# Node Embedding via Word Embedding for Network Community Discovery

Weicong Ding, Christy Lin, and Prakash Ishwar, *Senior Member, IEEE*

***Abstract*—Neural node embeddings have recently emerged as a powerful representation for supervised learning tasks involving graph-structured data. We leverage this recent advance to develop a novel algorithm for unsupervised community discovery in graphs. Through extensive experimental studies on simulated and real-world data, we demonstrate that the proposed approach consistently improves over the current state-of-the-art. Specifically, our approach empirically attains the information-theoretic limits for community recovery under the benchmark Stochastic Block Models for graph generation and exhibits better stability and accuracy over both Spectral Clustering and Acyclic Belief Propagation in the community recovery limits.**

***Index Terms*—Acyclic Belief Propagation, Community Detection, Neural Embedding, Spectral Clustering, Stochastic Block Model.**

## I. 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-structured data [1]–[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 as [7] has inspired the development of analogous embeddings for nodes in graphs [3], [8]. 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 [3], [4], [8].

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]–[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 communities in a 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) [11], [13]–[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 [9], [10], [16], [17]. 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 [11], [14], [15], 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.

Our implementation and scripts to recreate all the results in this paper are available at https://github.com/cy93lin/SBM_node_embedding

## II. Related Works

The community detection problem has been extensively studied in the literature [9], [10], [12], [18]. It has important applications in various real-world networks that are encountered in sociology, biology, statistics and computer science. One way to systematically evaluate the performance of a community detection algorithm and establish theoretical guarantees is to consider a generative model for graphs with a latent community structure. The most classic model is the widely-adopted Stochastic Block Model. SBM was first proposed in [16], [19], [20] as a canonical model for analysis and various community detection algorithms based on it have been proposed, e.g., [5], [21]–[23]. Among these approaches, algorithms that are based on the graph spectrum and semidefinite programming relaxations have been extensively analyzed [22], [23]. Only very recently have information-theoretic limits for community recovery under the general SBM model been established [11], [13], [15]. In [11], [15], a belief-propagation based algorithm has been shown to asymptotically detect the latent communities in an SBM and achieve the information-theoretic limits. It has also been shown that graph spectrum based algorithms cannot achieve the information-theoretic limits for recovering communities in SBM models [11].

Graph neural embeddings were motivated in the famous "word2vec" algorithm for natural language processing [7], [24]. In these works, each word in the vocabulary is represented as a low-dimensional vector. These representations

W. Ding is with Technicolor Research, Los Altos, CA, USA e-mail: Weicong.Ding@technicolor.com.

C. Lin is with the Division of Systems Engineering, Boston University, Boston, MA, 02215 USA e-mail: cy93lin@bu.edu.

P. Ishwar is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA, 02215 USA e-mail: pi@bu.edu.

Manuscript received November 1, 2016.

are then learned in an un-supervised fashion using large text corpora such as Wikipedia articles. Neural embeddings for words 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]. These embeddings have been used for supervised learning tasks in [3], [8] and semi-supervised learning tasks in [4].

Our work is most closely related to [3], [8]. While [3], [8] make use of node embeddings in supervised learning problems such as node attribute prediction and link prediction, this paper focuses on the unsupervised community detection problem. We also explore the information-theoretic limits for community recovery under the classic SBM generative model and empirically show that our algorithm can achieve these limits.

## III. 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 be found via clustering of the node embeddings.

In order to construct the node embedding, we proceed as in the skip-gram-based negative sampling framework for word embedding which was recently developed in the natural language processing literature [3], [7]. A document is an ordered sequence of words from a fixed 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 independent and identically distributed (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, in $d$-dimensional Euclidean space.

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 unigram probabilities can be approximated via the empirical frequencies of individual words (unigrams) in the document.

The $w$-skip-bigrams in $\mathcal{D}_+$ are labeled as positive samples ($D = +1$) and those in $\mathcal{D}_-$ are labeled as negative samples ($D = -1$). In the negative sampling framework [3], [7], the *posterior* probability that an observed $w$-skip-bigram will be labeled as positive is modeled as follows

$$p(D = +1|(i, j)) = 1 - p(D = -1|(i, j)) = \frac{1}{1 + e^{-\mathbf{u}_j^\top \mathbf{u}_i}} \quad (1)$$

Under this model, the likelihood ratio $p((i, j)|D = +1)/p((i, j)|D = -1)$, becomes proportional to $e^{\mathbf{u}_j^\top \mathbf{u}_i}$. Thus the negative sampling model posits that the ratio of the odds of observing a $w$-skip-bigram from a bonafide document to the odds of observing it due to pure chance is exponentially related to the inner product of the underlying embedding vectors of the nodes in the $w$-skip-bigram. The word embedding vectors $\{\mathbf{u}_i\}$ which are parameters of the posterior distributions are then selected to maximize the posterior likelihood of observing all the positive and negative samples, i.e.,

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

Substituting from Eq. (1) and taking negative log, this reduces to

$$\arg\min_{\mathbf{u}_i} \left[ \sum_{(i,j)\in\mathcal{D}_+} \log(1 + e^{-\mathbf{u}_j^\top \mathbf{u}_i}) + \sum_{(i,j)\in\mathcal{D}_-} \log(1 + e^{+\mathbf{u}_j^\top \mathbf{u}_i}) \right] \quad (2)$$

In order to apply this word embedding framework to node embedding, the key idea is to view nodes as words and and a document as a collection of sentences that correspond to paths of nodes in the graph. To operationalize this idea, we generate multiple paths (sentences) by performing random walks of suitable lengths starting from each node. Specifically, we simulate $r$ random walks on $\mathcal{G}$ of fixed length $\ell$ starting from each node. The set $\mathcal{D}_+$ is then taken to be 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 $\mathcal{D}_-$ (negative samples) is constructed as a multiset using the following approach: for each node pair $(i, j)$ in $\mathcal{D}_+$, we append $m$ node pairs $(i, j_1), \ldots, (i, j_m)$ to $\mathcal{D}_-$, where the $m$ nodes $j_1, \ldots, j_m$ are drawn in an IID manner from *all* the nodes according to the estimated unigram node (word) distribution across the document of node paths. Once $\mathcal{D}_+$ and $\mathcal{D}_-$ are generated, we optimize Eq. (2) 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. (2) can be parallelized, the proposed algorithm scales nicely to large graphs.

**Algorithm 1** vec: Community Discovery via Node Embedding

```
Input: Graph $\mathcal{G}$, Number of communities $K$; Paths per node $r$, Length of path $\ell$, 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 $\ell$ starting from node $i$
end for
$\{\hat{\mathbf{u}}_i\}_{i=1}^n \leftarrow$ solve (2) with paths $\{\mathbf{s}_{v,t}\}$ and window size $w$.
$\hat{\pi}_1, \ldots, \hat{\pi}_n \leftarrow K$-means($\{\hat{\mathbf{u}}_{i=1}^n\}_i, K$)
```

Our implementation is available at https://github.com/cy93lin/SBM_node_embedding. In the rest of this paper, we compare the proposed "vec" algorithm against two baseline approaches: (1) Spectral Clustering (SC) that is widely adopted in practice [5], [6], [21], [22] and (2) Acyclic Belief Propagation (ABP) which can achieve the information-theoretic limits in SBMs [11], [14], [15].

**Implementation details**: Unless otherwise mentioned, we set $r = 10$, $\ell = 60$, $w = 8$, and $d = 50$ for the proposed algorithm. In Sec. IV-E we will conduct a detailed assessment of how different choices of these algorithm hyper-parameters impacts the performance of our proposed algorithm. In particular, our results will demonstrate that the performance of vec remains remarkably stable across a wide range of values of these hyper parameters indicating that vec is very robust to different choices of hyper parameters. We set the number of negative samples per observation $m = 5$ as suggested in [7]. For SC we use a state-of-the-art implementation that can handle large scale sparse graphs [6]. In order to assure the best performance for ABP, we assume that the ground-truth SBM model parameters are known, and adopt the parameters suggested in [15] which are functions of the ground-truth parameters. In other words, we allow the competing algorithm ABP additional advantages that are not used in our proposed vec algorithm.

## IV. Experiments with the Stochastic Block Model

In this section, we present and discuss a comprehensive set experimental results on graphs that are synthetically generated using a Stochastic Block Model (SBM). SBMs have been widely used for both theoretical analysis as well as empirical validation of community detection algorithms [4], [9], [10], [16], [17].

### *A. The Stochastic Block Model and Simulation Framework*

**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 community membership probabilities given by the probability vector $\mathbf{p} = (p_1, \ldots, p_K)$; (2) For each unordered pair of nodes $\{i, j\}$, an edge is formed with probability $\mathbf{Q}_n(\pi_i, \pi_j) \in [0, 1]$. Here, $\mathbf{Q}_n$ 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 [18].

**Weak and exact recovery**: We consider two definitions of recovery studied in SBMs. Let accuracy $\alpha$ be the fraction of nodes for which the estimated communities $\hat{\pi}$ agree with $\pi$ (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 and scaling regimes**: In the bulk of our experiments, we synthesize graphs with *balanced* communities, i.e., $\mathbf{p} = (1/K, \ldots, 1/K)$, and *equal* community connection probabilities. Specifically, 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. In Sec. IV-C, we study how *unbalanced* communities and unequal connectivities affect the performance of different algorithms.

We consider two commonly studied scaling regimes and parameter settings for $Q_n$, namely

$(i)$ *constant expected node degree scaling:* $Q_n(k, k) = \frac{c}{n}$, $Q_n(k, k') = \frac{c(1-\lambda)}{n}$ and

$(ii)$ *logarithmic expected node degree scaling:* $Q_n(k, k) = \frac{c' \ln(n)}{n}$, $Q_n(k, k') = \frac{c' \ln(n)(1-\lambda)}{n}$.

Intuitively, $c$ and $c'$ influence the degree of sparsity whereas $\lambda$ controls the degree of separation between communities. Let $\mu := 1 + (K - 1)(1 - \lambda)$. The constant expected node degree scaling regime is more challenging for community recovery than the logarithmic expected node degree regime. The most recent results in [11], [13], [15] when specialized to the *planted partition model* can be summarized as follows:

*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 $c, c', K, \lambda$ in order 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 (*cf.* Sec. V). For each combination of model parameters, we synthesize $5$ random graphs and report the mean and standard deviation of all the performance metrics (discussed next).

**Performance Metrics**: In all our experiments, we adopt the commonly used **Normalized Mutual Information** (NMI) [9] and **Correct Classification Rate** (CCR) metrics to measure the clustering accuracy since ground-truth community assignments are available. In order to compare the *overall* performance of different algorithms across a large number of different simulation settings, we also compute the NMI and CCR **Performance Profile** (PP) curves [25] across a set of 250 distinct experiments. These curves provide a global performance summary of the compared algorithms.

Table I provides a bird's-eye view of all our experiments with synthetically generated graphs. The table indicates all key problem parameters that are held fixed as well as those which are varied. It also summarizes the main conclusion of each experimental study and includes pointers to the appropriate figures and subsections where the results can be found.

### *B. Weak Recovery Phase Transition*

To understand behavior around the weak recovery limit, we synthesized SBM graphs with $K = 2$, $n = 10000$, and $\lambda = 0.9$ at various sparsity levels $c$ in the constant scaling regime. For these parameter settings, weak recovery is possible if, and only if, $c > c_{\text{weak}} \approx 2.8$ (*cf. Condition 1*). The results are summarized in Fig. 1.

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 $\approx 0.5$ (random guess). This behavior can be also observed through the NMI metric. 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

TABLE I: Summary of all experiments with synthetic graphs

| # | Fig., Table & Sec. | Scaling regime | Sparsity $c$ or $c'$ | Graph size $n$ | Balanced $\mathbf{p}$ & unifrm. diag$(\mathbf{Q})$? | $K$ | Main observation |
|---|---|---|---|---|---|---|---|
| 1 | Fig. 1, Sec. IV-B | constant | variable | $1e4$ | yes | 2 | vec exhibits weak recovery phase transition behavior |
| 2 | Fig. 2, Sec. IV-B | constant | fixed | variable | yes | 2 | vec achieves weak recovery asymptotically when conditions are satisfied |
| 3 | Fig. 3, Sec. IV-B | constant | variable | $1e3$ | yes | 5 | vec can cross the weak recovery limit for $K > 4$ |
| 4 | Fig. 4, Sec. IV-B | constant | fixed | $1e4$ | yes | variable | vec is robust to the number of communities $K$ |
| 5 | Fig. 5, Sec. IV-C | constant | fixed | $1e4$ | unbalanced $\mathbf{p}$ | 2 | vec is robust to unbalanced communities |
| 6 | Fig. 6, Sec. IV-C | constant | fixed | $1e4$ | non-uniform diag$(\mathbf{Q})$ | 2 | vec is robust to unequal connectivities |
| 7 | Fig. 7, Sec. IV-D | logarithmic | variable | $1e4$ | yes | 2 | vec attains the exact recovery limit |
| 8 | Fig. 8, Sec. IV-D | logarithmic | fixed | variable | yes | 2 | vec achieves exact recovery asymptotically when conditions are satisfied |
| 9 | Fig. 9, Sec. IV-E | logarithmic | fixed | $1e4$ | yes | 5 | vec is robust to algorithm parameters |
| 10 | Table II, Sec. IV-E | both | fixed | $1e4$ | yes | 2, 5 | vec is robust to randomness in creating paths |
| 11 | Fig. 10, Sec. IV-F | constant | variable | variable | yes | variable | vec consistently outperforms baselines across 240 experiments |

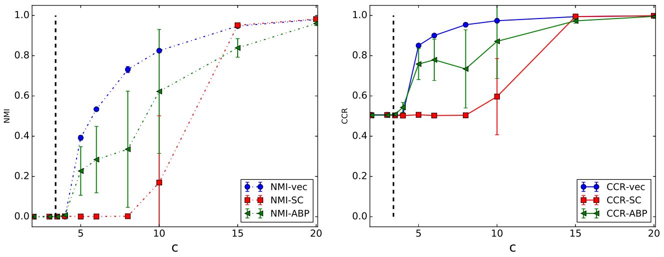


Fig. 1: NMI (left sub-figure, dashed curves) and CCR (right sub-figure, solid curves) versus sparsity level $c$ for vec, SC, and ABP on SBM graphs with constant degree scaling. Here, $\mathbf{p}$ is uniform, $K = 2, \lambda = 0.9$, and $n = 10000$. The vertical dashed line is $c_{\text{weak}} = 2.8$.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

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 [21].

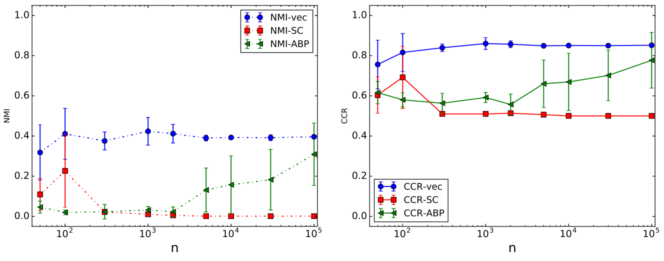


Fig. 2: NMI (left sub-figure, dashed curves) and CCR (right sub-figure, solid curves) versus $n$ for vec, SC, and ABP on SBM graphs with constant degree scaling. Here, $\mathbf{p}$ is uniform, $K = 2, \lambda = 0.9$, and $c = 5.0 > c_{\text{week}}$.

In order to reinforce our observations, we also synthesized SBM graphs with increasing graph size $n$ with $K = 2, \lambda = 0.9, c = 5.0$ held fixed in the constant degree scaling regime. Since $c = 5.0 > c_{\text{weak}}$, weak recovery is possible asymptotically as $n \to \infty$. As shown in Fig. 2, vec can empirically achieve weak recovery for both small and large graphs, and consistently outperforms ABP and SC. While ABP can provably achieve weak recovery asymptotically, its performance on smaller graphs is poor.

**Crossing below the weak recovery limit for $K > 4$:**

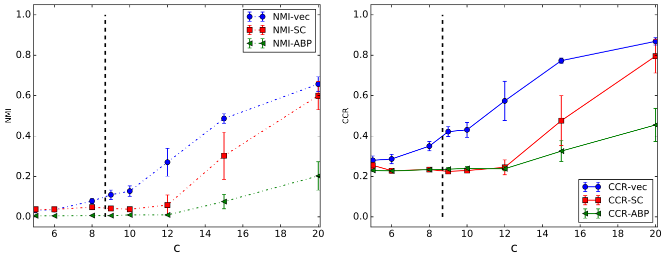


Fig. 3: NMI (left sub-figure, dashed curves) and CCR (right sub-figure, solid curves) versus sparsity level $c$ for vec, SC, and ABP on SBM graphs with constant degree scaling. Here, $\mathbf{p}$ is uniform, $K = 5, \lambda = 0.9$, and $n = 1000$. The vertical dashed line is $c_{\text{weak}} = 8.6$.

Here we explore the behavior of vec below the weak recovery limit for $K > 4$ since to-date there are no necessary and sufficient weak recovery bounds established for this setting (i.e., $K > 4$). Similar to Fig. 1, we synthesized 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. The results are summarized in Fig. 3.

As can be seen 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.

**Weak recovery with increasing number of communities $K$:** Next we consider the performance of vec as the number of communities $K$ increases. In particular, we synthesize *planted partition model* SBMs in the constant scaling regime with

$c = 10$, $\lambda = 0.9$, $N = 10000$, and uniform $\mathbf{p}$. Recall that according to *Condition 1* for weak recovery [15], weak recovery is possible if $\lambda^2 c > K(1 + (K-1)(1-\lambda))$. Hence as $K$ increases, recovery becomes impossible. For the above parameter settings, we have $K \le K_{\text{weak}} = 5$.

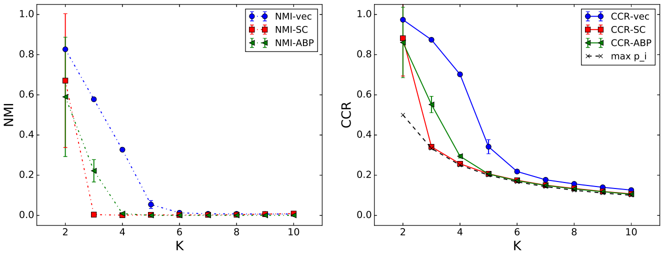


Fig. 4: NMI (left sub-figure, dashed curves) and CCR (right sub-figure, solid curves) versus number of communities $K$ for vec, SC, and ABP on SBM graphs with constant degree scaling. Here, $\mathbf{p}$ is uniform, $\lambda = 0.9$, $c = 10$, and $n = 10000$. Weak recovery if possible if $K \le K_{\text{weak}} = 5$. The black dashed curve in the CCR sub-figure is the plot of the maximum community weight $\max_k p_k$ versus $K$. It is the CCR of the rule which assigns the *apriori* most likely community to *all* nodes.

Figure 4 summarizes the performance of vec, SC, and ABP as a function of $K$. The performance of the three algorithms can be compared more straightforwardly by focusing on the NMI metric (plots in the upper sub-figure of Fig. 4). Similar to all the previous studies of this section, the proposed vec algorithm can empirically achieve weak-recovery whenever the information-theoretic sufficient conditions are satisfied, i.e., NMI $> 0$ with a significant margin for all $K \le K_{\text{weak}} = 5$. We note that in terms of the CCR metric, a “weak” recovery corresponds to CCR $> \max_k p_k \approx 1/K$ since $\max_k p_k$ it is the CCR of the rule which assigns the *apriori* most likely community label to all nodes. This is empirically attained by the vec algorithm as illustrated in plots in the bottom sub-figure of Fig. 4.

Note that the performance of SC drops significantly beyond $K > 2$. We also note that the CCR performance margin between ABP, which is a provably asymptotically consistent algorithm, and the best constant guess rule (CCR $= 1/K$) is much smaller than for vec.

### C. Weak Recovery with Unbalanced Communities and Unequal Connectivities

Here we study how unbalanced communities and unequal connectivities affect the performance of the proposed vec algorithm. By unbalanced communities we mean an SBM in which the community membership weights $\mathbf{p}$ are not uniform. In this scenario, some clusters will be more dominant than the others making it challenging to detect rare clusters. By unequal connectivities we mean an SBM in which $\mathbf{Q}_n(k,k)$ is not the same for all $k$ or $\mathbf{Q}_n(k,k')$ is not the same for all $k \neq k'$. Since it is unwieldy to explore all types of unequal connectivities, our study only focuses on unequal *self*-connectivities, i.e., $\mathbf{Q}_n(k,k)$ is not the same for all $k$. In this scenario, the densities of different communities will be different making it challenging to detect the sparser communities.

Here we compare NMI and CCR curves only for vec and ABP but not SC. When communities are unbalanced or the self-connectivities are unequal we observed that SC takes an inordinate amount to time to terminate. We decided therefore to omit NMI and CCR plots for SC from the experimental results in this subsection.

**Unbalanced Communities**: We first show results on SBMs with nonuniform community weights $\mathbf{p}$. For simplicity, we consider SBMs with $K = 2$ communities and set $p_1 = \gamma \in [0.5, 1)$. Then, $p_2 = 1 - \gamma$. For the other parameters we set $c = 8, N = 10000, \lambda = 0.9$. From the general weak recovery conditions for nonuniform $\mathbf{p}$ in [15],[1] it can be shown that as $\gamma \to 1$, the threshold for guaranteed weak-recovery will be broken. Specifically, for the above parameter settings, it can can be shown that $\gamma$ must not exceed $\gamma_{\text{weak}} \approx 0.65$ for weak recovery.

We summarize the results in Fig. 5. For comparison, the right (CCR) sub-figure of Fig. 5 also shows the plot of $\max_k p_k = \gamma$ which is the CCR of the rule which assigns the *apriori* most likely community to *all* nodes. From the figure it is evident that unlike ABP, the CCR performance of vec remains stable across a wide range of $\gamma$ values indicating that it can tolerate significantly unbalanced communities.

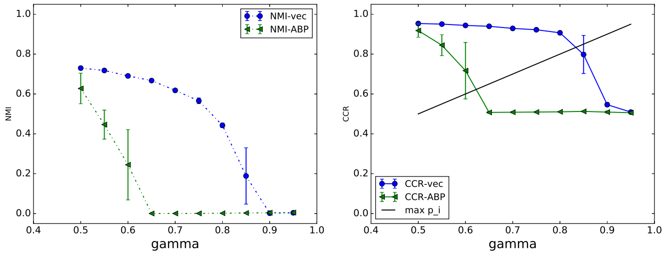


Fig. 5: NMI (left sub-figure, dashed curves) and CCR (right sub-figure, solid curves) versus maximum community weight $\gamma$ for vec and ABP on SBM graphs with constant degree scaling. Here, $K = 2$, $\mathbf{p} = [\gamma, 1-\gamma]$, $\lambda = 0.9$, $c = 8$, and $n = 10000$. Weak recovery if possible if $\gamma \le \gamma_{\text{weak}} = 0.65$. The solid unmarked curve in the CCR sub-figure indicates the maximum community weights $\max_k p_k = \gamma$ in each setting. It is the CCR of the rule which assigns the *apriori* most likely community to *all* nodes.

**Unequal Community Connectivity**: We next consider the situation in which the connectivity constants of different communities are distinct. For simplicity, we consider SBMs with $K = 2$ communities and balanced weights $p_1 = p_2 = 0.5$. We focus on the constant scaling regime and set

$$Q_n = \begin{bmatrix} \frac{c}{n} & \frac{c(1-\lambda)}{n} \\ \frac{c(1-\lambda)}{n} & \frac{c\beta}{n} \end{bmatrix}$$

Here, $\beta \in (0, 1]$ determines the relative densities of communities 1 and 2. For the other model parameters, we set $c = 8$, $\lambda = 0.9$ as in the previous subsection. From the general weak recovery conditions for unequal community connectivity in [15], it can be shown that weak recovery requires that $\beta \ge \beta_{\text{weak}} \approx 0.60$. We summarize the results in Fig. 6. From the figure it is once again evident that the performance of vec

[1] *Condition 1* in Sec. IV-A assumes uniform $\mathbf{p}$.

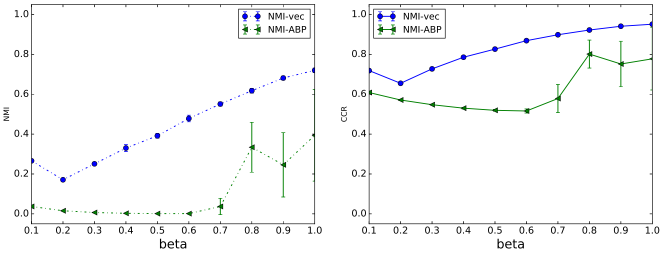


Fig. 6: NMI (left sub-figure, dashed curves) and CCR (right sub-figure, solid curves) versus community connectivity $\beta$ for vec and ABP on SBM graphs with constant degree scaling. Here, $\mathbf{p}$ is uniform, $\lambda = 0.9$, $c = 8$ and $n = 10000$. Weak recovery if possible if $\beta \leq \beta_{\text{weak}} = 0.60$. The dashed curve in the CCR sub-figure is the maximum community weight $\max_k p_k$ in each setting. It is the CCR of the rule which assigns the *apriori* most likely community to *all* nodes.

remains stable across a wide range of $\beta$ values when compared to ABP.

### D. Exact Recovery Limits

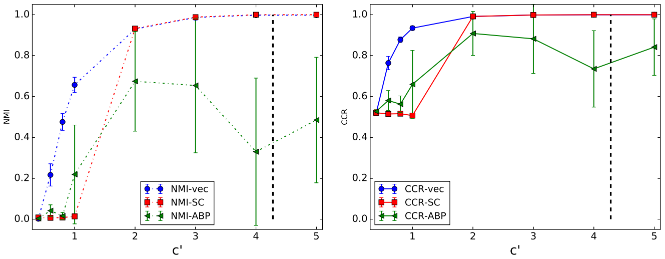


Fig. 7: NMI (left sub-figure, dashed curves) and CCR (right sub-figure, solid curves) versus sparsity level $c'$ for vec, SC, and ABP on SBM graphs with logarithmic scaling. Here, $\mathbf{p}$ is uniform, $K = 2, \lambda = 0.9$, and $n = 10000$. The vertical dashed-line is at $c'_{\text{exact}} = 4.3$.

We now turn to explore the behavior of vec near the exact recovery limit. Figure 7 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, $c' > c'_{\text{exact}} \approx 4.3$ (*cf. Condition 2* in Sec. IV-A). As can be seen in Fig. 7, the CCR and NMI values of vec converge to $1.0$ as $c'$ increases far beyond $c'_{\text{exact}}$. Therefore, vec empirically attains 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.

We also compared the behavior of vec, ABP, and SC algorithms for increasing graph sizes $n$. We set $K = 2, \lambda = 0.9$, and $\mathbf{p} \sim$ uniform. Figure 8 illustrates the performance of vec, SC, and ABP as a function of the number of nodes $n$ for three different choices of $c' : c' = 0.6, 2.5$, and $4.5$. Since exact recovery requires $c' \geq c'_{\text{exact}} \approx 4.3$, only the third choice of $c'$ guarantees exact recovery asymptotically.

As can be seen in Fig. 8, when $c'$ is above the exact recovery condition (see Fig. 8 $(c)$), the proposed algorithm vec can achieve exact recovery, i.e., CCR $\approx 1$ and NMI $\approx 1$. In this setting, the proposed vec algorithm can be observed to achieve exact recovery even when the number of nodes $n$ is relatively small. On the other hand, when $c'$ is below the exact recovery condition (see Figs. 8 $(a)$ and $(b)$), as $n$ increases, the accuracy of vec increases and converges to a value that is somewhere between random guessing (CCR $= 0.5$, NMI $= 0.0$) and exact recovery (CCR $= 1.0$, NMI $= 1.0$).

We note that among the compared baselines, the performance of SC is similar to that of vec when $c'$ is large (relatively dense graph) but its performance deteriorates when $c'$ is small (sparse graph). As shown in Fig. 8 $(a)$, when the SBM-synthesized graph is relatively sparse, the performance of SC is close to a random guess while the performance of vec and ABP increases with the number of nodes $n$. This observation is consistent with known theoretical results [11], [22].

### E. Robustness of proposed approach

**Parameter sensitivity**: The performance of vec depends on the number of random paths per node $r$, the length of each path $\ell$, the local window size $w$, and the embedding dimension $d$. We synthesized SBM graphs under logarithmic scaling with $K = 5, N = 10,000, c' = 2, \lambda = 0.9$ and applied vec with different choices for $r$, $\ell$, $w$, and $d$. The results are summarized in Fig. 9. While the performance of vec is remarkably insensitive to $r$, $\ell$, and $d$ across a wide swathe of values, a relatively large local window size $w \geq 3$ appears to be essential for attaining good performance (*cf.* Fig. 9(c)). This suggests that incorporating a larger graph neighborhood is critical to the success of vec.

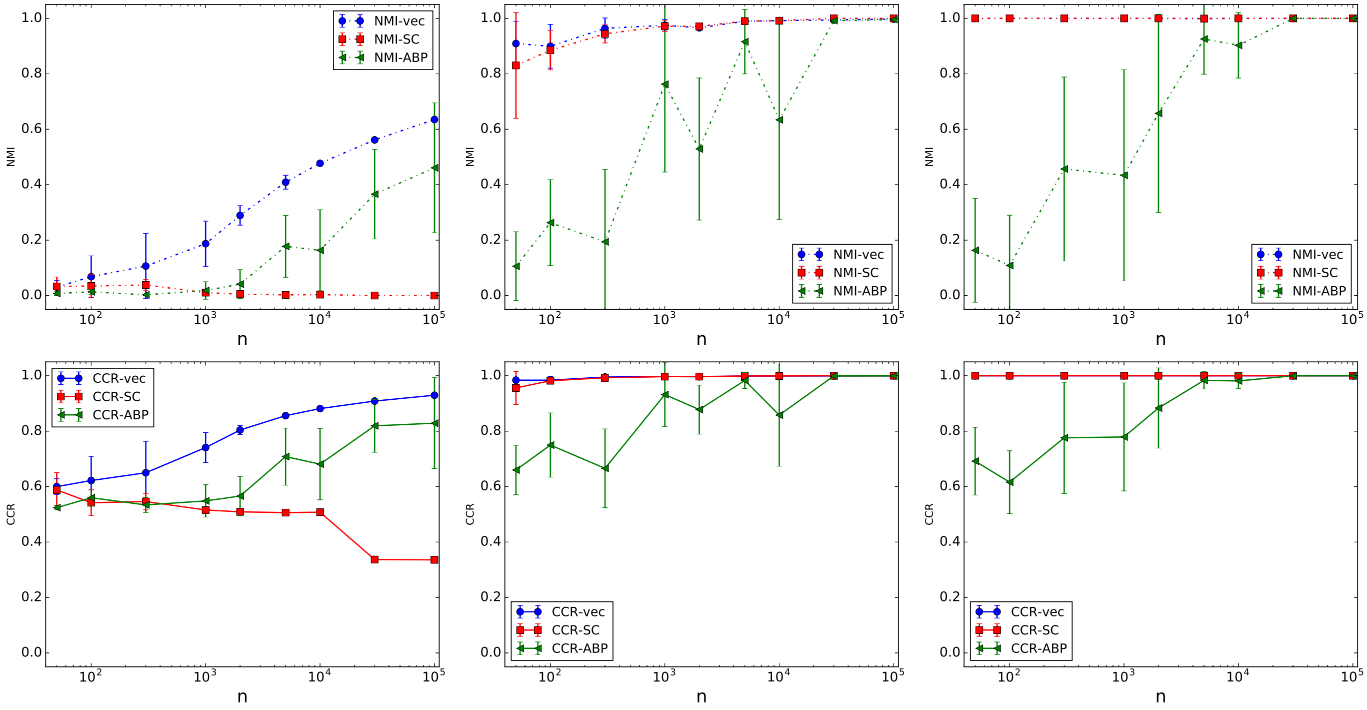
**Effect of random initialization in vec and ABP**: We also studied the effect of random initialization in vec and ABP. We synthesized two SBM graphs as described in Table II. For a fixed graph, we run vec and ABP 10 times and summarize the mean and standard deviation values of NMI and CCR. We observe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

TABLE II: Means and standard deviations 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 | $0.42 \pm 0.004$ | $0.14 \pm 0.03$ | $0.74 \pm 0.002$ | $0.42 \pm 0.06$ |
| Sim2 | $0.96 \pm 0.002$ | $0.73 \pm 0.37$ | $0.99 \pm 0.0003$ | $0.93 \pm 0.15$ |

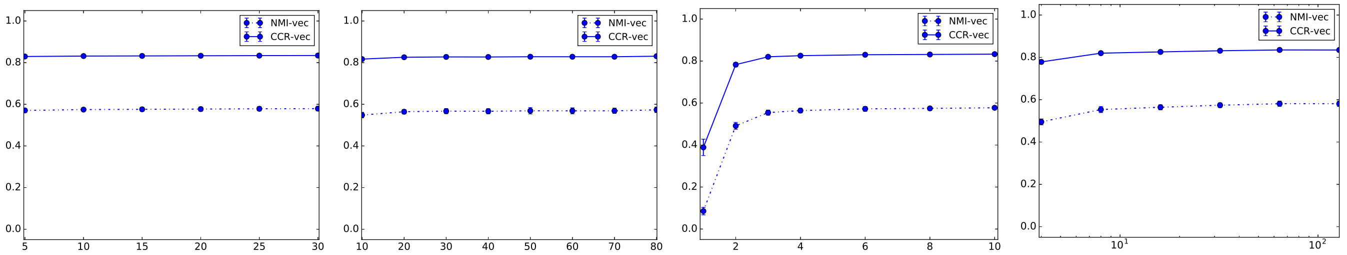
### F. Summarizing overall performance via Performance Profiles

So far we presented and discussed the performance of vec, ABP, and SC across a wide range of parameter settings. All results indicate that vec matches or outperforms both ABP and SC in almost all scenarios. In order to summarize and compare of the *overall* performance of all three algorithms across the wide range of parameter settings that we have considered, we adopt the commonly used Performance Profile [25] as a "global" evaluation metric. Formally, let $\mathcal{P}$ denote a set of experiments and $a$ a specific algorithm. Let $Q(a, e)$ denote



(a) $c' = 0.6$ (b) $c' = 2.5$ (c) $c' = 4.5$

Fig. 8: NMI (top sub-figure, dashed curves) and CCR (bottom sub-figure, solid curves) versus number of nodes $n$ for vec, SC, and ABP on SBM graphs with logarithmic scaling. Here, $\mathbf{p}$ is uniform, $K = 2$, and $\lambda = 0.9$. In subplots $(a)$ and $(b)$, $c' < c'_{\text{exact}}$ while in subplot $(c)$, $c' > c'_{\text{exact}}$. We note that in subplot $(c)$, the curves of vec and SC are on top of each other since they have very similar performance in this setting.



(a) $r$ (b) $\ell$ (c) $w$ (d) $\log(d)$

Fig. 9: NMI (dashed curves) and CCR (solid curves) as a function of different algorithm parameters of vec on SBM graphs: (a) the number of random paths simulated from each node $r$, (b) the length of each random path $\ell$, (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 value of a performance metric attained by an algorithm $a$ in experiment $e$ where higher $Q$ values correspond to better performance. Then the performance profile of $a$ at $\tau \in [0, 1]$ is the fraction of the experiments in which the performance of $a$ is at least a factor $(1 - \tau)$ times as good as the best performing algorithm in that experiment, i.e.,

$$PP_a(\tau) := \frac{|\{e : Q(a, e) \geq (1 - \tau) \max_{a'} Q(a', e)\}|}{|\mathcal{P}|} \quad (3)$$

The Performance Profile is thus an empirical cumulative distribution function of an algorithm's performance relative to the best-performing algorithm in each experiment. We calculate $PP_a(\tau)$ for $\tau \in (0, 1)$. The higher a curve corresponding to an algorithm, the more often it outperforms the other algorithms.

For simplicity, we *only* consider the simulation settings for the *planted partition model* SBMs in the constant degree scaling regime. We set $c \in \{2, 5, 10, 15\}$, $K \in \{2, 5, 10\}$, and $N \in \{1e2, 1e3, 1e4, 1e5\}$. For each combination of settings $(c, K, N)$, we conduct 5 independent random repetitions of the experiment. Thus overall the Performance Profile is calculated based on 240 experiments.

Figure 10 shows the performance profiles for both NMI and CCR metrics. From the figure it is clear that vec dominates both ABP and SC and that ABP and SC have similar performance across many experiments.

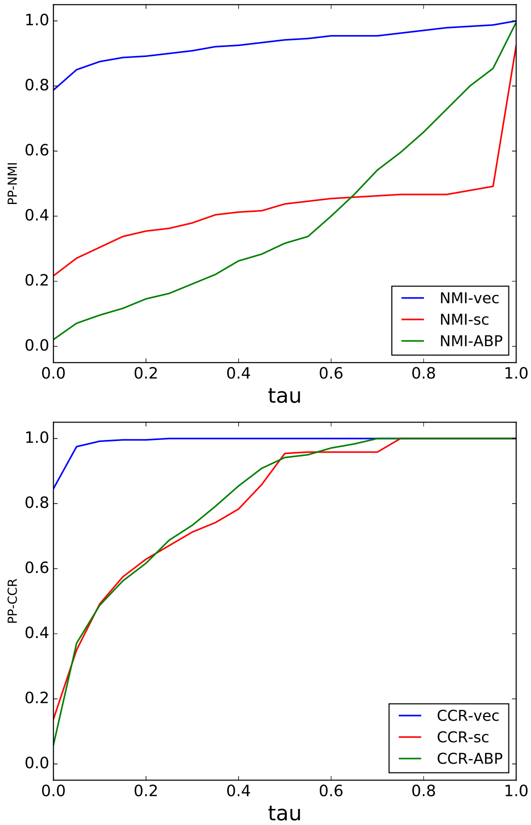


Fig. 10: NMI (top sub-figure) and CCR (bottom sub-figure) Performance Profiles for vec, SC, and ABP on SBM graphs with constant degree scaling. Here, $c \in \{2, 5, 10, 15\}$, $K \in \{2, 5, 10\}$, $N \in \{1e2, 1e3, 1e4, 1e5\}$, and 5 random runs for each combination of settings for a total of 240 distinct experiments.

## V. Experiments with Real-World Graphs

Having comprehensively studied the empirical performance of vec, ABP, and SC on SBM-based synthetic graphs, in this section we turn our attention to real-world datasets that have ground truth (non-overlapping) community labels. Here we use only NMI [9] to measure the performance. We also calculate the **Modularity** which is an oft-used measure of community quality which does not require the knowledge of ground-truth labels [26]. Modularity measures the quality of division of a network into communities (also called modules). Networks with high modularity have dense connections between the nodes within communities but sparse connections between nodes in different communities. Modularity can take values ranging from $-0.5$ to $1$ with higher values indicating better separation of communities.

We consider two benchmark real-world graphs: the *Political Blogs* network [27] and the *Amazon* co-purchasing network [28]. Since the original graphs are directed, we convert it to undirected graphs by forming an edge between two nodes if *either* direction is part of the original graph. The basic statistics of the datasets are summarized in Table III. Here, $\hat{\lambda}$ and $\hat{c'}$ are the maximum likelihood estimates of $\lambda$ and $c'$ respectively in the *planted partition model* SBM under logarithmic scaling. Note that in *Amazon*, the ground truth community proportions are highly unbalanced.

TABLE III: 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 |

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 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 IV, vec achieves better accuracy compared to SC and ABP. The performance of SC is noticeably poorer (in terms of both NMI and Modularity) compared to both vec and ABP. Interestingly, the NMI of SC on random graphs that are synthetically generated according to a *planted partition* SBM model that best fits (in a maximum-likelihood sense) the *Political Blogs* graph is surprisingly good: NMI = 1.0 (average NMI across 10 random graphs). This suggests that real-world graphs such as *Political Blogs* have additional characteristics that are not well-captured by a *planted partition* SBM model. This is further confirmed by the plots of empirical degree distributions of nodes in real-world and synthesized graphs in Fig. 11. The plots show that the node degree distributions are quite different in real-world and synthesized graphs even if the SBM model which is used to generate the graphs is fitted in a maximum-likelihood sense to real-world graphs. These results also suggest that the performance of SC is sensitive to model-mismatch and its good performance on synthetically generated graphs based on SBMs may not be indicative of good performance on matching real-world graphs. The good news is that both vec and ABP do not seem to suffer from this limitation.

TABLE IV: 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 |

Finally, we also visualize the learned embeddings in *Political Blogs* using the now-popular t-Distributed Stochastic Neighbor Embedding (t-SNE) tool [29] in Fig. 12. The picture is consistent with the intuition that nodes from the same community are close to each other in the latent embedding space.

## VI. Concluding Remarks

In this work we put forth a novel framework for community discovery in graphs based on node embeddings. We did this by first constructing, via random walks in the graph, a document made up of sentences of node-paths and then applying a well-known neural word embedding algorithm to it. We then conducted a comprehensive empirical study of community recovery performance on both simulated and real-world graph datasets and demonstrated the effectiveness and robustness of the proposed approach over two state-of-the-art alternatives. In

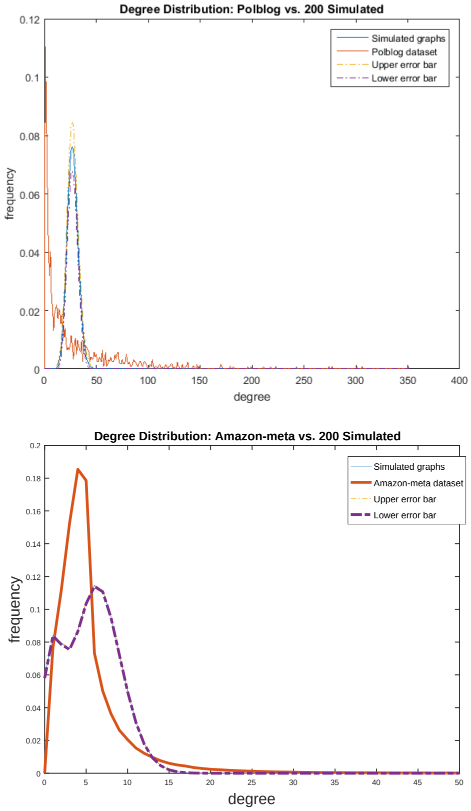


Fig. 11: Empirical degree distributions of real and synthesized graphs for the *Political Blogs* (top) and *Amazon* (bottom) datasets.

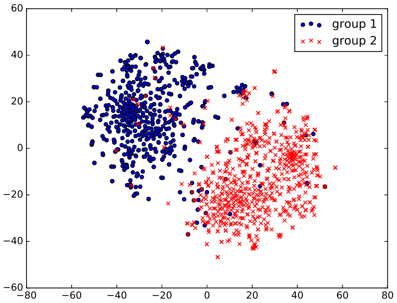


Fig. 12: t-SNE visualization of learned embedding vectors in the *Blogs* dataset. The markers reflect ground-truth groups.

particular, the new method is able to attain the information-theoretic limits for recovery in stochastic block models.

There are a number of aspects of the community recovery problem that we have not explored in this work, but which merit further investigation. First, we have focused on undirected graphs, but our algorithm can be applied 'as-is' to directed graphs as well. We have assumed knowledge of the number of communities $K$, but the node embedding part of the algorithm itself does not make use of this information. In principle, we can apply any $K$-agnostic clustering algorithm to the node embeddings. We have focused on non-overlapping community detection. It is certainly possible to convert an overlapping community detection problem with $K$ communities into a non-overlapping community detection problem with $2^K$ communities, but this approach is unlikely to work well in practice if $K$ is large. An alternative approach is to combine the node embeddings with topic models to produce a "soft" clustering. Finally, this study was purely empirical in nature. Establishing theoretical performance guarantees that can explain the excellent performance of our algorithm is an important task which seems challenging at this time.

## References

[1] Y. Bengio, A. Courville, and P. 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] M. Belkin and P. 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] Z. Yang, W. Cohen, and R. 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 20th 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, p. 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, p. 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] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation.” in *EMNLP*, vol. 14, 2014, pp. 1532–43.

[25] E. Dolan and J. Moré, “Benchmarking optimization software with performance profiles,” *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, 2002.

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

[27] 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.

[28] J. Leskovec and A. Krevl, “SNAP Datasets: Stanford large network dataset collection,” http://snap.stanford.edu/data, Jun. 2014.

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