Sandro Cavallari
School of Computer Science and Engineering, Nanyang Technological University, SINGAPORE

Erik Cambria
School of Computer Science and Engineering, Nanyang Technological University, SINGAPORE

Hongyun Cai
Advanced Digital Sciences Center, SINGAPORE

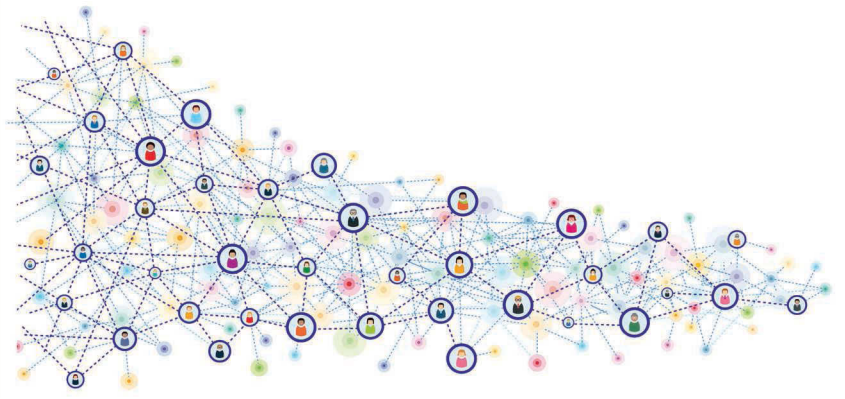
Kevin Chen-Chuan Chang
Department of Computer Science, University of Illinois at Urbana-Champaign, USA

Vincent W. Zheng
Advanced Digital Sciences Center, SINGAPORE

Embedding Both Finite and Infinite Communities on Graphs

Abstract

In this paper, we introduce a new setting for graph embedding, which considers embedding communities instead of individual nodes. We find that community embedding is not only useful for community-level applications such as graph visualization but also provide an exciting opportunity to improve community detection and node classification. Specifically, we consider the interaction between community embedding and detection as a closed loop, through node embedding. On the one hand, node embedding can improve community detection since the detected communities are used to fit a community embedding. On the other hand, community embedding can be used to optimize node embedding by introducing a community-aware high-order proximity. However, in practice, the number of communities can be unknown beforehand; thus we extend our previous Community Embedding (ComE) model. We propose ComE+, a new model which handles both: the unknown truth community assignments and the unknown number of communities present in the dataset. We further develop an efficient inference algorithm for ComE+ for keeping complexity low. Our extensive evaluation shows that ComE+ improves the state-of-the-art baselines in various application tasks, e.g., community detection and node classification.



©ISTOCKPHOTO.COM/AELITTA

I. Introduction

Graphs are a representational framework for many real-world applications, e.g., social media analytics, user's interest diffusion, knowledge representation and extraction, protein interaction, and others. Analyzing such graphs is a key factor to take strategic decisions. Thus, it has received significant attention in the last few decades. Traditionally, graph embedding, which aims to represent a graph structure on a d-dimensional space preserving some predefined graph's properties, became a popular technique to extract the hidden knowledge present in such a rich structure. However, to date research has primarily focused on a node level representation where each node is associated to a vector, such that two "related" nodes have similar vector representations [1].

While most of the previous works focus on the microscopic architectures formed between nodes [2]–[9], in this paper, we also focus on preserving the communities structures formed by densely connected nodes. Community embedding is useful for many community-level

Digital Object Identifier 10.1109/MCI.2019.2919396
Date of publication: 17 July 2019

Corresponding Author: Vincent W. Zheng (Email: vincent.zheng@adsc-create.edu.sg).

Given that a group of densely connected nodes forms a community, its embedding is expected to characterize how the nodes' member spreads in the low-dimensional space.

applications, e.g., for community visualization to help generate insights from big graphs, or community recommendation to search for similar communities.

If the concept of community appears self-explanatory, there is no commonly accepted quantitative definition for it. Indeed, communities are one of the most prominent and complex networks' features since blurred overlapping boundaries characterize them whereas nodes can simultaneously belong to different communities at the same time. According to the general definition: a community embedding should be a low-dimensional representation for a community. Given that a group of densely connected nodes forms a community, its embedding is expected to characterize how the nodes' member spreads in the low-dimensional space. As a consequence, we need to define the community embedding as a **distribution** over the same node embedding space, instead of using a new d'-dimensional vector to represent a community. To this end, we are inspired by the Gaussian Mixture Model (GMM) [10] to obtain a probabilistic representation where each community is defined as a multivariate Gaussian distribution in the node embedding space. In so doing, each node affects, in big or small proportion, the representation of each community, while a community provides an explicit representation of how the nodes spread in the latent space.

In Fig. 1, we use the well-studied Karate Club [11] graph to demonstrate community embedding in a 2D space. As shown in Fig. 1a, the Karate Club graph represents the friendship connection among the members of a university Karate club. The graph is characterized by 34 nodes, 78 edges and the presence of 2 distinct communities formed after the club partitioned due to some internal management divergence. It is also known that during the partitioning phase, some members are declared supporters of the instructor's partition (node 1), while others are supporters of the club administrator (node 34). Among the others, some nodes (i.e. node 9) are known to support both groups. We denote such nodes as "weak supporters" of both communities since they neither fully support the administrator nor the instructor partition. In Fig. 1c and 1d, we visualize both the nodes and their communities in a 2D space respectively for ComE and ComE+. Consequently, as shown in Fig. 1, according to our definition, we visualize the communities as two ellipses, each of which is characterized by a 2D mean vector and a 2×2 co-variance matrix.

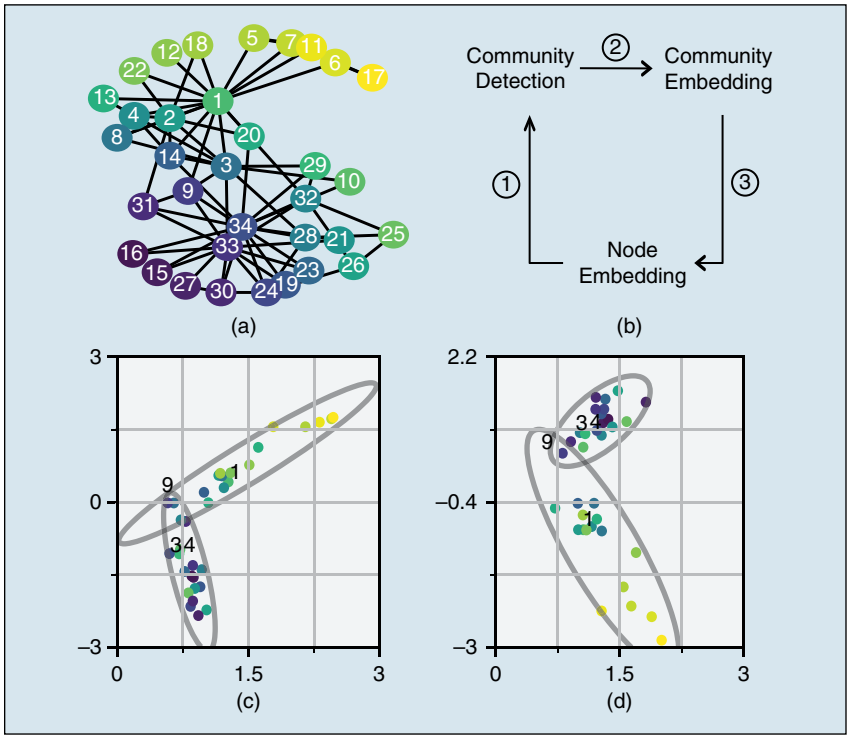


FIGURE 1 Embedding nodes and communities in a 2D space. (a) Karate club graph, (b) closed loop for community embedding, (c) ComE visualization and (d) ComE+ visualization.

Learning community embedding is non-trivial. To have meaningful community embedding, we first need to identify the communities accurately. A straightforward approach to community embedding is to: 1) run community detection, such as Spectral Clustering [12], on the graph Laplacian matrix to get community assignments for each node; 2) apply node embedding, such as DeepWalk [3] or Node2Vec [13], on the graph to get an embedding vector for each node; 3) aggregate the node embedding vectors in each community, to fit a (multivariate Gaussian) distribution as its community embedding. Such a pipeline approach is sub-optimal because its community detection is independent of its node embedding. Moreover, in practice, it is possible that the number of communities is unknown in advance.

Recent works, such as [2], [6], [14], demonstrated that node embedding improves community detection performance since can effectively preserve the network's structure. However, few works consider node embedding and community detection together; they either require extra supervision [7] or high computational complexity [15]. Furthermore, they embed a community in the low-dimensional space as a vector; thus they do not provide an easily interpretable representation.

As shown in Fig. 1b, we propose the existence of a **closed loop** between node embedding, community detection and community embedding. On the one hand, as overstated, node embedding has been proven useful to solve the community detection problem (i.e., ①). The communities assignment obtained can be used to fit meaningful community embedding (i.e., ②). On the other hand, node embedding could be enhanced by a relevant community embedding (i.e., ③), since related but unconnected nodes can be enforced to be similar in the embedding space.

Suppose for a community k to already has its community embedding as a multivariate distribution in a low-dimensional space. Then, we can enforce community k's member nodes to scatter nearly its community embedding's centroid even if the nodes are not directly linked together. Compared with first- and second-order proximity, community embedding does not focus on well-defined microscopic structure but instead focus on the complex mesoscopic construction generated by densely connected nodes. This introduces a **community-aware high-order proximity** to the node embedding.

Guided by the closed loop, in **ComE** [22], we formally define community embedding as a multivariate Gaussian distribution. Let us denote a graph as $G=(V,E)$, where V is the set of nodes, and E is the set of edges. We introduce node embedding to incorporate both first- and second-order proximity on G. Based on the node embedding, we detect K communities (K is known and finite) and fit their community embedding. As a closed loop, community assignments and community embedding can help node embedding to preserve the high-order community-aware proximity. Then, we develop an efficient iterative optimization algorithm for inference with community detection, community embedding and node embedding.

In this work, we further propose **ComE+** for community embedding with an unknown number of communities (i.e., K is unknown and can be infinite), which is a typical scenario in practice. Note that recent studies [23]–[25] demonstrated the absence of a universal consensus on which is the best model for community detection; while they also highlight how the graph's metadata is not always related to the mesoscopic network's structure. Hence, it is important to maintain a self-explanatory community structure and automatically detect the number of communities from the data distribution. For ComE+, we also aim to model the closed loop in Fig. 1b, yet with special consideration of modeling an infinite number of communities in detection and embedding. To realize this infinite community embedding, we are inspired by Infinite Gaussian Mixture (IGM) [26] to adopt a Bayesian modeling framework. We first take a simple stick-breaking construction approach [27] to generate up to an infinite number of communities and compute each node's community assignment accordingly. Then, we sample the node embedding from the community embedding, and later optimize it w.r.t. the first- and second-order proximity. We then develop an efficient iterative inference algorithm for ComE+. Given a finite dataset, our inference algorithm can let the data automatically decide the exact number of communities for embedding. In Fig. 1d, we show that ComE+ manages to correctly infer

TABLE I Comparison with related work. Here, "●" indicates an property as explicitly defined, and "○" indicates an property as implicitly derived. Complexity is defined over only graph-related factors.

	NODE EMBEDDING						
	1ST-ORDER	2ND-ORDER	HIGHER-ORDER	COMM. EMBED.	COMM. DETECT.	INFINITE COMM.	COMPLEXITY
DEEPWALK [3]		●					$O(\|V\|\log\|V\|)$
NODE2VEC [13]		●					$O(\|V\|\log\|V\|+\|V\|a^2)$
LINE [16]	●	●					$O(a\|E\|)$
SDNE [17]	●	●					$O(a\|V\|)$
GRAREP [2]			●				$O(\|V\|^3)$
STRUC2VEC [18]		●					$O(\|V\|^3)$
METAPATH2VEC [19]		●					$O(\|V\|)$
SPECTRAL [12]					●		$O(\|V\|^3)$
PRUNE [20]	●	○	○				$O(\|E\|)$
DNR [7]			●		●		$O(\|V\|^2)$
M-NMF [15]	●	●	●	○	●		$O(a\|V\|^2)$
LPCM [21]	●		●	●	●		$O(\|V\|^2)$
COME [22]	●	●	●	●	●		$O(\|V\|+\|E\|)$
COME+ (THIS WORK)	●	●	●	●	●	●	$O(\|V\|+\|E\|)$

$K = 2$ and embed the two overlapping communities in the 2D space.

We summarize our contributions as follows:

- We contribute with a comprehensive literature review and complexity analysis summarized in Tab. I.
- We further extend ComE to ComE+ to better handle the unknown number of communities for community embedding. To the best of our knowledge, this is the first node embedding model able to automatically infer the number of communities from the data distribution while preserving this structure during the embedding process.
- We designed a new variational inference model for ComE+ while maintaining the inference algorithm complexity as low as $O(|V| + |E|)$.
- We evaluate ComE+ on seven public real-world datasets with various application tasks. We relatively improve state-of-the-art baselines across datasets by at least 0.3%–17.1% (Conductance) and 0.8%–6.7% (NMI) in community detection, 0.8%–25.4% (Macro-F1) and 0.3%–34.7% (Micro-F1) in node classification. Detailed settings for these improvements are reported in Sec. V.

II. Related Work

As our task uses graphs as input, we first review the literature on graph embedding. Then we review the literature on community detection since it is our key notion. Finally, as we use a Bayesian definition for community embedding, we review the literature on Bayesian embedding.

A. Graph Embedding

Most graph embedding work focuses on nodes [28]–[30]. For example, earlier methods, such as Laplacian eigenmap [31] and IsoMap [32], aim to preserve first-order proximity by extracting leading eigenvectors of a graph affinity matrix. Recent work starts to exploit deep learning to learn node representations. For example, DeepWalk [3], models second-order proximity based on path sampling, and it has an inference complexity of $O(|V| \log |V|)$. Node2Vec [13] extends DeepWalk with a controlled path sampling process, which requires an additional $O(|V|a^2)$ factor where a is the average degree of the graph. LINE [16] and SDNE [17] preserve both first- and second-order proximity at the price of a higher complexity with $O(a|E|)$ and $O(a|V|)$, respectively. Both metapath2vec [19] and struct2vec [18] consider second-order proximity for node embedding. GraRep [2] and HOPE [33] preserve high-order proximity by learning node embedding from multi-step Katz index matrix or PageRank matrix. Particularly, GraRep runs Singular Value Decomposition (SVD), which requires $O(|V|^3)$ in general. Compared with our ComE and ComE+, the above work neither explicitly detects nor represents the communities.

Community structure is an important network property, yet it is underexplored in graph embedding. For example, SAE [6] builds a normalized similarity matrix with complexity $O(|E|)$, and learns node embedding by reconstructing the matrix by stacked Auto-Encoder with $O(|V|)$ complexity. It further applies K-means clustering to detect the communities. DNR [7] constructs a modularity matrix from the graph with $O(|V|^2)$ complexity, then it applies stacked Auto-Encoder on the matrix with must-link supervision to generate node embedding. PRUNE [20] constructs a Pointwise Mutual Information (PMI) matrix to model global node ranking and first-order proximity by $O(|E|)$. Both the PMI matrix and the edge representation are argued to implicitly preserve the community structure as well as the second-order proximity. There is limited work on explicitly modeling communities in graph embedding. For example, M-NMF [15] constructs the modularity matrix with $O(|V|^2)$ complexity, then applies non-negative matrix factorization to learn node embedding and detect communities together with $O(|V|^2)$ again. Compared with our work, M-NMF uses the modularity as a distance metric and embeds each community with a vector instead of a distribution; thus it is unable to fully characterize the community. Finally, refer to [1] for a comprehensive survey on graph embedding methods.

B. Community Detection

Community detection aims to discover groups of nodes in a graph, such that the intra-group connections are denser than the inter-group ones [34]. Most of existing work first embeds the nodes in a low-dimensional space by either feature engineering, graph embedding or linear coding [35]. Then they apply clustering, e.g., spectral clustering [12], Laplacian regularized GMM [36] or neural networks [6], [14], on the nodes for community detection. Despite the success of these methods, their results are suboptimal since their node embedding does not explicitly consider the community structure.

Recent work advances community detection by exploiting rich types of interaction on a social network such as content [37], attributes [38] and information diffusion [39]. Additionally, some methods exploit opportunities of coupling community detection with other tasks, such as hole spanner detection [40] or social role detection [41]. Comparatively, we are different in that: 1) we particularly consider graph embedding, whereas the above community detection work generally does not; 2) we focus on the most basic graph setting where we have a homogeneous graph as input, whereas some community detection work may require additional inputs of the graph.

C. Bayesian Embedding

Most graph embedding tries to embed a node into a "point" vector space [3], [42]. As a community has to characterize both what its member nodes are and how they spread, we consider a Bayesian definition of community embedding. That is: we define each community embedding as a distribution rather than a point in the low-dimensional space. Little work considers Bayesian embedding. For example, GenVector [43] embeds each social network user and each word as a

vector, then tries to infer latent topics from each user's social posts, which are assumed to generate the user/word embedding. However, our work is different: 1) GenVector considers graphs with content, but we consider basic homogeneous graphs; 2) it neither considers community nor tries to improve node embedding by community detection.

There is some close work to consider communities in Bayesian graph embedding. For example, LPCM (latent position cluster model) [21] embeds each node as a vector, and models the first-order proximity by a logistic function. It also models each node's embedding as generated by its communities' multivariate Gaussian distributions. However, its number of communities is finite and given. An extension of LPCM is considered in [44] with a more efficient inference algorithm, and a different likelihood function of LPCM is explored in [45] for quantifying the residual uncertainty of model parameters. Our work is different from the above work: 1) they cannot handle an infinite number of communities; 2) they overlook the second-order proximity in node embedding.

III. Problem Formulation

Traditional graph embedding aims to learn a node embedding for each $v_i \in V$ as $\phi_i \in \mathbb{R}^d$ while we also try to learn community embedding. Suppose there are K communities in a graph G, where K can be either known or unknown. For each node v_i, we denote its community assignment as $z_i \in \{1, \ldots, K\}$. Then, we can formally define a community embedding as Def. III.1.

Definition III.1

Community embedding of a community k (with $k \in \{1, \ldots, K\}$) in a d-dimensional space is a multivariate Gaussian distribution $\mathcal{N}(\psi_k, \Sigma_k)$, where $\psi_k \in \mathbb{R}^d$ is a mean vector and $\Sigma_k \in \mathbb{R}^{d \times d}$ is a covariance matrix.

As a final goal we aim to learn:

1) node embedding ϕ_i for each node $v_i \in V$;
2) community membership π_{ik}, such that $\sum_{k=1}^{K} \pi_{ik} = 1$, for each node $v_i \in V$ and each community $k \in \{1, \ldots, K\}$;
3) community embedding parameters (ψ_k, Σ_k) for each community $k \in \{1, \ldots, K\}$.

To jointly optimize node embedding and community embedding, we exploit a similar coordinated learning process like the one proposed in ComE. That is, we also apply iterative optimization between (Φ, Φ′) and *q*.

In the next section, we are going to define the learning procedure for the node embedding and the community embedding with an infinite the number of communities[1].

A. Node Embedding

Traditionally, node embedding focuses on preserving first- or second-order proximity between nodes. To this end, ComE+ share the same node embedding formulation originally proposed in ComE [22].

First-Order Proximity

Based on LINE [16], we define the first-order proximity as:

$$O_1 = -\sum_{(v_i, v_l) \in E} \log \sigma(\phi_l^T \phi_i), \qquad (1)$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is a sigmoid function. In doing so, Eq. 1 enforces direct neighbors nodes to have similar embedding.

Second-Order Proximity

DeepWalk [3] is the first work that proposes a path sampling approach to preserve the neighbors' similarity. To achieve its purpose, each node is exploiting two roles: as a node for itself and as context for a "close" node in the path. Thus an extra context embedding $\phi'_j \in \mathbb{R}^d$ is introduced. In the original DeepWalk statement a hierarchical softmax generative process was proposed, instead based on ComE we adopt a negative sampling [46] approach to approximate how well v_i generates its contexts nodes $v_j \in C_i$. We formulate the learning process as:

$$\Delta_{ij} = \log \sigma(\phi'^T_j \phi_i) + \sum_{t=1}^{m} \mathbb{E}_{v_l \sim P_n(v_l)} [\log \sigma(-\phi'^T_l \phi_i)], \qquad (2)$$

where $v_l \sim P_n(v_l)$ denotes sampling a node $v_l \in V$ as a "negative context" of v_i according to a probability $P_n(v_l)$. As described in [22] we set $P_n(v_l) \propto a_{v_l}^{3/4}$ where a_{v_l} is v_l's degree and m represent the total number of negative contexts. Finally, to preserve the second order proximity, the subsequent loss is used:

$$O_2 = -\alpha \sum_{v_i \in V} \sum_{v_j \in C_i} \Delta_{ij}, \qquad (3)$$

where $\alpha > 0$ is a trade-off parameter.

B. Community Detection and Embedding

Based on the multivariate Gaussian distribution $\mathcal{N}(\psi_k, \Sigma_k)$ as formulation for a community $z_i = k$. In ComE+ we are rather inspired by IGM [26] to sample the node embedding from an infinite number of (ψ_k, Σ_k)'s. This means that we have both ψ_k and Σ_k governed by some prior distributions. Note that we are technically different from IGM in two aspects. Firstly, instead of trying to derive the posterior under an infinity limit, we take a simpler stick-breaking construction approach [27] to generate an infinite number of communities and sample each node's community assignment. Secondly, we take a **variational inference** approach to approximate the posterior for the community embedding's random variables.

Overall, it is possible to summarize the generative process for ComE+ as:

- Draw $\pi_k \sim \mathcal{B}(1, \rho)$, for $k = 1, 2, \ldots$;
- Draw $\psi_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, for $k = 1, 2, \ldots$;
- Draw $\Sigma_k^{-1} \sim \mathcal{W}(\mathbf{I}, \nu)$, for $k = 1, 2, \ldots$;
- For each node v_i:

[1] A graphical model for ComE+ is provided at http://sentic.net/graph-model.pdf

– Draw $z_i \sim SBC(\boldsymbol{\pi})$;
– Draw $\phi_i \sim \mathcal{N}(\psi_{z_i}, \Sigma_{z_i})$.

Specifically, we denote $\mathcal{B}(a, b)$ as a Beta distribution, where $a > 0$ and $b > 0$ are the parameters. We denote $\mathcal{W}(A, \nu)$ as a Wishart distribution, with a positive definite scale matrix $A \in \mathbb{R}^{d \times d}$ and degrees of freedom $\nu > d - 1$. Denote $\mathbf{I}$ as a $d \times d$ dimensional identity matrix. We denote $\rho > 0$ and $\nu > 0$ as the hyper-parameters for the prior distributions of ψ_k and Σ_k respectively, describing how many latent components to activate and how small the Gaussian covariance can be. Additionally, $z_i \sim SBC(\boldsymbol{\pi})$ is a stick-breaking construction, where given an infinite dimensional vector $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots]$, we compute the mixture probability for each community k as

$$\eta_k = \pi_k \prod_{j=1}^{k-1} (1 - \pi_j), \quad \text{for } k = 1, 2, \ldots. \tag{4}$$

Based on the resulting infinite dimensional vector $\boldsymbol{\eta} = [\eta_1, \eta_2, \ldots]$, we draw the community assignment indicator $z_i \in \{1, 2, \ldots\}$ for each node v_i by a multinomial distribution:

$$z_i \sim \text{Multi}(\boldsymbol{\eta}), \quad \text{for } i = 1, \ldots, |V|. \tag{5}$$

For notation simplicity, we denote $Z = \{z_1, \ldots, z_N\}$ as the community assignments for N nodes and $\Omega = \{(\psi_1, \Sigma_1), (\psi_2, \Sigma_2), \ldots\}$ as the infinite community embeddings. Finally, we denote $W = \{\pi, \Omega, Z\}$. Given the node embeddings $\boldsymbol{\Phi}$ for a graph G, the primary purpose is to infer the posterior distribution $p(W|\boldsymbol{\Phi}; \rho, \nu)$, which identifies community assignments and community embedding.

C. Closing the Loop

As for ComE, also in ComE+, we aim to realize the closed loop shown in Fig. 1b, yet in an infinite community setting. Effectively, learning node embedding requires a Maximum A Posteriori (MAP) estimation, where we see O_1 (Eq. 1) and O_2 (Eq. 3) as modeling the likelihood of ϕ_i's, and $\phi_i \sim \mathcal{N}(\psi_{z_i}, \Sigma_{z_i})$ as modeling the prior. In contrast, to handle the infinite number of communities, we need Bayesian inference. Thus, to optimize the ComE+ model we need to manage MAP estimation and Bayesian inference together. In general, there can be different approaches to Bayesian inference. For example, [21], [43], [45] adopt a Markov chain Monte Carlo (MCMC) methods; to the other side [47] optimize the likelihood function by stochastic gradient descent (SGD). In this work, we propose to use a variational inference formulation as the objective function of ComE+. However, we resort to variational inference to develop an analytical form, which directly infers the posterior distribution of the Gaussian random variables and encodes the node embedding prior by $\mathcal{N}(\psi_{z_i}, \Sigma_{z_i})$. Specifically, we introduce a variational distribution $q(W)$ to approximate the posterior $p(W|\boldsymbol{\Phi}; \rho, \nu)$, by minimizing the KL-divergence between them:

$$\begin{aligned} &KL(q(W) \,|\, p(W \,|\, \boldsymbol{\Phi}; \rho, \nu)) \\ &= \mathbb{E}_q\left[\log \frac{q(W)}{p(W|\boldsymbol{\Phi}; \rho, \nu)}\right] \\ &= \mathbb{E}_q[\log q(W)] - \mathbb{E}_q[\log p(W|\boldsymbol{\Phi}; \rho, \nu)] \\ &= \mathbb{E}_q[\log q(W)] - \mathbb{E}_q[\log p(W, \boldsymbol{\Phi}|\rho, \nu)] \\ &\quad + \log p(\boldsymbol{\Phi}|\rho, \nu). \end{aligned} \tag{6}$$

As $\log p(\boldsymbol{\Phi}|\rho, \nu)$ does not depend on q, minimizing Eq. 6 is equivalent to:

$$\begin{aligned} &\max_q \mathbb{E}_q[\log p(W, \boldsymbol{\Phi}|\rho, \nu)] \\ &\quad - \mathbb{E}_q[\log q(W)]. \end{aligned} \tag{7}$$

Notice that $p(W, \boldsymbol{\Phi}|\rho, \nu)$ is the joint probability describing our generative process in Sec. III.B. Thus it naturally includes the formulation of $\mathcal{N}(\psi_{z_i}, \Sigma_{z_i})$, which can be used to optimize together with O_1 and O_2. To this end, we define the objective function for inferring the community embedding's posterior distribution, which also enables the higher-order proximity for node embedding as:

$$\begin{aligned} O_3^{ComE+} = -\beta(&\mathbb{E}_q[\log p(W, \boldsymbol{\Phi}|\rho, \nu)] \\ &- \mathbb{E}_q[\log q(W)]), \end{aligned} \tag{8}$$

where $\beta > 0$ is a trade-off parameter. Finally, we define the overall objective function for ComE+ as:

$$\begin{aligned} \mathcal{L}^{ComE+}(\boldsymbol{\Phi}, \boldsymbol{\Phi}', q) = &O_1(\boldsymbol{\Phi}) + O_2(\boldsymbol{\Phi}, \boldsymbol{\Phi}') \\ &+ O_3^{ComE+}(\boldsymbol{\Phi}, q). \end{aligned} \tag{9}$$

IV. Inference

To jointly optimize node embedding and community embedding, we exploit a similar coordinated learning process like the one proposed in ComE. That is, we also apply iterative optimization between $(\boldsymbol{\Phi}, \boldsymbol{\Phi}')$ and q. Given $(\boldsymbol{\Phi}, \boldsymbol{\Phi}')$, optimizing q is to infer the infinite community embedding. Given q, optimizing $(\boldsymbol{\Phi}, \boldsymbol{\Phi}')$ is to learn the node embedding with all types of proximity.

Fix $(\boldsymbol{\Phi}, \boldsymbol{\Phi}')$, **optimize** q. For variational inference, we try to define $q(W)$ with a tractable form. Note that $q(W)$ is a distribution over an infinite set of $(\pi_k, z_k, \psi_k, \Sigma_k)$'s. To make $q(W)$ tractable, we follow [27] to truncate $q(W)$ at a value K, by setting $q(\pi_K = 1) = 1$. According to Eq. 4, we have all the $\eta_k(\boldsymbol{\pi}) = 0$ for $k > K$. In so doing, K only serves as a maximum number of possible communities to be detected in the graph. Due to this truncation, we revise $q(W)$ as $q(W, K)$, where K is a variational parameter. We will empirically evaluate the model performance under different K values in the experiments. Then, we use mean-field approximation to factorize $q(W, K)$ as:

$$\begin{aligned} q(W, K) = &\prod_{k=1}^{K} q_{\gamma_{k,1}, \gamma_{k,2}}(\pi_k) \prod_{k=1}^{K} q_{\tau_k}(\psi_k) \\ &\prod_{k=1}^{K} q_{B_k, c_k}(\Sigma_k^{-1}) \prod_{i=1}^{|V|} q_{\xi_i}(z_i), \end{aligned} \tag{10}$$

where we define $q_{\gamma_{k,1}, \gamma_{k,2}}(\pi_k)$ as a Beta distribution parameterized with $\gamma_{k,1} > 0$ and $\gamma_{k,2} > 0$, $q_{\tau_k}(\psi_k)$ as a multivariate Gaussian distribution parameterized with mean $\boldsymbol{\tau}_k \in \mathbb{R}^d$, $q_{B_k, c_k}(\Sigma_k^{-1})$ as a Wishart distribution parameterized with a positive definite scale matrix $B_k^{-1} \in \mathbb{R}^{d \times d}$ and degrees of freedom $c_k > 0$, and $q_{\xi_i}(z_i)$ as a multinomial distribution parameterized with $\boldsymbol{\xi}_i \in \Delta$ (where Δ is a simplex). With the truncated variational distribution $q(W, K)$, we revise Eq. 8 as:

$$\begin{aligned} O_3^{ComE+'}(\boldsymbol{\Phi}, q) = -\frac{\beta}{K}(&\mathbb{E}_q[\log p(W, \boldsymbol{\Phi}|\rho, \nu)] \\ &- \mathbb{E}_q[\log q(W, K)]). \end{aligned} \tag{11}$$

Ultimately, we replace O_3^{ComE+} in Eq. 9 with $O_3^{ComE+'}$ for inference. Due to space

limit, we skip the derivation details, and summarize the updates for each variational parameter in $q(W, K)$ as[2]:

$$\gamma_{k,1} = 1 + \sum_{i=1}^{|V|} \xi_{i,k}, \tag{12}$$

$$\gamma_{k,2} = \rho + \sum_{i=1}^{|V|} \sum_{j=k+1}^{K} \xi_{i,j}, \tag{13}$$

$$B_k = \left(\sum_{i=1}^{|V|} \xi_{i,k} + 1\right)\mathbf{I} + \sum_{i=1}^{|V|} \xi_{i,k} (\phi_i - \boldsymbol{\tau}_k)(\phi_i - \boldsymbol{\tau}_k)^T, \tag{14}$$

$$\boldsymbol{\tau}_k = \left(\mathbf{I} + B_k^{-1} c_k \sum_{i=1}^{|V|} \xi_{i,k}\right)^{-1} \left(B_k^{-1} c_k \sum_{i=1}^{|V|} \xi_{i,k} \phi_i\right), \tag{15}$$

$$c_k = 2 + \nu + \sum_{i=1}^{|V|} \xi_{i,k}, \tag{16}$$

$$\xi_{i,k} \propto e^{\Lambda(\gamma_{k,1}, \gamma_{k,2}) + \sum_{j=1}^{k} \Lambda(\gamma_{j,1}, \gamma_{j,2}) + \mathbb{E}_q[\log p(\phi_i | z_i = k)]}, \tag{17}$$

where we define $\Lambda(\gamma_{l,1}, \gamma_{l,2}) = \Psi(\gamma_{l,1}) - \Psi(\gamma_{l,1} + \gamma_{l,2})$, given $\Psi(\cdot)$ as a digamma function.

Fix q, optimize (Φ, Φ'). We use MAP estimation to learn the node embedding, given O_1 and O_2 as the likelihood terms and $O_3^{ComE+'}$ as the prior term for ϕ_i's. Optimizing O_1 and O_2 is straightforward and is similar to ComE's inference. Specifically, for each $v_i \in V$, we have:

$$\frac{\partial O_1}{\partial \phi_i} = -\sum_{(i,j) \in E} \sigma(-\phi_j^T \phi_i)\, \phi_j, \tag{18}$$

$$\frac{\partial O_2}{\partial \phi_i} = -\alpha \sum_{v_j \in C_i} \left[\sigma(-\phi_j'^T \phi_i)\, \phi_j' + \sum_{t=1}^{m} \mathbb{E}_{v_l \sim P_n(v_l)} [\sigma(\phi_l'^T \phi_i)(-\phi_l')]\right]. \tag{19}$$

We also compute the gradient for context embedding as:

$$\frac{\partial O_2}{\partial \phi_j'} = -\alpha \sum_{v_i \in V} \left[\delta(v_j \in C_i)\, \sigma(-\phi_j'^T \phi_i)\, \phi_i + \sum_{t=1}^{m} \mathbb{E}_{v_l \sim P_n(v_l)} [\delta(v_l = v_j)\, \sigma(\phi_l'^T \phi_i)(-\phi_i)]\right]. \tag{20}$$

Instead, optimizing $O_3^{ComE+'}$ requires some more simplification. In Eq. 11, only the last term depends on ϕ_i's; thus we can simplify $O_3^{ComE+'}$ as:

[2] The full derivation can be found at the following link: http://sentic.net/derivation.pdf

Algorithm 1 Inference algorithm for ComE+.

Input: graph $G = (V, E)$, variational parameter k, #(paths per node) γ, walk length ℓ, context size ζ, embedding dimension d, negative context size m, parameters (α, β).
Output: node embedding Φ, context embedding Φ', community embedding posterior distribution q

1 $\mathcal{P} \leftarrow$ SamplePath(G, ℓ);
2 Initialize Φ and Φ' by DeepWalk [3] with $\mathcal{P}$;
3 **for** *iter* $= 1 : T_1$ **do**
4 **for** *subiter* $= 1 : T_2$ **do**
5 Optimize $q(W, K)$ by Eq. 12–17 given Φ;
6 **end**
7 **foreach** *edge* $(i, j) \in E$ **do**
8 SGD on ϕ_i and ϕ_j by Eq. 18;
9 **end**
10 **foreach** *path* $p \in \mathcal{P}$ **do**
11 **foreach** v_i *in path* p **do**
12 SGD on ϕ_i by Eq. 19;
13 SGD on ϕ_j''s within ζ hops by Eq. 20;
14 **end**
15 **end**
16 **foreach** *node* $v_i \in V$ **do**
17 SGD on ϕ_i by Eq. 21;
18 **end**
19 **end**

$$O_3^{ComE+''}(\Phi) = -\frac{\beta}{K} \sum_{i=1}^{|V|} \mathbb{E}_q[\log p(\phi_i | \psi_{z_i}, \Sigma_{z_i})] \propto \frac{\beta}{K} \sum_{i=1}^{|V|} \sum_{k=1}^{K} \xi_{i,k} \cdot \frac{1}{2} (\phi_i - \tau_k)^T (B_k^{-1} c_k)(\phi_i - \tau_k) \tag{21}$$

and compute the gradient over ϕ_i as:

$$\frac{\partial O_3^{ComE+''}}{\partial \phi_i} = \frac{\beta}{K} \sum_{k=1}^{K} \xi_{i,k} (B_k^{-1} c_k)(\phi_i - \tau_k). \tag{22}$$

Finally, we have the total gradient for each ϕ_i defined as:

$$\frac{\partial \mathcal{L}'}{\partial \phi_i} = \frac{\partial O_1}{\partial \phi_i} + \frac{\partial O_2}{\partial \phi_i} + \frac{\partial O_3^{ComE+''}}{\partial \phi_i}. \tag{23}$$

Algorithm and complexity. We summarize the inference algorithm of ComE+ in Alg. 1. In line 1, for each $v_i \in V$, we sample γ paths starting from v_i with length ℓ on G. In line 2, we initialize (Φ, Φ') by DeepWalk.

In lines 4–5, we fix (Φ, Φ') and optimize $q(W, K)$ for community detection and embedding. In lines 6–11, we fix $q(W, K)$ and optimize (Φ, Φ') for node embedding. We analyze the complexity of Alg. 1.

Path sampling in line 1 takes $O(|V|\gamma\ell)$. Parameter initialization by DeepWalk in line 2 takes $O(|V|)$. Optimizing the variational parameters $q(W, K)$ in line 5 takes $O(|V|K)$. Node embedding w.r.t. first-order proximity in lines 7 takes $O(|E|)$. Node embedding w.r.t. second-order proximity in lines 9–11 takes $O(|V|\gamma\ell)$. Node embedding w.r.t. community-aware high-order proximity in lines 12–13 takes $O(|V|K)$. In total, the complexity of Alg. 1 is $O(|V|\gamma\ell + |V| + T_1 \times (T_2|V|K + K + |E| + |V|\gamma\ell + |V|K))$, which is still linear to the graph size (i.e., $|V|$ and $|E|$).

V. Experiments

In this section, we evaluate ComE+ on six real-world datasets under two application tasks, including node classification and community detection[3]. Compared

[3] Due to space limitation: 1) the graph visualization performance, 2) the ComE clustering results, 3) the complete set of node classification experiments and 4) impact of some parameters like α, β and the embedding size are reported at the following link: http://sentic.net/node-classification.pdf

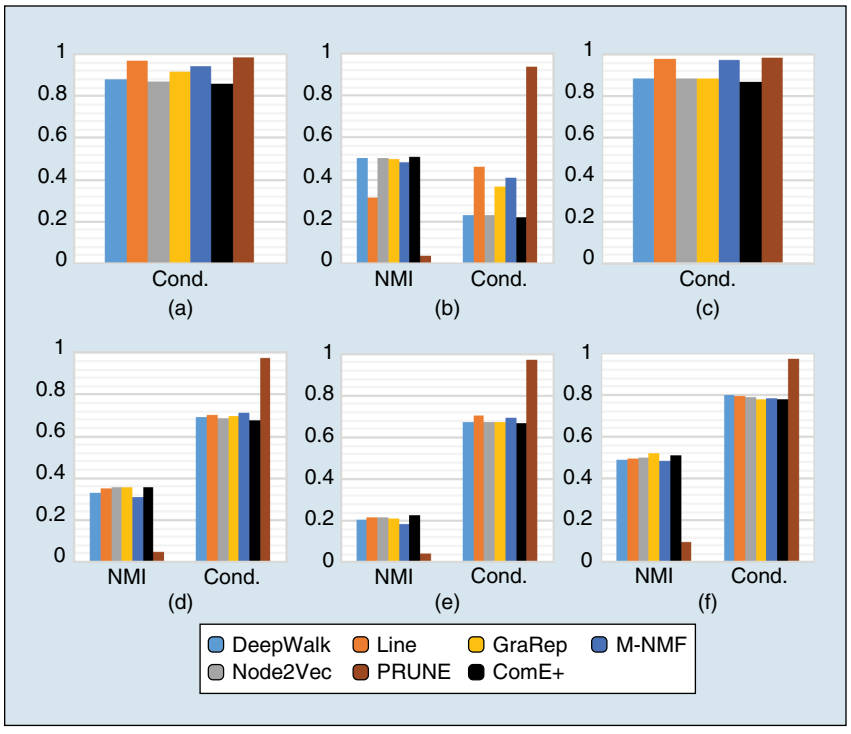


FIGURE 2 ComE+ community detection results. The smaller conductance/bigger NMI, the better. (a) BlogCatalog, (b) DBLP, (c) Wikipedia, (d) Rochester, (e) Mich and (f) Amherst.

TABLE II DBLP dataset labels.

CONFERENCE	LABEL
EMNLP, ACL, CONLL, COLING	NLP
CVPR, ICCV, ICIP, SIGGRAPH	COMPUTER VISION
KDD, ICDM, CIKM, WSDM	DATA MINING
SIGMOD, VLDB/PVLDB, ICDE	DATABASE
INFOCOM, SIGCOM, MobiHoc, MobiCom	NETWORKING

with our previous work [22], we introduce three more benchmark datasets for evaluation as well as a sensitivity analysis of the hyperparameters present in ComE and ComE+.

Datasets. To evaluate our models, we decided to use different types of graphs, ranging from social networks, word co-occurrence network and academic paper citation networks. Following there is a short description of each graph:

- BlogCatalog is a social network for blogger. Each node represents a user, and each edge represents a friendship connection. It has to be noted that this dataset present multiple labels for each node.
- Wikipedia is a co-occurrence network of words appearing in the first million bytes of English Wikipedia dump on Mar. 3, 2006. Each word has its Part-of-Speech annotations as multi-labels.
- DBLP is an academic paper citation network built upon the DBLP repository. Formally, we selected 19 of the major conferences on five computer science research areas (i.e., the entire list is reported in Tab. II). Each node is a paper, and each edge represents a citation that the current paper has to others selected articles. The label of the node indicates its conference venue (i.e., one of the five areas).
- Rochester, Mich and Amherst are three Facebook networks formed by students from the same universities. Each node is a student, while each edge is a friendship connection. The matriculation year is used as node's label. Note that we removed all the isolated and unlabeled nodes for evaluations.

Evaluation metrics. In literature, there are multiple possible evaluation metric for community detection and node classification. However, we will use Conductance [48] and Normalized Mutual Information (NMI) [6] to evaluate the community detection performance. Conductance is a ratio between the number of edges leaving a community and that within it. NMI measures the closeness between predicted community labels with ground truth node labels, but it is formally defined only for single label datasets. Instead, to evaluate node classification, we use Micro-F1 and Macro-F1 [3]. Micro-F1 is the overall F1 w.r.t. all labels. Macro-F1 is the average of F1 score w.r.t. each individual label.

Baselines. We compare the proposed models with the following baselines, using their author-published codes whenever possible, on all the datasets.

- DeepWalk [3]: as state in Sec. II, it only models second-order proximity.
- LINE [16]: it extend the DeepWalk model to consider both first- and second-order proximity.
- Node2Vec [13]: exploit a guided path sampling strategy to better exploits homophily and structural roles in embedding.
- GraRep [2]: a model able to capture the higher-order proximity in random walk.
- M-NMF [15]: it jointly models nodes and communities by non-negative matrix factorization.
- PRUNE [20]: it models first-order and global ranking by siamese neural networks.

Parameters and environment. All the tested models need a specific embedding dimension value: thus we set $d = 128$. DeepWalk, Node2Vec, ComE and ComE+ have also the additional parameters γ, ζ, ℓ and m. For consistency, we followed the results obtained in [3], [13] setting $\gamma = \zeta = 10$, $\ell = 80$ and $m = 5$.

Node2vec has two more parameters p and q. We set their values with the best performance on BlogCatalog (i.e., $p = 0.25$ and $q = 0.25$) and Wikipedia ($p = 4$ and $q = 1$) as reported in [13] while for the other four datasets, we followed the same parameter tuning

procedure, and we found that: on DBLP, Rochester and Mich, $q = 0.25$ and $p = 0.25$ work at best; while on Amherst, $p = 0.25$ and $q = 1$ is the outstanding setting.

M-NMF has three more parameters α, β and K. We experimentally tuned α and β in the range [0.1, 1, 5, 10] while keeping the other parameter fixed. The final setting is: $\alpha = 0.1$ and $\beta = 5$ for BlogCatalog and Wikipedia; $\alpha = 1$ and $\beta = 5$ for Rochester, Mich and Amherst; while for DBLP we used $\alpha = 10$ and $\beta = 5$. Finally, we set K as the number of labels present in the dataset.

Compared to the others methods, PRUNE uses as input only the edges of a network. To make a fair comparison, we set the number of training epochs so to obtain a comparable amount of training instances w.r.t. the others models.

As for M-NMF, also ComE and ComE+ have three parameters to tune. The impact of K is evaluated in Sec. V-C, but if not otherwise stated we used $\alpha = 0.1$ and $\beta = 0.1$ for all the ComE experiments. While, for ComE+, we used $\alpha = 0.1$ and $\beta = 0.1$ on BlogCatalog, Wikipedia and karate Club, but $\alpha = 0.1$ and $\beta = 1$ is used for the remaining datasets. Finally, in Sec. V-A and V-B $K = 50$ is used as an upper bound on all but the DBLP dataset, where $K = 20$ is used for the ComE+ experiments. Instead, the ComE's results are obtained setting K as the number of unique labels.

A. Community Detection

Community detection is known to be an unsupervised learning task aiming at predicting the most likely community assignment for each node. We employed the following steps for evaluation: firstly, we learned a node embedding for the entire graph; then, we applied a clustering algorithm to derive the associated communities for each node. However, since it does not exist a commonly accepted method to compute the NMI on multi-label datasets, for BlogCatalog and Wikipedia we only report the Conductance w.r.t. the first label. Observe that we do not compare ComE+ w.r.t. ComE, because it assume different prior knowledge. To this end, we only report the comparison between ComE+ and the remaining baselines. Finally, each reported result is the average of 10 independent evaluations with different initial centroids.

In general, ComE+ can outperform all the baselines in terms of Conductance by 0.75% to 5.17%. Regarding NMI, it can outperform all the baselines on DBLP (0.6%) and Mich (4.7%). However, it obtains comparable performance w.r.t. the best performing baseline on Rochester dataset (–0.28%); but GraRep significantly outperforms it on Amherst (–3.7%). This could be related to GraRep's ability to build a higher order transition probability matrix over dense graphs, which avoids the limitations associated with the path-sampling procedure. Moreover, an in-deep analysis of the datasets shows that Rochester and Mich present communities formed by a single node which is a challenging scenario to model using a probabilistic approach. Overall, the NMI improvements are relatively smaller than the one obtained on Conductance. This suggests that node embedding methods tend to create multiple sub-clusters for the same community. Hence it can create small homogeneous clusters (i.e., for Conductance), but it becomes less accurate when inferring the value of K (i.e., for NMI). Finally, the robust performance of ComE+ suggests that modeling community detection together with node embedding is better than solving them separately.

B. Node Classification

In node classification, the goal is to categorize each node into one or more classes, depending on whether it is a single-label or multi-label setting. We follow [3] to first train graph embedding on the whole graph. Then, we randomly split 80% of the nodes as a training set and the remaining 20% is used for testing. Finally, an SVM classifier [49] is used to infer the node labels. So far, we report the results with the SVM classifier parameter $c = 1$, which was suggested by [3], for all the methods. We observed a similar trend with $c \in [0.1, 10]$, and due to space limitations, we skip the results with different c values. Note that, the reported results are the average of 10 experiments executed with different random sampling. As over-state, for ComE we set $\alpha = 0.1$, $\beta = 0.1$ and K as the number of labels present in each dataset. Instead, the settings for ComE+ are: $\alpha = 0.1$ and $\beta = 0.1$ for BlogCatalog and Wikipedia; while the remaining datasets perform better with $\alpha = 0.1$ and $\beta = 1$.

From Tab. III it is possible to do some interesting observations. At first, our ComE and ComE+ are generally better than all the baselines concerning both Macro-F1 and Micro-F1. In particular, ComE+ can improve the baselines by $0.8\% - 25.4\%$ on Macro-F1 and $0.3\% - 34.7\%$ on Micro-F1. Secondly, ComE, and ComE+ have comparable performance on DBLP, BlogCatalog, Mich and Wikipedia. On Rochester, ComE+ can outperform ComE in terms of Macro-F1, but they are comparable regarding Micro-F1. Instead, on Amherst ComE+ is outperforming ComE only according the Micro-F1 metric. The similar performance between ComE+ and ComE on DBLP dataset could be related to the presence of a stronger community structure. However, ComE+ can better model multi-label datasets, indicating that it can solve the multiple membership problems better or detect a more meaningful community structure [24] thanks to its variational inference algorithm. Third, note that, on Rochester and Mich, also GraRep or M-NMF present strong performance. Fourth, on Amherst, the random walk based methods present poor performance w.r.t. the matrix factorization ones. We suggest that the high average degree of the dataset, which inherently generates a denser adjacency matrix, could positively affect the factorization methods. Instead, the fixed number of paths sampled by the random walk methods cannot take advantage from such situation. DBLP, having a low average degree, also supports this hypothesis since Node2Vec appears to be the best baseline while matrix factorization methods suffer from the sparsity of the affinity matrix.

Nevertheless, ComE, and ComE+ can partially overcome this limitation because they leverage community information, validating the importance of jointly perform node and community embedding. Fifth, on average, among all the baseline, GraRep appears to be the best performing methods. This suggests that the higher-order transition probability matrix contains valuable information, but comes at the price of an overall higher complexity ($O(|V|^3)$). Whereas, on the university Facebook data sets, M-NMF presents results comparable with the GraRep models, validating the importance of modeling communities in a network structure. Overall, the results validate the existence of a closed loop between mesoscopic communities structure and the microscopic nodes structure. That is, community information is not only useful for community-related tasks but also node related tasks.

C. Parameter sensitivity

In this section we are going to explore the performance of the ComE+ algorithm for different parameter setting. Also, we focus on the comparison between ComE+ and ComE to evaluate better the variational inference approach used for community detection and embedding.

1) Impact of K

As the main difference between ComE+ and ComE is the variational inference process, we compare the impact of parameter K for both algorithms. Let us denote K' as the real number of communities present in datasets. Then, we generate different embedding with K equal to the values $[3, K', 10, 20]$ for DBLP and $[5, 10, K', 50, 100]$ for all the other datasets. This enable us to evaluate the robustness of the proposed approach w.r.t. the uncertainty in the number of communities.

From Fig. 3 we make the following observations. First, on DBLP, neither ComE nor ComE+ is sensitive to the setting of K regarding node classification, but both of them are sensitive to K concerning community detection. Such behavior indicates that the SVM can exploit the RBF kernel to separate the nodes, while the generated node embedding misleads a traditional clustering algorithm. On the one hand, the performance of ComE reaches the top when $K = K'$, while they reduce by a relative -140% to -189% in terms of Conductance and -17.6% to -25.6% w.r.t. NMI. On the other hand, as expected, ComE+ is more robust when $K \geq K'$. That is, the ComE+ performance only varies between -1.4% to $+0.2\%$ for the NMI metric and -26.1% to -2.3% according to Conductance. Secondly, the models usually perform the best when $K = K'$ in most cases, although on rare occasions (e.g., on Amherst) ComE+ has better performance when $K > K'$. Although the improvement is not significant, this suggests that the communities present in the data distribution do not adequately reflect the number of labels. Alternatively, it expresses the tendency of node embedding methods to create small sets of homogeneous nodes; thus,

TABLE III Node classification results. Note that all the experiments are conduced with 80% of the total nodes as training set while the remaining 20% is used for evaluation.

	BLOGCATALOG		WIKIPEDIA		DBLP	
	MACRO-F1(%)	**MICRO-F1(%)**	**MACRO-F1(%)**	**MICRO-F1(%)**	**MACRO-F1(%)**	**MICRO-F1(%)**
COME	26.5 ($P = 0.12$)	41.7 ($P = 0.06$)	9.8 ($P = 0.15$)	44.0 ($P = 0.13$)	92.2 ($P = 0.61$)	92.6 ($P = 0.47$)
COME+	**27.1**	**42.5**	**10.2**	**45.4**	**92.2**	**92.6**
DEEPWALK	22.2 ($P < 0.01$)	38.3 ($P < 0.01$)	4.6 ($P < 0.01$)	28.0 ($P < 0.01$)	91.2 ($P < 0.01$)	91.6 ($P < 0.01$)
LINE	10.9 ($P < 0.01$)	30.2 ($P < 0.01$)	5.3 ($P < 0.01$)	30.5 ($P < 0.01$)	90.4 ($P < 0.01$)	91.0 ($P < 0.01$)
NODE2VEC	24.1 ($P < 0.01$)	39.9 ($P < 0.01$)	6.1 ($P < 0.01$)	31.1 ($P < 0.01$)	91.5 ($P < 0.01$)	92.0 ($P < 0.01$)
GRAREP	23.6 ($P < 0.01$)	40.9 ($P < 0.01$)	8.1 ($P = 0.09$)	33.4 ($P < 0.01$)	90.6 ($P < 0.01$)	91.1 ($P < 0.01$)
M-NMF	15.7 ($P < 0.01$)	33.8 ($P < 0.01$)	7.0 ($P < 0.01$)	33.7 ($P < 0.01$)	89.6 ($P < 0.01$)	90.3 ($P < 0.01$)
PRUNE	4.6 ($P < 0.01$)	15.6 ($P < 0.01$)	4.9 ($P < 0.01$)	35.0 ($P < 0.01$)	22.4 ($P < 0.01$)	38.4 ($P < 0.01$)
	ROCHESTER		**MICH**		**AMHERST**	
	MACRO-F1(%)	**MICRO-F1(%)**	**MACRO-F1(%)**	**MICRO-F1(%)**	**MACRO-F1(%)**	**MICRO-F1(%)**
COME	49.7 ($P < 0.05$)	86.6 ($P = 0.42$)	36.9 ($P = 0.61$)	63.2 ($P = 0.21$)	65.7 ($P = 0.5$)	91.1 ($P = 0.05$)
COME+	**53.7**	**86.8**	**37.3**	**64.1**	**66.6**	**91.6**
DEEPWALK	44.1 ($P < 0.01$)	82.9 ($P < 0.01$)	33.2 ($P < 0.01$)	60.9 ($P < 0.01$)	57.6 ($P < 0.01$)	88.5 ($P < 0.01$)
LINE	47.4 ($P < 0.01$)	85.4 ($P < 0.05$)	34.1 ($P < 0.01$)	61.5 ($P < 0.01$)	59.5 ($P < 0.01$)	88.9 ($P < 0.01$)
NODE2VEC	46.6 ($P < 0.01$)	82.6 ($P < 0.01$)	34.4 ($P < 0.05$)	61.6 ($P < 0.01$)	57.6 ($P < 0.01$)	89.4 ($P < 0.01$)
GRAREP	48.8 ($P < 0.01$)	86.5 ($P < 0.01$)	35.4 ($P = 0.29$)	63.0 ($P = 0.13$)	62.9 ($P = 0.07$)	91.0 ($P = 0.08$)
M-NMF	48.3 ($P < 0.01$)	86.4 ($P = 0.23$)	34.3 ($P < 0.05$)	61.6 ($P < 0.05$)	60.0 ($P < 0.05$)	90.8 ($P < 0.05$)
PRUNE	13.2 ($P < 0.01$)	29.1 ($P < 0.01$)	11.6 ($P < 0.01$)	23.6 ($P < 0.01$)	12.7 ($P < 0.01$)	27.3 ($P < 0.01$)

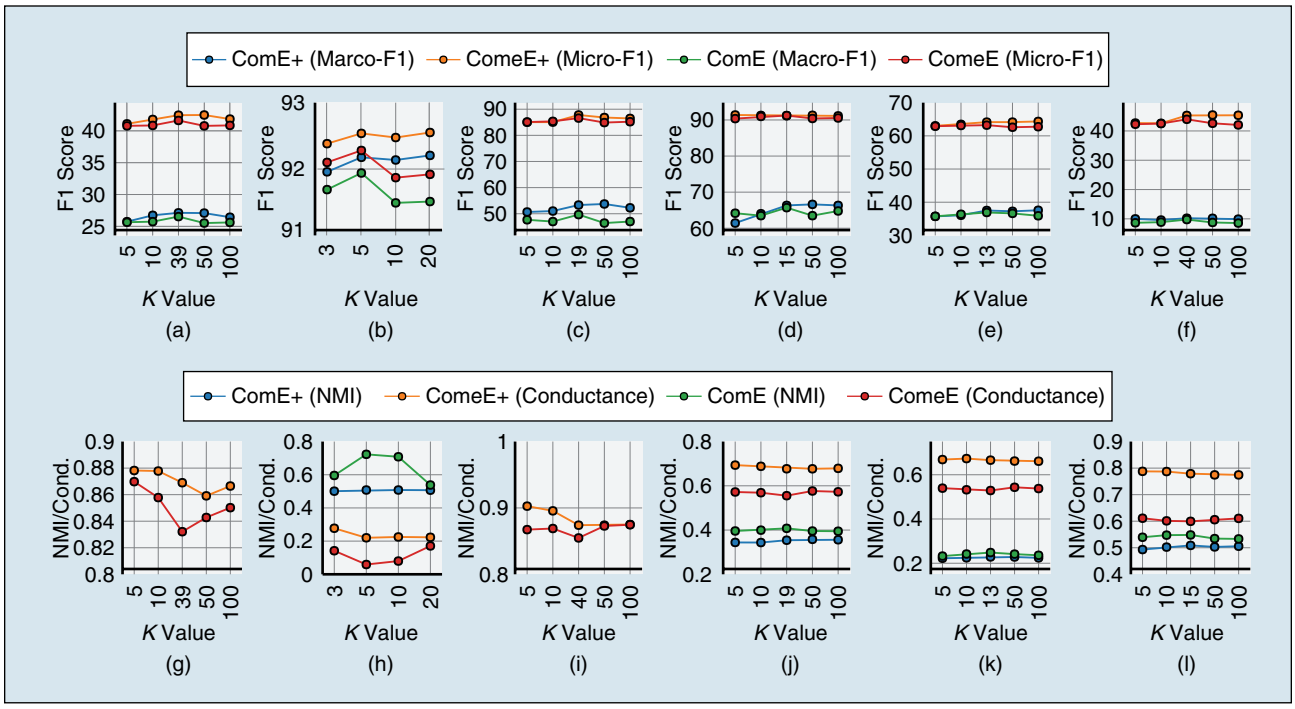


FIGURE 3 Impact of parameter K. The smaller variance of the performance when $K \geq K'$ suggests that ComE+ is more stable with respect to ComE. (a) BlogCatalog, (b) DBLP, (c) Wikipedia, (d) Rochester, (e) Mich, (f) Amherst, (g) BlogCatalog, (h) DBLP, (i) Wikipedia, (j) Rochester, (k) Mich and (l) Amherst.

over-segmentation appears to be useful in modeling this behavior. Finally, as expected, the performance of ComE+ are more robust than ComE concerning both community detection and node classification especially when $K \geq K'$, validating the Bayesian inference process used to handle the uncertainty in the number of the community.

D. Convergence and efficiency

As final experiments, we compare the convergence and efficiency of both models. We record the value of the loss functions at the end of every iteration. As shown in Fig. 4, the loss of both ComE and ComE+ converge quickly within 2–3 iterations.

To demonstrate the efficiency of our models, we test them on all the six datasets at different scales. More precisely, for each dataset, we generate four subgraphs in which we keep 25%, 50%, 75% and 100% of the total number of nodes and edges. It has to be noted that, to speed up the computation time of those experiments, we set $d = 2$ and $\zeta = 5$. The diagram in Fig. 5 shows the processing time of ComE and ComE+ in

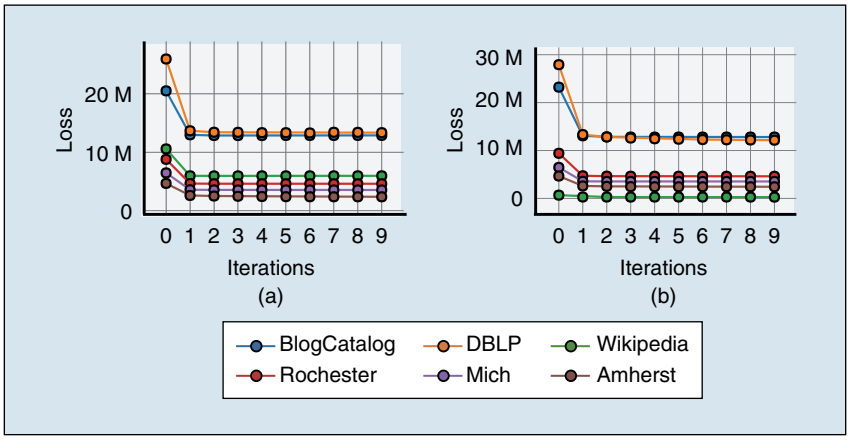


FIGURE 4 Model convergence. (a) ComE+ and (b) ComE.

different datasets. Clearly, the processing time of our algorithms is linear to the graph size (i.e., $|V|$ and $|E|$). This validates our complexity analysis at the end of Sec. IV.

VI. Conclusion

In this paper, we studied the important (yet largely under-explored) problem of embedding communities on graphs. We have investigated the existence of a closed loop among community embedding, community detection and node embedding that preserve a community-aware higher-order proximity. More in detail, we extend the ComE algorithm to achieve such closed loop in a Bayesian inference setting. The proposed ComE+ algorithm can better handle the uncertainty related to the unknown number of communities. We also designed an efficient iterative inference algorithm for ComE+, which can still retain a low complexity of $O(|V|+|E|)$. We evaluated our model on seven real-world datasets and with multiple application tasks.

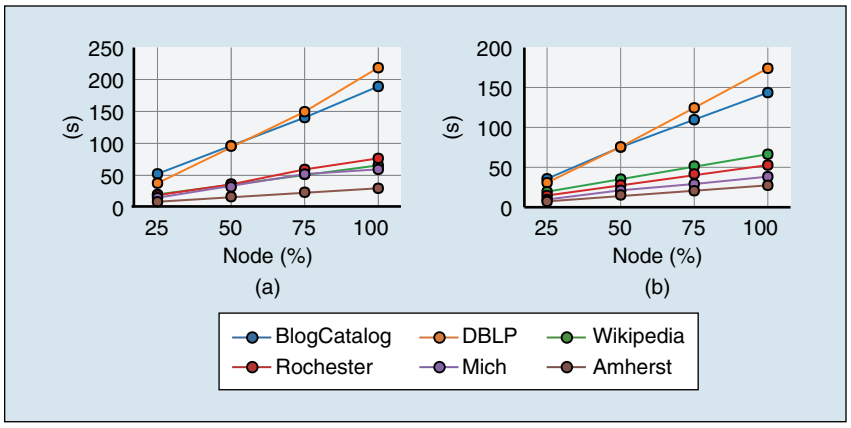


FIGURE 5 Model efficiency. Note that the average time per iteration is reported. (a) ComE+ and (b) ComE.

We showed that our models outperform the state-of-the-art baselines across the datasets by at least 0.8%–6.7% (NMI) and 0.3%–17.1% (Conductance) in community detection, 0.8%–30.3% (Macro-F1) and 0.3%–48% (Micro-F1) in node classification. Finally, we studied the parameter sensitivity, model convergence and model efficiency. Our models can converge quickly, and scale well w.r.t. the graph size $|V|+|E|$.

In the future, we are interested in exploring the graphs with additional content or attribute information. Additionally, we are also interested in exploring community embedding in a dynamic graph setting, which is useful for many applications such as biology and finance.

References

[1] H. Cai, V. W. Zheng, and K. C. Chang, "A comprehensive survey of graph embedding: Problems, techniques and applications," *CoRR*, vol. abs/1709.07604, 2017.

[2] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proc. CIKM*, 2015, pp. 891–900.

[3] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. KDD*, 2014, pp. 701–710.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, 2017.

[5] A. Garcia Duran and M. Niepert, "Learning graph representations with embedding propagation," in *Proc. NIPS*, 2017, pp. 5125–5136.

[6] F. Tian, B. Gao, Q. Cui, E. Chen, and T. Liu, "Learning deep representations for graph clustering," in *Proc. AAAI*, 2014, pp. 1293–1299.

[7] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, "Modularity based community detection with deep learning," in *Proc. IJCAI*, 2016, pp. 2252–2258.

[8] C. Zhou, Y. Liu, X. Liu, Z. Liu, and J. Gao, "Scalable graph embedding for asymmetric proximity," in *Proc. AAAI*, 2017, pp. 2942–2948.

[9] Q. Zhang and H. Wang, "Not all links are created equal: An adaptive embedding approach for social personalized ranking," in *Proc. 39th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, ACM, 2016, pp. 917–920.

[10] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. New York: Springer-Verlag, 2006.

[11] W. W. Zachary, "An information flow model for conflict and fission in small groups," *J. Anthropological Res.*, vol. 33, no. 4, pp. 452–473, 1977.

[12] L. Tang and H. Liu, "Leveraging social media networks for classification," *Data Min. Knowl. Discov.*, vol. 23, no. 3, pp. 447–478, 2011.

[13] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. KDD*, 2016.

[14] M. Kozdoba and S. Mannor, "Community detection via measure space embedding," in *Proc. NIPS*, 2015, pp. 2890–2898.

[15] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. AAAI*, 2017, pp. 203–209.

[16] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. WWW*, 2015, pp. 1067–1077.

[17] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. KDD*, 2016, pp. 1225–1234.

[18] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "Struc2vec: Learning node representations from structural identity," in *Proc. KDD*, 2017, pp. 385–394.

[19] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proc. KDD*, 2017, pp. 135–144.

[20] Y.-A. Lai, C.-C. Hsu, W. H. Chen, M.-Y. Yeh, and S.-D. Lin, "Prune: Preserving proximity and global ranking for network embedding," in *Proc. NIPS*, 2017, pp. 5263–5272.

[21] M. S. Handcock, A. E. Raftery, and J. M. Tantrum, "Model-based clustering for social networks," *J. Roy. Stat. Soc. A (Stat. Soc.)*, vol. 170, no. 2, pp. 301–354, 2007.

[22] S. Cavallari, V. W. Zheng, H. Cai, K. C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *Proc. CIKM*, 2017, pp. 377–386.

[23] H. Zhang, C. L. Giles, H. C. Foley, and J. Yen, "Probabilistic community discovery using hierarchical latent Gaussian mixture model," in *Proc. AAAI*, 2007, vol. 7, pp. 663–668.

[24] L. Peel, D. B. Larremore, and A. Clauset, "The ground truth about metadata and community detection in networks," *Sci. Adv.*, vol. 3, no. 5, p. e1602548, 2017.

[25] N. Veldt, D. F. Gleich, and A. Wirth, "A correlation clustering framework for community detection," in *Proc. 2018 World Wide Web Conf. World Wide Web and Proc. Int. World Wide Web Conf. Steering Committee*, 2018, pp. 439–448.

[26] C. E. Rasmussen, "The infinite Gaussian mixture model," in *Proc. NIPS*, 2000, pp. 554–560.

[27] D. M. Blei and M. I. Jordan, "Variational inference for Dirichlet process mixtures," *Bayesian Anal.*, vol. 1, no. 1, pp. 121–144, 2006.

[28] G. Nikolentzos, P. Meladianos, and M. Vazirgiannis, "Matching node embeddings for graph similarity," in *Proc. AAAI*, 2017, pp. 2429–2435.

[29] F. Nie, W. Zhu, and X. Li, "Unsupervised large graph embedding," in *Proc. AAAI*, 2017, pp. 2422–2428.

[30] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NIPS*, 2017.

[31] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. NIPS*, 2001, pp. 585–591.

[32] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[33] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proc. KDD*, 2016, pp. 1105–1114.

[34] M. Wang, C. Wang, J. X. Yu, and J. Zhang, "Community detection in social networks: An in-depth benchmarking study with a procedure-oriented framework," *PVLDB*, vol. 8, no. 10, pp. 998–1009, June 2015.

[35] M. Shao, S. Li, Z. Ding, and Y. Fu, "Deep linear coding for fast graph clustering," in *Proc. IJCAI*, 2015, pp. 3798–3804.

[36] X. He, D. Cai, Y. Shao, H. Bao, and J. Han, "Laplacian regularized Gaussian mixture model for data clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 9, pp. 1406–1418, 2011.

[37] M. Sachan, A. Dubey, S. Srivastava, E. P. Xing, and E. Hovy, "Spatial compactness meets topical consistency: Jointly modeling links and content for community detection," in *Proc. WSDM*, 2014, pp. 503–512.

[38] Y. Sun, C. C. Aggarwal, and J. Han, "Relation strength-aware clustering of heterogeneous information networks with incomplete attributes," *PVLDB*, vol. 5, no. 5, pp. 394–405, Jan. 2012.

[39] H. Cai, V. W. Zheng, F. Zhu, K. C. Chang, and Z. Huang, "From community detection to community profiling," *PVLDB*, vol. 10, no. 7, pp. 817–828, 2017.

[40] L. He, C.-T. Lu, J. Ma, J. Cao, L. Shen, and P. S. Yu, "Joint community and structural hole spanner detection via harmonic modularity," in *Proc. KDD*, 2016.

[41] Y. Han and J. Tang, "Probabilistic community and role model for social networks," in *Proc. KDD*, 2015, pp. 407–416.

[42] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. NIPS*, 2013, pp. 2787–2795.

[43] Z. Yang, J. Tang, and W. W. Cohen, "Multi-modal Bayesian embeddings for learning social knowledge graphs," in *Proc. IJCAI*, 2016, pp. 2287–2293.

[44] A. E. Raftery, X. Niu, P. D. Hoff, and K. Y. Yeung, "Fast inference for the latent space network model using a case-control approximate likelihood," *J. Comput. Graph. Stat.*, vol. 21, no. 4, pp. 901–919, 2012.

[45] S. Suwan, D. S. Lee, R. Tang, D. L. Sussman, M. Tang, and C. E. Priebe, "Empirical Bayes estimation for the stochastic blockmodel," *Electron. J. Stat.*, vol. 10, no. 1, pp. 761–782, 2016.

[46] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. NIPS*, 2013, pp. 3111–3119.

[47] H. Xiao, M. Huang, and X. Zhu, "TransG: A generative model for knowledge graph embedding," in *Proc. ACL*, 2016, vol. 1, pp. 2316–2325.

[48] K. Kloster and D. F. Gleich, "Heat kernel based community detection," in *Proc. KDD*, 2014, pp. 1386–1395.

[49] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011.