CommunityGAN: Community Detection with Generative Adversarial Nets

Yuting Jia, Qinqin Zhang, Weinan Zhang, Xinbing Wang
Shanghai Jiao Tong University
{hnxxjyt,zqq.lh.377,wnzhang,xwang8}@sjtu.edu.cn

ABSTRACT

Community detection refers to the task of discovering groups of vertices sharing similar properties or functions so as to understand the network data. With the recent development of deep learning, graph representation learning techniques are also utilized for community detection. However, the communities can only be inferred by applying clustering algorithms based on learned vertex embeddings. These general cluster algorithms like K-means and Gaussian Mixture Model cannot output much overlapped communities, which have been proved to be very common in many real-world networks. In this paper, we propose CommunityGAN, a novel community detection framework that jointly solves overlapping community detection and graph representation learning. First, unlike the embedding of conventional graph representation learning algorithms where the vector entry values have no specific meanings, the embedding of CommunityGAN indicates the membership strength of vertices to communities. Second, a specifically designed Generative Adversarial Net (GAN) is adopted to optimize such embedding. Through the minimax competition between the motif-level generator and discriminator, both of them can alternatively and iteratively boost their performance and finally output a better community structure. Extensive experiments on synthetic data and real-world tasks demonstrate that CommunityGAN achieves substantial community detection performance gains over the state-of-the-art methods.

CCS CONCEPTS

• **Information systems** → **Clustering**; **Social networks**.

KEYWORDS

Community Detection; Graph Representation Learning; Generative Adversarial Nets

1 INTRODUCTION

Network is a powerful language to represent relational information among data objects from social, natural and academic domains [32].

One way to understand network is to identify and analyze groups of vertices which share highly similar properties or functions. Such groups of vertices can be users from the same organization in social networks [16], proteins with similar functionality in biochemical networks [12], and papers from the same scientific fields in citation networks [20]. The research task of discovering such groups is known as the community detection problem [36].

Graph representation learning, also known as network embedding, which aims to represent each vertex in a graph as a low-dimensional vector, has been consistently studied in recent years. The application of deep learning algorithms including Skip-gram [8, 19] and convolutional network [10] has improved the efficiency and performance of graph representation learning dramatically. Moreover, Generative Adversarial Nets (GAN) has also been introduced for learning better graph representation [25]. The learned vertex representation vectors can benefit a wide range of network analysis tasks including link prediction [6, 26], recommendation [34, 35], and node classification [23].

However, there still exists many limitations for the application of such embedding in overlapping community detection problems because of the dense overlapping of communities. Generally, the useful information of vertex embedding vectors is the relevant distance of these vectors, while the specific value in vertex embedding vectors has no meanings. Thus, given the representation vectors of the vertices, one has to adopt other algorithms like logistic regression to accomplish the real-world application tasks. To detect communities, a straightforward way is to run some clustering algorithms in the vector space. However, in some real-world datasets, one vertex may belong to tens of communities simultaneously [29, 30], while most clustering algorithms cannot handle such dense overlapping. In recent years, some researchers try to perform the community detection and network embedding simultaneously in a unified framework [3, 27] but still fail to solve the dense overlapping problem.

In this paper, we propose *CommunityGAN*, a novel community detection framework that jointly solves overlapping community detection and graph representation learning. The input-output overview of CommunityGAN is shown in Figure 1. With CommunityGAN, we aim to learn network embeddings like AGM (Affiliation Graph Model) through a specifically designed GAN. AGM [30, 31] is a framework which can model densely overlapping community structures. It assigns each vertex-community pair a nonnegative factor which represents the degree of membership of the vertex to the community. Thus, the strengths of membership from a vertex to all communities compose the representation vector of it.

Moreover, in recent years, motifs have been proved as essential features in community detection tasks [11]. Thus in this paper, unlike most previous works considering relationship between only
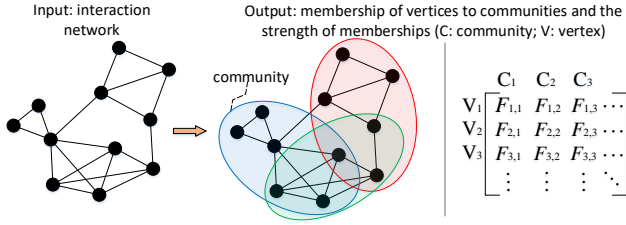
Figure 1: The input-output overview of CommunityGAN.

two vertices (the relationship between a center vertex and one of the other vertices in a window), we try to generate and discriminate motifs. Specifically, CommunityGAN trains two models during the learning process: 1) a generator $G(s|v_c)$, which tries to generate the most likely vertex subset s to compose a specified kind of motif; 2) a discriminator $D(s)$, which attempts to distinguish whether the vertex subset s is a real motif or not. In the proposed Community-GAN, the generator G and the discriminator D act as two players in a minimax game. Competition in this game drives both of them to improve their capability until the generator is indistinguishable from the true connectivity distribution.

The contributions of our work are threefold:

- We combine AGM and GAN in a unified framework, which achieves both the outstanding performance of GAN and the direct vertex-community membership representation in AGM.
- We study the motif distribution among ground-truth communities and analyze how they can help improve the quality of detected communities.
- We propose a novel implementation for motif generation called *Graph AGM*, which can generate the most likely motifs with graph structure awareness in a computationally efficient way.

Empirically, two experiments were conducted on a series of synthetic data, and the results prove: 1) the ability of CommunityGAN to solve dense overlapping problem; 2) the efficacy of motif-level generation and discrimination. Additionally, to complement these experiments, we evaluate CommunityGAN on two real-world scenarios, i.e., community detection and clique prediction, using five real-world datasets. The experiment results show that CommunityGAN substantially outperforms the state-of-the-art methods in the field of community detection and graph representation learning. Specifically, CommunityGAN outperforms baselines by 7.9% to 21.0% in terms of F1-Score for community detection. Additionally, in 3-clique and 4-clique prediction tasks, Community-GAN improves AUC score to at least 0.990 and 0.956 respectively. The superiority of CommunityGAN relies on its joint advantage of particular embedding design, adversarial learning and motif-level optimization.

2 RELATED WORK

Community Detection. Many community detection algorithms have been proposed from different perspectives. One direction is to design some measure of the quality of a community like modularity,

and community structure can be uncovered by optimizing such measures [17, 28]. Another direction is to adopt the generative models to describe the generation of the graphs, and the communities can be inferred by fitting graphs to such models [9, 36]. Moreover, some models focus on the graph adjacency matrix and output the relationship between vertices and communities by adopting matrix factorization algorithms on the graph adjacency matrix [14, 31]. These models often consider the dense community overlapping problem and detect overlapping communities. However, the performance of these methods are restricted by performing pair reconstruction with bi-linear models.

Graph representation learning. In recent years, several graph representation learning algorithms have been proposed. For example, DeepWalk [19] shows the random walk in a graph is similar to the text sequence in natural language. Thus it adopts Skip-gram, a word representation learning model [15], to learn vertex representation. Node2vec [8] further extends the idea of DeepWalk by proposing a biased random walk algorithm, which provides more flexibility when generating the sampled vertex sequence. LINE [24] first learns the vertex representation preserving both the first and second order proximities. Thereafter, GraRep [2] applies different loss functions defined on graphs to capture different k-order proximities and the global structural properties of the graph. Moreover, GAN has also been introduced into graph representation learning. GraphGAN [25] proposes a unified adversarial learning framework, which naturally captures structural information from graphs to learn the graph representation. ANE [5] utilizes GAN as a regularizer for learning stable and robust feature extractor. However, all of the above algorithms focus on general graph representation learning. For community detection tasks, we have to adopt other clustering algorithms on vertex embeddings, which cannot handle the dense community overlapping problem. Compared to above mentioned methods, our CommunityGAN can output the membership of vertices to communities directly.

Unified framework for graph representation learning and community detection. In recent years, some unified frameworks for both graph representation learning and community detection have been proposed. Wang et al. [27] developed a modularized nonnegative matrix factorization algorithm to preserve both vertex similarity and community structure. However, because of the complexity of matrix factorization, this model cannot be applied on many real-world graphs containing millions of vertices. Cavallari et al. [3] designed a framework that jointly solves community embedding, community detection and node embedding together. However, although the community structures and vertex embeddings are optimized together, the community structures are still inferred through a clustering algorithm (e.g., Gaussian Mixture Model) based on the vertex embedding, which means the dense community overlapping problem cannot be avoided.

3 EMPIRICAL OBSERVATION

In this section, data analysis of the relationship between motifs and communities is presented, which motivates the development of CommunityGAN. We first describe the network datasets with

Table 1: Dataset statistics. V: number of vertices, E: number of edges, T: number of 3-clique (triangle), C: number of communities, A: community memberships per vertex.

Dataset	V	E	T	C	A
LiveJournal	4.0M	34.9M	178M	310K	3.09
Orkut	3.1M	120M	528M	8.5M	95.93
Youtube	1.1M	3.0M	3.1M	30K	0.26
DBLP	0.43M	1.3M	2.2M	2.5K	2.57
Amazon	0.34M	0.93M	0.67M	49K	14.83

Table 2: The occurrence probability of cliques for vertices sampled from all vertices or from one community. R: from all vertices. C: from one community.

Dataset	2-Clique		3-Clique		4-Clique	
	R	C	R	C	R	C
LiveJournal	4E-6	0.80	2E-11	0.18	0	0.08
Orkut	2E-5	0.82	8E-11	0.18	0	0.06
Youtube	4E-6	0.73	0	0.10	0	0.02
DBLP	1E-5	0.52	1E-10	0.31	0	0.23
Amazon	6E-6	0.53	0	0.18	0	0.05

Figure 2: Normalized probability for vertices to be a clique as a function of number of shared communities. Probabilities are scaled so that maximum value is one.

ground-truth communities and then present our empirical observations.
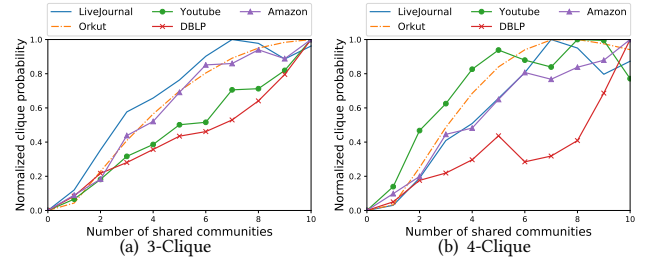
3.1 Datasets

We study the five networks proposed by Yang and Leskovec [30], where LiveJournal, Orkut and Youtube are all online social networks, while DBLP and Amazon are collaboration network and product network respectively. Table 1 provides the statistics of these networks.

3.2 Empirical Observations

Now we present our empirical observations by answering two critical questions. How do the communities contribute to the generation of motifs? What is the change in motif generation with communities overlapping?

First, we investigate the relationship between motif generation and community structure. For each time, we randomly select one community. Then we sample 2/3/4 vertices from this community and judge whether they could compose a motif or not. We repeat this process for billions of times and get the occurrence probability of motifs in one community. We compare this value with the motif occurrence probability in the whole network. Because, in this paper, we mostly focus on a particular kind of motifs (clique), we only demonstrate the occurrence probability of 2/3/4-vertex cliques. As shown in Table 2, the average occurrence probabilities of cliques for vertices in one community are much higher than that for vertices randomly selected from the whole network. This observation demonstrates that the occurrence of motifs is strongly correlated to the community structure.

Next, we study the influence of community overlapping on the generation of motifs. Yang and Leskovec [31] have conducted the

relationship of 2-clique (edge) probability with community overlapping that the more communities two vertices have in common, the higher probability for them to be 2-clique. In this paper, we further study the influence of community overlapping on the generation of 3-clique and 4-clique. As shown in Figure 2, similar to 2-clique, the probability curve increases in the overall trend as the number of shared communities increases. Such observation accords with the base assumption of AGM framework that vertices residing in communities' overlaps are more densely connected to each other than the vertices in a single community. So we can extend the AGM from edge generation to clique generation, and the details will be explained in §4.4.

4 METHODOLOGY

Notation. In this paper, all vectors are column vectors and denoted by lower-case bold letters like $\mathbf{d}_v$ and $\mathbf{g}_v$. Calligraphic letters are used to represent sets like $\mathcal{V}$ and $\mathcal{E}$. And for simplicity, the corresponding normal letters are used to denote their size like $V = |\mathcal{V}|$. The details of notations in this paper are listed in Table 3.

In this section, we introduce the framework of Community-GAN and discuss the implementation and optimization of the generator and discriminator in CommunityGAN.

4.1 CommunityGAN Framework

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{M})$ represent the studied graph, where $\mathcal{V} = \{v_1, \ldots, v_V\}$ is a set of vertices, $\mathcal{E} = \{e_{ij}\}_{i,j=1}^{V}$ is the set of edges among the vertices, and $\mathcal{M}$ is the set of motifs in graph $\mathcal{G}$. In this paper, notably, we only focus on a particular kind of motifs: cliques. Moreover, the D and G are also designed to discriminate and generate cliques. For a given vertex v_c, we define $M(v_c)$ as the set of motifs in the graph covering v_c, whose size is generally much smaller than the total number of vertices V. Moreover, we define conditional probability $p_{true}(m|v_c)$ as the preference distribution of motifs covering v_c over all other motifs in $\mathcal{M}$. Thus $M(v_c)$ can be seen as a set of observed motifs drawn from $p_{true}(m|v_c)$. Given the graph $\mathcal{G}$, we aim to learn the following two models: **Generator** $G(s|v_c; \theta_G)$, which tries to approximate $p_{true}(m|v_c)$, and generate (or select) subsets of vertices from $\mathcal{V}$ covering v_c most likely to be real motifs; and **Discriminator** $D(s; \theta_D)$, which aims to estimate the probability that a vertex subset s is a real motif, i.e., comes from $\mathcal{M}$.

Table 3: Notations used in this paper

| Symbol	Description
$\mathcal{G}$	The studied graph
$\mathcal{V}, \mathcal{E}, \mathcal{M}, \mathcal{C}$	Set of vertices, edges, motifs, communities in graph $\mathcal{G}$
V, E, M, C	Number of vertices, edges, motifs, communities in graph $\mathcal{G}$
$M(v_c)$	Set of motifs covering v_c in graph $\mathcal{G}$
$N(v_c)$	Set of neighbors of vertex v_c in graph $\mathcal{G}$
s	The subset of vertices from $\mathcal{V}$
$p_{true}(m\|v_c)$	The preference distribution of motifs covering v_c over all other motifs in $\mathcal{M}$
$G(s\|v_c; \theta_G)$	Generator trying to generate vertex subsets from $\mathcal{V}$ covering v_c most likely to be real motifs
$D(s, \theta_D)$	Discriminator aiming to estimate the probability that a vertex subset s is a real motif
$\mathbf{d}_v, \mathbf{g}_v \in \mathbb{R}_+^C$	Nonnegative C-dimensional representation vector of vertex v in discriminator D and generator G, respectively
$\theta_D, \theta_G \in \mathbb{R}_+^{V \times C}$	The union of all $\mathbf{d}_v$ and $\mathbf{g}_v$, respectively
$F_{uc} \in \mathbb{R}_+$	The nonnegative strength for vertex u to be affiliated to community c used in AGM
$p_c(v_1, v_2, \ldots, v_m)$	The probability of m vertices v_1 to v_m to be a clique through of community c
$p(v_1, v_2, \ldots, v_m)$	The probability of m vertices v_1 to v_m to be a clique through of any one community
$\odot(F_{v_1}, F_{v_2}, \ldots, F_{v_m})$	Sum of the entrywise product from F_{v_1} to F_{v_m}, i.e., $\sum_{c=1}^{C} \prod_{i=1}^{m} F_{v_i c}$
$G_v(v_{s_m}\|v_{s_1}, \ldots, v_{s_{m-1}})$	The vertex generator based on selected vertices $v_{s_1}, \ldots, v_{s_{m-1}}$ in vertex subset generation process
$p_{v_v}(v_i\|v_c)$	The relevance probability of v_i given v_c with root v_v in random walk process

The generator G and discriminator D are combined by playing a minimax game: generator G would try to perfectly approximate $p_{true}(m\|v_c)$ and generate the most likely vertex subsets similar to real motifs covering v_c to fool the discriminator, while discriminator D would try to distinguish between ground-truth motifs from $p_{true}(m\|v_c)$ and the ones generated by G. Formally, G and D act as two opponents in the following two-player minimax game with the joint value function $V(G, D)$:

$$\min_{\theta_G} \max_{\theta_D} V(G, D) = \sum_{c=1}^{V} \Big(\mathbb{E}_{m \sim p_{true}(\cdot|v_c)}[\log D(m; \theta_D)] $$
$$+ \mathbb{E}_{s \sim G(s|v_c; \theta_G)}[\log(1 - D(s; \theta_D))] \Big). \tag{1}$$

Based on Eq. (1), the optimal G and D can be learned by alternatively maximizing (D) and minimizing (G) the value function $V(G, D)$. The CommunityGAN framework is illustrated in Figure 3. Discriminator D is trained with positive samples from $p_{true}(\cdot|v_c)$ and negative samples from $G(\cdot|v_c; \theta_G)$, and generator G is updated with policy gradient technique [22] under the guidance of D (detailedly described later in this section). Competition between G and D drives both of them to improve until G is indistinguishable from the true motif distribution p_{true}.

4.2 CommunityGAN Optimization

Given positive samples from true motif distribution p_{true} and negative samples from the generator G, the discriminator aims to maximize the log-probability of correctly classifying these samples, which could be solved by gradient ascent if D is differentiable for
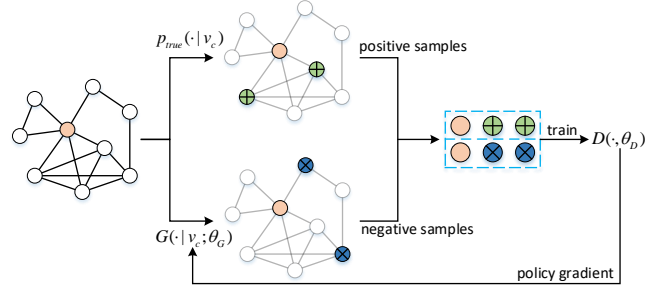
Figure 3: Framework of CommunityGAN

θ_D, i.e.,

$$\nabla_{\theta_D} V(G, D) = \sum_{c=1}^{V} \Big(\mathbb{E}_{m \sim p_{true}(\cdot|v_c)}[\nabla_{\theta_D} \log D(m; \theta_D)]$$
$$+ \mathbb{E}_{s \sim G(s|v_c; \theta_G)}[\nabla_{\theta_D} \log(1 - D(s; \theta_D))] \Big). \tag{2}$$

In contrast to the discriminator, the objective of the generator is to minimize the log-probability that the discriminator correctly distinguishes samples from G. Because the sampling of s is discrete, we follow [21, 25, 33] to compute the gradient of $V(G, D)$ with respect to θ_G by policy gradient:

$$\nabla_{\theta_G} V(G, D) = \nabla_{\theta_G} \sum_{c=1}^{V} \mathbb{E}_{s \sim G(\cdot|v_c)}[\log(1 - D(s))]$$
$$= \sum_{c=1}^{V} \mathbb{E}_{s \sim G(\cdot|v_c)}[\nabla_{\theta_G} \log G(s|v_c) \log(1 - D(s))]. \tag{3}$$

4.3 A Naive Implementation of D and G

A naive implementation of the discriminator and generator is based on sigmoid and softmax functions, respectively.

For the discriminator D, intuitively we can define it as the multiplication of the sigmoid function of the inner product of every two vertices in the input vertex subset s:

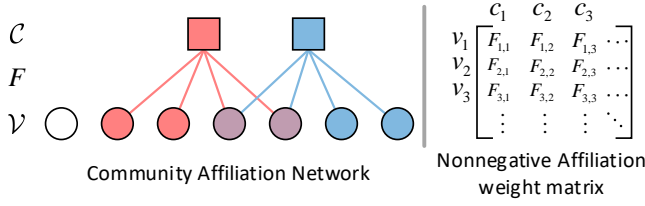$$D(s) = \prod_{(u, v) \in s, u \neq v} \sigma(\mathbf{d}_u^\top \cdot \mathbf{d}_v), \tag{4}$$

where $\mathbf{d}_u, \mathbf{d}_v \in \mathbb{R}^k$ are the k-dimensional representation vectors for discriminator D of vertices u and v respectively, and θ_D is the union of all $\mathbf{d}_v$'s.

For the implementation of G, to generate a vertex subset s covering vertex v_c, we can regard the subset as a sequence of vertices $(v_{s_1}, v_{s_2}, \ldots, v_{s_m})$ where $v_{s_1} = v_c$. Then the generator G can be defined as follows:

$$G(s|v_c)$$
$$= G_v(v_{s_2}|v_{s_1})G_v(v_{s_3}|v_{s_1}, v_{s_2}) \cdots G_v(v_{s_m}|v_{s_1}, \ldots, v_{s_{m-1}}). \tag{5}$$

Notably, in Eq. (5), the generation of v_{s_m} is based on vertices from v_{s_1} to $v_{s_{m-1}}$, not only on $v_{s_{m-1}}$. Because if we generate v_{s_m} based only on $v_{s_{m-1}}$, it will be very likely that vertex v_{s_m} and the other vertices belong to different communities. For example, vertices from v_{s_1} to $v_{s_{m-1}}$ are students in one university, while vertex v_{s_m} is the parent of $v_{s_{m-1}}$. Simply, we can know the probability of the vertex subset s being a clique will be very low. Thus, we generate v_{s_m} based on all the vertices from v_{s_1} to $v_{s_{m-1}}$.

Figure 4: AGM framework. $\mathcal{V}$: vertices; C: communities; F: affiliation of vertices to communities.

For the implementation of the vertex generator G_v, straightforwardly, we can define it as a softmax function over all other vertices, i.e.,

$$G_v(v_{s_m}|v_{s_1},\ldots,v_{s_{m-1}}) = \frac{\exp(\sum_{i=1}^{m-1}\mathbf{g}_{v_{s_m}}^\top\mathbf{g}_{v_{s_i}})}{\sum_{v\notin(v_{s_1},\ldots,v_{s_{m-1}})}\exp(\sum_{i=1}^{m-1}\mathbf{g}_v^\top\mathbf{g}_{v_{s_i}})}, \tag{6}$$

where $\mathbf{g}_v \in \mathbb{R}^k$ is the k-dimensional representation vectors for generator G_v of vertex v, and θ_G is the union of all $\mathbf{g}_v$'s.

4.4 Graph AGM

Sigmoid and softmax function provide concise and intuitive definitions for the motif discrimination in D_s and vertex generation in G_v, but they have three limitations in community detection task: 1) To detect community, after learning the vertex representation vectors based on Eq. (4) and (6), we still need to adopt some clustering algorithms to detect the communities. According to [29], the overlap is indeed a significant feature of many real-world social networks. Moreover, in [30], the authors showed that in some real-world datasets, one vertex might belong to tens of communities simultaneously. However, general clustering algorithms cannot handle such dense overlapping. 2) The calculation of softmax in Eq. (6) involves all vertices in the graph, which is computationally inefficient. 3) The graph structure encodes rich information of proximity among vertices, but the softmax in Eq. (6) completely ignores it.

To address these problems, in CommunityGAN we propose a novel implementation for the discriminator and generator, which is called Graph AGM.

AGM (Affiliation Graph Model) [30, 31] is based on the idea that communities arise due to shared group affiliation, and views the whole network as a result generated by a community-affiliation graph model. The framework of AGM is illustrated in Figure 4, which can be either seen as a bipartite network between vertices and communities, or written as a nonnegative affiliation weight matrix. In AGM, each vertex could be affiliated to zero, one or more communities. If vertex u is affiliated to community c, there will be a nonnegative strength F_{uc} of this affiliation. For any community $c \in C$, it connects its member vertices u, v with probability $1 - \exp(-F_{uc} \cdot F_{vc})$. Moreover, each community c creates edges independently. If the pair of vertices u, v are connected multiple times through different communities, the probability is $1 - \exp(-\sum_c F_{uc} \cdot F_{vc})$. So that the probability that vertices u, v are connected (through any possible communities) is $p(u, v) = 1 - \exp(-F_u^\top \cdot F_v)$, where F_u and F_v are the nonnegative C-dimensional affiliation vectors for vertices u and v respectively.

We extend AGM from edge generation to motif generation. For any m vertices v_1 to v_m, we assume that the probability of them to be a clique through of community c is defined as $p_c(v_1, v_2, \ldots, v_m) = 1 - \exp(-\prod_{i=1}^m F_{v_ic})$. Then the probability that these m vertices compose a clique through any possible communities can be calculated via

$$p(v_1, v_2, \ldots, v_m) = 1 - \prod_c (1 - p_c(v_1, v_2, \ldots, v_m))$$
$$= 1 - \exp(- \odot (F_{v_1}, F_{v_2}, \ldots, F_{v_m})), \tag{7}$$

where $\odot(F_{v_1}, F_{v_2}, \ldots, F_{v_m})$ means the sum of the entrywise product from F_{v_1} to F_{v_m}, i.e., $\sum_{c=1}^C \prod_{i=1}^m F_{v_ic}$.

Then the discriminator, which was defined as the product of sigmoid in a straightforward way, can be redefined as

$$D(s) = 1 - \exp(- \odot (\mathbf{d}_{v_1}, \mathbf{d}_{v_2}, \ldots, \mathbf{d}_{v_m})), \tag{8}$$

where $\mathbf{d}_v \in \mathbb{R}^C$ is the nonnegative C-dimensional representation vectors of vertex v for discriminator D, and θ_D is the union of all $\mathbf{d}_v$'s.

Moreover, the generator G_v can be redefined as the softmax function over all other possible vertices to compose a clique with $m - 1$ chosen vertices:

$$G_v(v_{s_m}|v_{s_1},\ldots,v_{s_{m-1}})$$
$$= \frac{1 - \exp(- \odot (\mathbf{g}_{v_{s_1}},\ldots,\mathbf{g}_{v_{s_m}}))}{\sum_{v\notin(v_{s_1},\ldots,v_{s_{m-1}})} 1 - \exp(- \odot (\mathbf{g}_{v_{s_1}},\ldots,\mathbf{g}_{v_{s_{m-1}}},\mathbf{g}_v))}, \tag{9}$$

where $\mathbf{g}_v \in \mathbb{R}^C$ is the nonnegative C-dimensional representation vectors of vertex v for generator G_v, and θ_G is the union of all $\mathbf{g}_v$'s. With this setting, the learned vertex representation vectors $\mathbf{g}_v$ will represent the affiliation weight between vertex v and communities, which means we need no additional clustering algorithms to find the communities and the first aforementioned limitation is omitted.

To further overcome the other two limitations, inspired by [25], we design graph AGM as follows. To calculate $G_v(v_{s_m}|v_{s_1},\ldots,v_{s_{m-1}})$, we first assume there is a virtual vertex v_v which is connected to all the vertices in the union of neighbors of vertices from v_{s_1} to $v_{s_{m-1}}$, i.e., $N(v_v) = N(v_{s_1}) \cup \cdots \cup N(v_{s_{m-1}})$ where $N(v)$ represents the set of neighbors of vertex v. Moreover, we assign $\mathbf{g}_{v_v} = \mathbf{g}_{v_{s_1}} \circ \cdots \circ \mathbf{g}_{v_{s_{m-1}}}$, i.e., the representation vector of v_v is the entrywise product of the representation vectors of vertices from v_{s_1} to $v_{s_{m-1}}$. For simplicity, we substitute $G_v(v_{s_m}|v_{s_1},\ldots,v_{s_{m-1}})$ with $G_v(v_{s_m}|v_v)$. Then we adopt random walk starting from the virtual vertex v_v based on a well designed probability distribution (Eq. (10)). During the process of random walk, if the currently visited vertex is v and generator G decides to visit v's previous vertex, then v is chosen as the generated vertex and the random walk process stops. The whole process for generating a vertex subset with the size of 3 is illustrated in Figure 5.

Moreover, in the random walk process, we wish the walk path is always relevant to the root vertex v_v for maximizing the probability of the generated vertex subset to be a motif. So that for a given vertex v_c and one of its neighbors $v_i \in N(v_c)$, we define the relevance probability of v_i given v_c as

$$p_{v_v}(v_i|v_c) = \frac{1 - \exp(- \odot (\mathbf{g}_{v_i}, \mathbf{g}_{v_c}, \mathbf{g}_{v_v}))}{\sum_{v_j\in N(v_c)} 1 - \exp(- \odot (\mathbf{g}_{v_j}, \mathbf{g}_{v_c}, \mathbf{g}_{v_v}))}, \tag{10}$$

which is actually a softmax function over $N(v_c)$ for composing clique with vertices v_v and v_c.
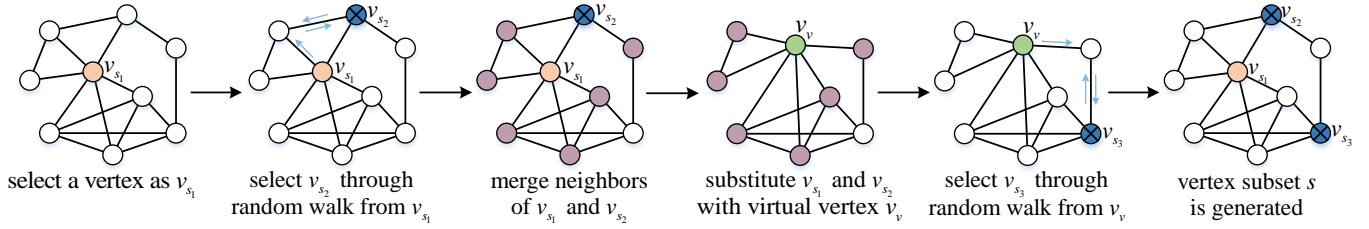
Figure 5: The generation process for vertex subset s. Blue arrows indicate the path of random walk. At the two blue vertices, because the vertex generator decides to visit the previous vertex, the random walk finishes and the blue vertices are selected.

Algorithm 1: CommunityGAN framework

Input: number of communities c, size of discriminating samples m, size of generating samples n.

Output: generator $G(s|v_c; \theta_G)$, discriminator $D(s, \theta_D)$.

1 Initialize and pre-train $G(s|v_c; \theta_G)$ and $D(s, \theta_D)$;
2 **while** *CommunityGAN not converge* **do**
3   **for** *G-steps* **do**
4   Generate n vertex subsets from $G(s|v_c; \theta_G)$ for each vertex v_c based on the random walk process;
5   Update θ_G according to Eq. (3), (5), (10) and (11);
6   **end**
7   **for** *D-steps* **do**
8   Sample m positive vertex subsets from p_{true} and s negative vertex subsets from $G(s|v_c; \theta_G)$ for each vertex v_c;
9   Update θ_D according to Eq. (2) and (8);
10   **end**
11 **end**
12 **return** $G(s|v_c; \theta_G)$ and $D(s, \theta_D)$;

If we denote the path of random walk as $P_r = (v_{r_1}, v_{r_2}, \ldots, v_{r_n})$ where $v_{r_1} = v_v$, the probability for selecting this path will be $p_{v_v}(v_{r_{n-1}}|v_{r_n}) \cdot \prod_{i=1}^{n-1} p_{v_v}(v_{r_{i+1}}|v_{r_i})$. In the policy gradient, we regard the selection of this path as an action and the target is to maximize its reward from D. Thus, although there may be multiple paths between v_{r_1} and v_{r_n}, if we have selected the path P_r, we will optimize the policy gradient on it and neglect other paths. In other words, if we select the path P_r, we assign $G_v(v_{s_m}|v_v)$ as follows:

$$G_v(v_{s_m}|v_v) = p_{v_v}(v_{r_{n-1}}|v_{r_n}) \cdot \prod_{i=1}^{n-1} p_{v_v}(v_{r_{i+1}}|v_{r_i}). \quad (11)$$

Finally, the overall solution of CommunityGAN is summarized in Algorithm 1.

4.5 Other Issues

Model Initialization. We have two methods to initialize the generator G and the discriminator D. The first is that we can deploy AGM model on the graph to learn a community affiliation vector F_i for each vertex v_i, and then we can set $g_{v_i} = d_{v_i} = F_i$ directly. The second is that we can use locally minimal neighborhoods [7] to initialize θ_G and θ_D. We can regard each vertex v_i along with its neighbors $N(v_i)$, denoted as $C(v_i)$, as a community. Community $C(v_i)$ is called locally minimal if $C(v_i)$ has lower conductance than all the $C(v_j)$ for vertices v_j who are connected to vertex v_i.

Gleich and Seshadhri [7] have empirically showed that the locally minimal neighborhoods are good seed sets for community detection algorithms. For a node v_i who belongs to a locally minimal neighborhood c, we initialize $F_{v_i}c = 1$, otherwise $F_{v_i}c = 0$.

The second method can save the training time of AGM on the graph. However, we find that the performance of such initialization is a little lower than the first initialization. To achieve best performance, in this paper, we choose to adopt the first initialization method. Moreover, in the efficiency analysis experiment (refer to §6.4), the training time of CommunityGAN includes the time of the pre-training process.

Determining community membership. After learning parameters θ_G and θ_D, we need to determine the "hard" community membership of each node. We achieve this by thresholding θ_G and θ_D with a threshold δ. The basic intuition is that if two nodes belong to the same community c, then the probability of having a link between them through community c should be larger than the background edge probability $\epsilon = 2E/V(V-1)$. Following this idea, we can obtain the threshold δ as below:

$$1 - \exp(-\delta^2) = \epsilon \Rightarrow \delta = \sqrt{-\log(1-\epsilon)} \quad (12)$$

With δ obtained, we consider vertex v_i belonging to community c if $\mathbf{g}_{v_i c} \geq \delta$ or $\mathbf{d}_{v_i c} \geq \delta$ for generator G and discriminator D respectively.

Choosing the number of communities. We follow the method proposed in [1] to choose the number of communities C. Specifically, we reserve 20% of links for validation and learn the model parameters with the remaining 80% of links for different C. After that, we use the learned parameters to predict the links in validation set and select the C with the maximum prediction score as the number of communities.

5 SYNTHETIC DATA EXPERIMENTS

In this section, in order to evaluate the effectiveness of CommunityGAN, we conduct a series of experiments based on synthetic datasets. Specifically, we conduct two experiments to prove: 1) the ability of CommunityGAN to solve the dense overlapping problem; 2) the efficacy of motif-level generation and discrimination. To simulate the real-world graphs with ground-truth overlapping communities, we adopt CKB Graph Generator - a method that can generate large random social graphs with realistic overlapping community structure [4] - to generate such synthetic graphs. The

Table 4: Synthetic Graphs statistics. V: number of vertices, E: number of edges, C: number of communities, A: average number of community memberships per vertex, P: percentage of communities that have overlapping with others

Graph	V	E	C	A	P
ckb-290	1078	29491	226	2.76	100%
ckb-280	1044	30055	252	3.03	100%
ckb-276	1023	38775	339	3.74	100%
ckb-273	1018	28025	372	3.94	100%
ckb-270	1007	30740	371	4.20	100%
ckb-267	1123	65813	381	4.57	100%

code of CommunityGAN (including pre-training model) and demo datasets are available online[1].

5.1 Evaluation Metric

The availability of ground-truth communities allows us to quantitatively evaluate the performance of community detection algorithms. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{M})$, we denote the set of ground-truth communities as C^* and the set of detected communities as $\hat{C}$, where each ground-truth community $c_i^* \in C^*$ and each detected community $\hat{c}_i \in \hat{C}$ are both defined by a set of their member nodes. We consider the average F1-Score as the metric to measure the performance of the detected communities $\hat{C}$. To compute the F1-Score between detected communities and ground-truth communities, we need to determine which $c_i^* \in C^*$ corresponds to which $\hat{c}_j \in \hat{C}$. We define final F1-Score as the average of the F1-score of the best-matching ground-truth community to each detected community, and the F1-score of the best-matching detected community to each ground-truth community:

$$F1(C^*, \hat{C}) = \frac{1}{2}\left(\frac{1}{C^*}\sum_{c_i^* \in C^*}\max_{\hat{c}_j \in \hat{C}} F1(c_i^*, \hat{c}_j) + \frac{1}{\hat{C}}\sum_{\hat{c}_i \in \hat{C}}\max_{c_j^* \in C^*} F1(c_j^*, \hat{c}_i)\right)$$

(13)

The higher F1-Score means that the detected communities are more accurate and have better qualities.

5.2 Datasets

To prove CommunityGAN's ability to solve the dense overlapping problem, we utilize the CKB Graph Generator to generate synthetic graphs with ground-truth communities of different overlapping levels. All the parameters of CKB are set as default except two: 'power law exponent of user-community membership distribution (β_1)' and 'power law exponent of community size distribution (β_2)' [4]. By setting $\beta_1 = \beta_2 = 2.9, 2.8, 2.76, 2.73, 2.7, 2.67$, we can get a series of graphs with different levels of community overlapping. The detailed statistics of these graphs are shown in Table 4. We can see that the percentages of overlapped communities (denoted by P) are all 100%, indicating that all the communities overlap more or less with others. Besides, with the decrease of β_1 and β_2, the average number of community memberships per vertex (denoted by A) increases, which means the density of community overlapping is becoming higher and higher.
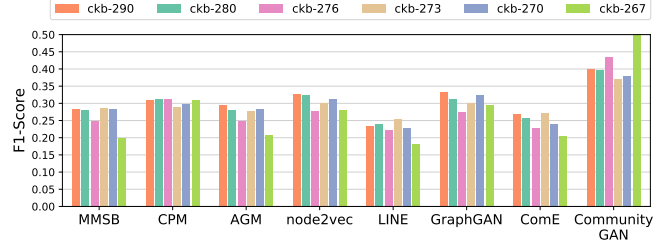
Figure 6: Performance of methods on the synthetic graphs.

5.3 Comparative Methods

We compare CommunityGAN with 7 baseline methods, of which 3 are traditional overlapping community detection methods (MMSB, CPM and AGM), 3 are recent graph representation learning methods (node2vec, LINE and GraphGAN), and 1 combines graph representation learning as well as community detection (ComE).

MMSB [1] is one of the representatives of overlapping community detection methods based on dense subgraph extraction.

CPM [18] builds up the communities from k-cliques and allows overlapping between the communities in a natural way.

AGM [31] is a framework which can model densely overlapping community structures.

Node2vec [8] adopts biased random walk and Skip-Gram to learn vertex embeddings.

LINE [24] preserves the first-order and second-order proximity among vertices in the graph.

GraphGAN [25] unifies generative and discriminative graph representation learning methodologies via adversarial training in a minimax game.

**ComE** [3] jointly solves the graph representation learning and community detection problem.

5.4 Experiment Setup

For all the experiments, we perform stochastic gradient descent to update parameters with learning rate 0.001. In each iteration, m and n are both 5, i.e., 5 positive and 5 negative vertex subsets will be sampled or generated for each vertex (refer to Algorithm 1), and then we update G and D on these vertex subsets for 3 times. The final learned communities and vertex representations are $\mathbf{g}_v$'s. Hyperparameter settings for all baselines are as default.

For CommunityGAN, CPM, AGM, MMSB and ComE, which directly output the communities for vertices, we fit the network into them to detect communities directly. For node2vec, LINE and GraphGAN, which can only output the representation for vertices, we fit the network into them and apply K-means to the learned embeddings to get the communities. Because of the randomness of K-means, we repeat 5 times and report the average results. Moreover, the number of ground-truth communities is set as the input parameters to the community detection models and K-means.

5.5 Ability of CommunityGAN to Solve Dense Overlapping Problem

For each method and each dataset, we calculate the average value of F1-Score. The performance of 8 methods on the series of graphs is
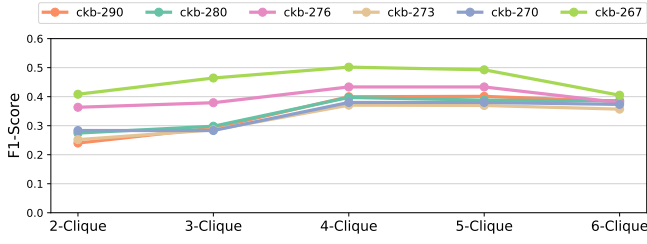
Figure 7: Performance of CommunityGAN with different size of cliques on the synthetic graphs.

Table 5: Statistics of cliques in the synthetic graphs.

Number of cliques					
Model	2-clique	3-clique	4-clique	5-clique	6-clique
ckb-290	29491	105585	82617	31919	14354
ckb-280	30055	115047	96661	33735	9539
ckb-276	38775	175639	189300	126720	76247
ckb-273	28025	122566	166760	163765	165575
ckb-270	30740	140547	203114	211604	214018
ckb-267	65813	377686	580950	811522	1097762

Number of covered vertices					
Model	2-clique	3-clique	4-clique	5-clique	6-clique
ckb-290	1078	1078	1078	1065	595
ckb-280	1044	1044	1044	1034	617
ckb-276	1023	1023	1023	1018	714
ckb-273	1018	1018	1018	1012	798
ckb-270	1007	1007	1007	1005	797
ckb-267	1123	1123	1123	1120	907

shown in Figure 6. As we can see: 1) CommunityGAN significantly outperforms all the baselines on all the synthetic graphs. 2) Although CommunityGAN is based on AGM and utilizes AGM as the pre-train method, with the minimax game competition between the discriminator and generator, CommunityGAN achieves huge improvements compared to AGM. 3) On the whole, the performance of 4 graph representation learning methods (node2vec, LINE, Graph-GAN, ComE) falls with the increase of overlapping density among communities, while 3 traditional overlapping community detection methods (MMSB, CPM, AGM) keep relatively steady. Among these 3 methods, the only exception is that the performance of MMSB and AGM drops largely on ckb-267. One possible reason is that the average degree of ckb-267 increases significantly compared to other graphs, and thus the performance of the two methods are affected. However, although based on AGM, CommunityGAN still performs very well on the graph ckb-267 with the well designed positive/negative sampling and the minimax game between the generator and discriminator.

5.6 Efficacy of Motif-Level Generation and Discrimination

To evaluate the efficacy of the motif-level generation and discrimination, we tune the size of cliques in the training process of CommunityGAN. Figure 7 shows the performance of CommunityGAN with different sizes of cliques on the series of synthetic graphs. We can find that, generally, cliques with a size of 4 or 5 can reach the best performance. It means that, compared to edges and triangles, the larger cliques in graphs correspond more to the community structure. In other words, community detection methods should not only focus on the edges in the graph, but also pay more attention to the motifs, especially larger motifs. However, we can see that in most graphs the performance falls when CommunityGAN generates and discriminates 6-vertex cliques. Our explanation is that too-large cliques are rare in graphs, which can only cover a relatively small number of nodes. Because of these uncovered nodes, few positive samples can be used in the training process, which in turn causes CommunityGAN to fail to learn appropriate representation vectors.

To prove such an explanation, we have counted the number of cliques and number of vertices covered by such cliques, as shown in Table 5. We can see that no matter whether the number of cliques increases or not, the number of covered vertices does not change very much from 2-clique to 5-clique. Thus CommunityGAN can benefit from the larger cliques to gather better community structure, and then reaches the best performance when clique size is 4

or 5. However, when the size of clique reaches 6, the number of covered vertices drops dramatically, and thus the performance of CommunityGAN decreases.

5.7 Discussion

With the two experiments on the series of synthetic graphs, we can draw the following conclusions:

- With the same special vector design as AGM, CommunityGAN holds the ability to solve the dense overlapping problem. With the increase of overlapping among communities, the performance of CommunityGAN can maintain steadiness and does not fall like the baseline methods.
- With the minimax game competition between the discriminator and generator, CommunityGAN gains significant performance improvements based on AGM.
- With proper sizes, the motif-level generation and discrimination can help models learn better community structures compared to edge-level optimization.

6 REAL-WORLD SCENARIOS

To complement the previous experiments, in this section, we evaluate the performance of CommunityGAN on a series of real-world datasets. Specifically, we choose two application scenarios for experiments, i.e., community detection and clique prediction.

6.1 Datasets

We evaluate our model using two categories of datasets, one with ground-truth communities and one without.

Datasets with ground-truth communities.[2] Because of the training time of some baselines, we only sample three subgraphs with 100 ground-truth communities as the experiment networks from these three large networks.

---

[2] http://snap.stanford.edu/data/#communities

Table 6: F1-Score and NMI on community detection.

F1-Score			
Model	Amazon	Youtube	DBLP
MMSB	0.366	0.124	0.104
CPM	0.214	0.058	0.318
AGM	0.711	0.158	0.398
node2vec	0.550	0.288	0.265
LINE	0.532	0.170	0.208
GraphGAN	0.518	0.303	0.276
ComE	0.562	0.213	0.240
CommunityGAN	**0.860**	**0.327**	**0.456**

NMI			
Model	Amazon	Youtube	DBLP
MMSB	0.068	0.031	0.000
CPM	0.027	0.000	0.066
AGM	0.635	0.025	0.059
node2vec	0.370	0.071	0.068
LINE	0.248	0.070	0.027
GraphGAN	0.417	0.049	0.083
ComE	0.413	**0.091**	0.059
CommunityGAN	**0.853**	**0.091**	**0.153**

Amazon is collected by crawling Amazon website. The vertices represent products; the edges indicate the frequently co-purchase relationships; the ground-truth communities are defined by the product categories in Amazon. This graph has 3,225 vertices and 10,262 edges.

Youtube is a network of social relationships of Youtube website users. The vertices represent users; the edges indicate friendships among the users; the user-defined groups are considered as ground-truth communities. This graph has 4,890 vertices and 20,787 edges.

DBLP is a co-authorship network from DBLP. The vertices represent researchers; the edges indicate the co-author relationships; authors who have published in a same journal or conference form a community. This graph has 10,824 vertices and 38,732 edges.

Datasets without ground-truth communities.[3]

arXiv-AstroPh is from the e-print arXiv and covers scientific collaborations between authors with papers submitted to the Astro Physics category. The vertices represent authors, and the edges indicate co-author relationships. This graph has 18,772 vertices and 198,110 edges.

arXiv-GrQc is also from arXiv and covers scientific collaborations between authors with papers submitted to the General Relativity and Quantum Cosmology categories. This graph has 5,242 vertices and 14,496 edges.

6.2 Community Detection

The community detection experiment is conducted on three networks with ground-truth community memberships, which allows us

---

[3] http://snap.stanford.edu/data/#canets
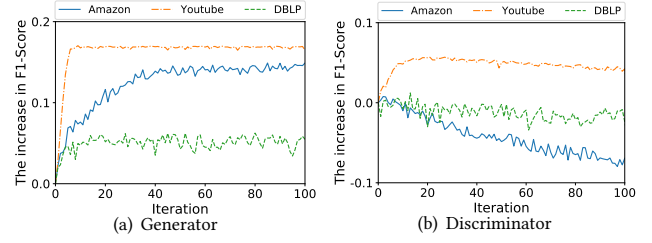
Figure 8: Learning curves of the generator and the discriminator of CommunityGAN in community detection.

to quantify the accuracy of community detection methods by evaluating the level of correspondence between detected and ground-truth communities.

Setup. In the community detection experiment, all the baselines and setups are the same as the synthetic data experiments, whose details refer to §5.4.

Evaluation Metric. Besides the F1-Score described in synthetic dataset experiments (refer to §5.1), to comprehensively evaluate the performance of methods, we also introduce the Normalized Mutual Information (NMI) in the community detection experiment. NMI is a measure of similarity borrowed from information theory. Later, it is extended to measure the quality of overlapping communities. Similar to F1-Score, the higher NMI means the better performance. Please refer to [13] for details.

Results. Table 6 shows the F1-Score and NMI of 8 methods on 3 networks, respectively. As we can see: 1) Performance of CPM, MMSB and LINE is relatively poor in community detection, which means they cannot capture the pattern of community structure in graphs well. 2) Node2vec, ComE and GraphGAN perform better than CPM, MMSB and LINE. One possible reason is that node2vec and GraphGAN adopt the random walk to capture the vertex context which is similar to community structure in some way, and ComE solves the network embedding and community detection task in one unified framework. 3) CommunityGAN outperforms all the baselines (including its pre-train method AGM) in community detection task. Specifically, CommunityGAN improves F1-Score by 7.9% (on Youtube) to 21.0% (on Amazon) and improves NMI by at most 34.3% (on Amazon). Our explanation is that the consideration of dense community overlapping provides CommunityGAN a higher learning flexibility than the non-overlapping or sparse overlapping community detection baselines. Moreover, the minimax game between the generator and discriminator drives CommunityGAN to gain significant performance improvements based on AGM.

To intuitively understand the learning stability of CommunityGAN, we further illustrate the learning curves of the generator and the discriminator on the three datasets in Figure 8. Because the F1-Score on the three datasets differs very much, we only demonstrate the relative increase in the score. As we can see, the generator performs outstandingly well after convergence, while the performance of the discriminator falls a little or boosts at first and then falls. Maybe it is because that the discriminator aims to distinguish

Table 7: AUC on clique prediction.

arXiv-AstroPh			
Model	2-clique	3-clique	4-clique
AGM	0.919	0.987	0.959
node2vec	0.579	0.514	0.544
LINE	0.918	0.980	0.963
GraphGAN	0.799	0.859	0.855
ComE	0.904	0.951	0.953
CommunityGAN	**0.923**	**0.990**	**0.970**

arXiv-GrQc			
Model	2-clique	3-clique	4-clique
AGM	0.900	0.980	0.871
node2vec	0.632	0.569	0.534
LINE	**0.969**	0.989	0.880
GraphGAN	0.756	0.880	0.728
ComE	0.924	0.962	0.914
CommunityGAN	0.904	**0.993**	**0.956**

Figure 9: Performance as training time (Sec.).

ground truth from generated samples better. However, the generated samples are all around the center vertex, which causes the discriminator to lose some global discrimination ability.

6.3 Clique Prediction

In clique prediction, our goal is to predict whether a given subset of vertices is a clique. Therefore, this task shows the performance of graph local structure extraction ability of different graph representation learning methods.

Setup. In the clique prediction experiment, because some traditional community detection methods (including MMSB and CPM) cannot predict the existence of edges among vertices, these methods are omitted in this experiment. To analyze the effect of motif generation and discrimination in CommunityGAN, in this experiment, we have evaluated the prediction for 2-clique (same to edge), 3-clique and 4-clique. With the size of cliques determined, we randomly hide some cliques, which cover 10% of edges, in the original graph as ground truth, and use the left graph to train all graph representation learning models. After training, we obtain the representation vectors for all vertices and use logistic regression method to predict the probability of being clique for a given vertex set. Our test set consists of the hidden vertex sets (cliques) in the original graph as the positive samples and the randomly selected non-fully connected vertex sets as negative samples with the equal number.

Results. We use arXiv-AstroPh and arXiv-GrQc as datasets, and report the results of AUC in Table 7. As we can see, even though in 2-clique (edge) prediction CommunityGAN does not always get the highest score, CommunityGAN outperforms all the baselines in both 3-clique and 4-clique prediction. For example, on arXiv-GrQc, CommunityGAN achieves gains of 0.40% to 74.52% and 4.60% to 79.03% in 3-clique and 4-clique prediction, respectively. This indicates that though CommunityGAN is designed for community detection, with the design of motif generation and discrimination,
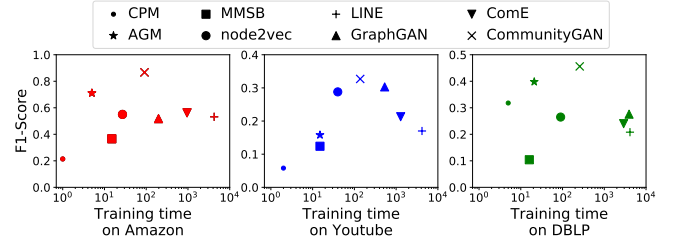
it can still effectively encode the information of clique structures into the learned representations.

6.4 Efficiency analysis

In this paper, we propose Graph AGM for efficiently generating the most likely motifs with graph structure awareness. Because of the random walk process, in which the exact time complexity is not easy to infer, we evaluate the efficiency of CommunityGAN by directly comparing the training time with baselines. In this evaluation, the number of threads is set as 16 if the model supports parallelization and other parameters for baselines are as default. Figure 9 illustrates the performance and training time. Notably, the training time of CommunityGAN includes the time of the pretraining process. Even though CommunityGAN is not the fastest model, the training time of CommunityGAN is still acceptable and its performance significantly outperforms the faster models.

7 CONCLUSION

In this paper we proposed CommunityGAN that jointly solves the overlapping community detection and graph representation learning. Unlike the embedding of general graph learning algorithms in which the vector values have no meanings, the embedding in CommunityGAN indicates the membership strength of vertices to communities, which enables CommunityGAN to detect densely overlapped communities. Then a specifically designed GAN is adopted to optimize such embedding. Through the minimax game of motif-level generator and discriminator, both of them can boost their performance and finally output better community structures. We adopted CKB Graph Generator to create a series of synthetic graphs with ground-truth communities. Two experiments were conducted on these graphs to prove the ability of CommunityGAN to solve dense overlapping problem and its efficacy of motif generation and discrimination. Additionally, to complement the experiments on the synthetic datasets, we did experiments on five real-world datasets in two scenarios, where the results demonstrate that CommunityGAN substantially outperforms baselines in all experiments due to its specific embedding design and motif-level optimization.

8 ACKNOWLEDGEMENTS

REFERENCES

[1] Edo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. 2009. Mixed membership stochastic blockmodels. In *Advances in Neural Information Processing Systems*. 33–40.

[2] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning Graph Representations with Global Structural Information. In *CIKM*. ACM, New York, NY, USA, 891–900. https://doi.org/10.1145/2806416.2806512

[3] Sandro Cavallari, Vincent W Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. 2017. Learning community embedding with community detection and node embedding on graphs. In *CIKM*. ACM, 377–386.

[4] Kyrylo Chykhradze, Anton Korshunov, Nazar Buzun, Roman Pastukhov, Nikolay Kuzyurin, Denis Turdakov, and Hangkyu Kim. 2014. Distributed Generation of Billion-node Social Graphs with Overlapping Community Structure. In *Complex Networks V*, Pierluigi Contucci, Ronaldo Menezes, Andrea Omicini, and Julia Poncela-Casasnovas (Eds.). Springer International Publishing, Cham, 199–208.

[5] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. 2017. Adversarial Network Embedding. *CoRR* abs/1711.07838 (2017). arXiv:1711.07838 http://arxiv.org/abs/1711.07838

[6] Sheng Gao, Ludovic Denoyer, and Patrick Gallinari. 2011. Temporal link prediction by integrating content and structure information. In *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 1169–1174.

[7] D. F. Gleich and C. Seshadhri. 2012. Neighborhoods are good communities. In *KDD'12*.

[8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. *CoRR* abs/1607.00653 (2016). arXiv:1607.00653 http://arxiv.org/abs/1607.00653

[9] Zhiting Hu, Junjie Yao, Bin Cui, and Eric Xing. 2015. Community level diffusion extraction. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 1555–1569.

[10] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR* abs/1609.02907 (2016). arXiv:1609.02907 http://arxiv.org/abs/1609.02907

[11] Christine Klymko, David F. Gleich, and Tamara G. Kolda. 2014. Using Triangles to Improve Community Detection in Directed Networks. *CoRR* abs/1404.5874 (2014). arXiv:1404.5874 http://arxiv.org/abs/1404.5874

[12] Nevan J Krogan, Gerard Cagney, Haiyuan Yu, Gouqing Zhong, Xinghua Guo, Alexandr Ignatchenko, Joyce Li, Shuye Pu, Nira Datta, Aaron P Tikuisis, et al. 2006. Global landscape of protein complexes in the yeast Saccharomyces cerevisiae. *Nature* 440, 7084 (2006), 637–643.

[13] Andrea Lancichinetti, Santo Fortunato, and János Kertész. 2009. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics* 11, 3 (2009), 033015.

[14] Ye Li, Chaofeng Sha, Xin Huang, and Yanchun Zhang. 2018. Community Detection in Attributed Graphs: An Embedding Approach. In *AAAI Conference on Artificial Intelligence*. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17142

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *CoRR* abs/1310.4546 (2013). arXiv:1310.4546 http://arxiv.org/abs/1310.4546

[16] Mark EJ Newman. 2004. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems* 38, 2 (2004), 321–330.

[17] Mark EJ Newman. 2006. Modularity and community structure in networks. *Proceedings of the national academy of sciences* 103, 23 (2006), 8577–8582.

[18] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *Nature* 435, 7043 (2005), 814–818.

[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. *CoRR* abs/1403.6652 (2014). arXiv:1403.6652 http://arxiv.org/abs/1403.6652

[20] Yiye Ruan, David Fuhry, and Srinivasan Parthasarathy. 2013. Efficient community detection in large networks using content and links. In *Proceedings of the 22nd international conference on World Wide Web*. 1089–1098.

[21] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. 2015. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*. 3528–3536.

[22] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. 1057–1063.

[23] Jiliang Tang, Charu Aggarwal, and Huan Liu. 2016. Node classification in signed social networks. In *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 54–62.

[24] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. *CoRR* abs/1503.03578 (2015). arXiv:1503.03578 http://arxiv.org/abs/1503.03578

[25] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. GraphGAN: Graph Representation Learning With Generative Adversarial Nets. In *AAAI Conference on Artificial Intelligence*. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16611

[26] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. 2018. SHINE: signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 592–600.

[27] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding.. In *AAAI*. 203–209.

[28] Ju Xiang, Tao Hu, Yan Zhang, Ke Hu, Jian-Ming Li, Xiao-Ke Xu, Cui-Cui Liu, and Shi Chen. 2016. Local modularity for community detection in complex networks. *Physica A: Statistical Mechanics and its Applications* 443 (2016), 451–459.

[29] Jierui Xie, Stephen Kelley, and Boleslaw K. Szymanski. 2011. Overlapping Community Detection in Networks: the State of the Art and Comparative Study. *CoRR* abs/1110.5813 (2011). arXiv:1110.5813 http://arxiv.org/abs/1110.5813

[30] Jaewon Yang and Jure Leskovec. 2012. Community-affiliation graph model for overlapping network community detection. In *2012 IEEE 12th International Conference on Data Mining*. IEEE, 1170–1175.

[31] Jaewon Yang and Jure Leskovec. 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*. ACM, 587–596.

[32] Jaewon Yang, Julian McAuley, and Jure Leskovec. 2014. Detecting cohesive and 2-mode communities indirected and undirected networks. In *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 323–332.

[33] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient.. In *AAAI*. 2852–2858.

[34] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 283–292.

[35] Chuxu Zhang, Lu Yu, Yan Wang, Chirag Shah, and Xiangliang Zhang. 2017. *Collaborative User Network Embedding for Social Recommender Systems*. 381–389. https://doi.org/10.1137/1.9781611974973.43 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611974973.43

[36] Hongyi Zhang, Irwin King, and Michael R. Lyu. 2015. Incorporating Implicit Link Preference into Overlapping Community Detection. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*. AAAI Press, 396–402. http://dl.acm.org/citation.cfm?id=2887007.2887063