table 1. For a fixed graph, we run vec and ABP 10 times and summarize the mean and standard deviation of NMI and CCR. We observe that the variance of ABP is an order of magnitude higher than vec indicating its sensitivity to initialization.

3.2. Real World Graphs

Finally, we considered two real-world datasets with ground truth (non-overlapping) community labels: the Political Blogs network [25] and the Amazon co-purchasing network [26]. Their basic statistics are summarized in Table 2. Here, $\hat{\lambda}, \hat{c'}$ are maximum likelihood estimates of λ, c' under the planted partition SBM. Note that in Amazon, the ground truth community proportions are highly unbalanced. We report NMI and Modularity values for vec, SC, and ABP applied to these datasets. We do not report CCR since it does not account for the unbalanced communities in real-world data. To apply ABP, we set the algorithm parameters using the fitted SBM parameters as suggested in [15]. As shown in Table 3, vec achieves better accuracy compared to SC and ABP. We also visualize the learned embeddings in Political Blogs using the standard t-SNE tool [27] in Fig. 6. The picture is consistent with the intuition that nodes from the same community are close to each other in the latent embedding space.

Table 1: Mean and std.dev of NMI and CCR for 10 runs on the same graph. Sim1: a graph with constant scaling, $K = 5, N = 10000, c = 15.0, \lambda = 0.9$. Sim2: a graph with logarithmic scaling, $K = 2, N = 10000, c' = 2.0, \lambda = 0.9$.

	NMI		CCR	
Expt.	vec	ABP	vec	ABP
Sim1	0.42 ± 0.004	0.14 ± 0.03	0.74 ± 0.002	0.42 ± 0.06
Sim2	0.96 ± 0.002	0.73 ± 0.37	0.99 ± 0.0003	0.93 ± 0.15

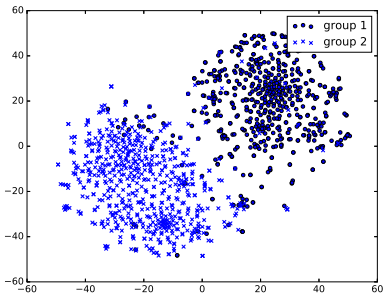

Fig. 6: t-SNE visualization of learned embedding vectors in the Blogs dataset. The markers reflect ground-truth groups.

Table 2: Summary of real world dataset parameters.

Dataset	n	K	# edges	$\hat{\lambda}$	$\hat{c'}$	$\max_k \hat{p}_k$
Blogs	$1,222$	2	$16,714$	0.89	6.9	0.52
Amazon	$334,844$	4	$925,803$	0.94	0.7	0.74

Table 3: Results on real world datasets.

	NMI			Modularity		
Data	vec	SC	ABP	vec	SC	ABP
Blogs	0.745	0.002	0.686	0.425	-0.058	0.406
Amazon	0.310	0.006	0.025	0.663	0.002	0.470

4. REFERENCES

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[2] Mikhail Belkin and Partha Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems*, 2002, pp. 585–591.

[3] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Aug. 2014, pp. 701–710.

[4] Zhilin Yang, William Cohen, and Ruslan Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," *arXiv preprint arXiv:1603.08861*, 2016.

[5] F. McSherry, "Spectral partitioning of random graphs," in *Foundations of Computer Science (FOCS), 2001 IEEE 42th Annual Symposium on*, 2001, pp. 529–537.

[6] W. Chen, Y. Song, H. Bai, C. Lin, and E. Chang, "Parallel spectral clustering in distributed systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 568–586, 2011.

[7] T. Mikolov and J. Dean, "Distributed representations of words and phrases and their compositionality," Dec. 2013.

[8] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Aug. 2016.

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

[10] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3, pp. 75–174, 2010.

[11] E. Abbe and C. Sandon, "Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery," in *Proc. of the 56th Annual Symposium on Foundations of Computer Science (FOCS)*, Sep. 2015, pp. 670–688.

[12] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.

[13] E. Mossel, J. Neeman, and A. Sly, "Belief propagation, robust reconstruction and optimal recovery of block models," in *Proc. of the 27th Conference on Learning Theory (COLT)*, 2014, pp. 356–370.

[14] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, "Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications," *Physical Review E*, vol. 84, no. 6, pp. 066106, 2011.

[15] E. Abbe and C. Sandon, "Detection in the stochastic block model with multiple clusters: proof of the achievability conjectures, acyclic bp, and the information-computation gap," in *Advances in Neural Information Processing Systems (NIPS)*, Dec. 2016.

[16] H. White, S. Boorman, and R. Breiger, "Social structure from multiple networks, blockmodels of roles and positions," *American Journal of Sociology*, pp. 730–780, 1976.

[17] M. Newman, D. Watts, and S. Strogatz, "Random graph models of social networks," *Proc. of the National Academy of Sciences*, vol. 99, no. 1, pp. 2566–2572, 2002.

[18] P. Holland, K. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social Networks*, vol. 5, no. 2, pp. 109–137, 1983.

[19] R. Boppana, "Eigenvalues and graph bisection: An average-case analysis," in *Proc. of the 28th Anuual Symposium on Foundations of Computer Science (FOCS)*, 1987, pp. 280–285.

[20] J. Lei and A. Rinaldo, "Consistency of spectral clustering in stochastic block models," *The Annals of Statistics*, vol. 43, no. 1, pp. 215–237, 2015.

[21] K. Rohe, S. Chatterjee, and B. Yu, "Spectral clustering and the high-dimensional stochastic blockmodel," *The Annals of Statistics*, vol. 39, no. 4, pp. 1878–1915, 2011.

[22] B. Hajek, Y. Wu, and J. Xu, "Achieving exact cluster recovery threshold via semidefinite programming," *IEEE Transactions on Information Theory*, vol. 62, no. 5, pp. 2788–2797, 2016.

[23] M. Newman, "Modularity and community structure in networks," *Proc. of the national academy of sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[24] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney, "Statistical properties of community structure in large social and information networks," in *Proc. of the 17th International Conference on World Wide Web (WWW)*. ACM, 2008, pp. 695–704.

[25] L. Adamic and N. Glance, "The political blogosphere and the 2004 u.s. election: Divided they blog," in *Proc. of the 3rd International Workshop on Link Discovery*, 2005, pp. 36–43.

[26] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," `http://snap.stanford.edu/data`, June 2014.

[27] L. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.