Expert Systems With Applications 224 (2023) 119994

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Graph neural networks via contrast between separation and aggregation for self and neighborhood

Xiaoyu Xu [a,b], Xiaoyu Shi [a,b], Mingsheng Shang [a,b,*]

[a] Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China

[b] Chongqing School, University of Chinese Academy of Sciences (UCAS Chongqing), Chongqing, China

ARTICLE INFO

Keywords:
Graph neural networks
Graph representation learning
Contrastive learning

ABSTRACT

Graph Neural Networks (GNNs) including Graph Convolutional Networks (GCNs) have demonstrated state-of-the-art performance in various analytical tasks. Current GNNs approaches focus on learning node representations based on the proximity similarity principle. However, they may fail to generalize to complex relations such as disassortative structure, because irrelevant information may be aggregated and generate unreasonable representations. To this end, this research introduces a novel approach – Graph neural networks via contrast between separation and aggregation for self and neighborhood, to effectively learn the node representations in the disassortative structure. To search friendly neighbor space with mitigating the unreasonable of connections, the proposed approach synthesizes the feature semantic space and the structure semantic space. To learn reasonable representations for disassortative graphs, the proposed approach stacks multiple GCNs and learns the intrinsic interrelationship between the optimal friendly aggregated information and the separated information originating from self and neighborhood by contrastive learning. At the same time, the proposed approach integrates the variational expectation maximization to train the whole framework for exploring the approximated posterior. Extensive experimental results on eight real-world graph datasets show that the proposed approach outperforms eight state-of-the-art GNN models in diverse graph mining tasks, including node classification and link prediction, demonstrating the proposed approach's superior graph representation ability.

1. Introduction

Network-based explorations are ubiquitous in numerous real-world applications which can be tackled to the crucial problems in diverse domains, such as social network analysis (Dornaika, 2022; Zhang, Yin, Zhu, & Zhang, 2020), fraud detection (Zhang et al., 2020), recommendation (Wu et al., 2021), and disease network analysis (Cai, Zheng, & Chang, 2018). Graph Neural Networks (GNNs) are arguably the most popular approaches to discovering and learning effective feature representations for the graph nodes, and have been demonstrated remarkable success in tackling graph learning tasks, such as node classification (Goyal & Ferrara, 2018; Zhang et al., 2020), link prediction (Scarselli, Gori, Tsoi, Hagenbuchner, & Monfardini, 2009; Wang, Liang, Cui, & Liang, 2021), etc.

Current GNN approaches typically use a recursive message propagating scheme to aggregate the local neighborhood information of each graph node to learn discriminative patterns in a semantic-rich, non-Euclidean space (Zhang et al., 2020). They focus on learning representations under the assortative assumption that nodes prefer to connect with similar nodes. This is also consistent with the actual situation that many networks show strong assortativity like Fig. 1(a) which is an academic research network existing two research teams of basic subjects, like Law and Physics. Nevertheless, not all neighbors are similar in the real world, and some connect nodes with different labels. It is also widespread that networks have disassortative structures such as the toy example graph G in Fig. 1(b) which can occur in the relationships among the highly synthetic interdisciplinary composed of many different - fields, like computer science and Bioinformatics. After aggregation with noise (see Fig. 1 (c) for a basic propagation process of GNNs on a toy example graph G in Fig. 1 (b)), the node embedding space of G learned by these GNNs is as Fig. 1(e) which shows that neighbor nodes are often pulled closer, otherwise pushed far away from each other. This is

* Corresponding author at: Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China.

E-mail addresses: xuxiaoyu18@mails.ucas.ac.cn (X. Xu), xiaoyushi@cigit.ac.cn (X. Shi), msshang@cigit.ac.cn (M. Shang).

https://doi.org/10.1016/j.eswa.2023.119994
Received 18 July 2022; Received in revised form 16 March 2023; Accepted 27 March 2023
Available online 31 March 2023

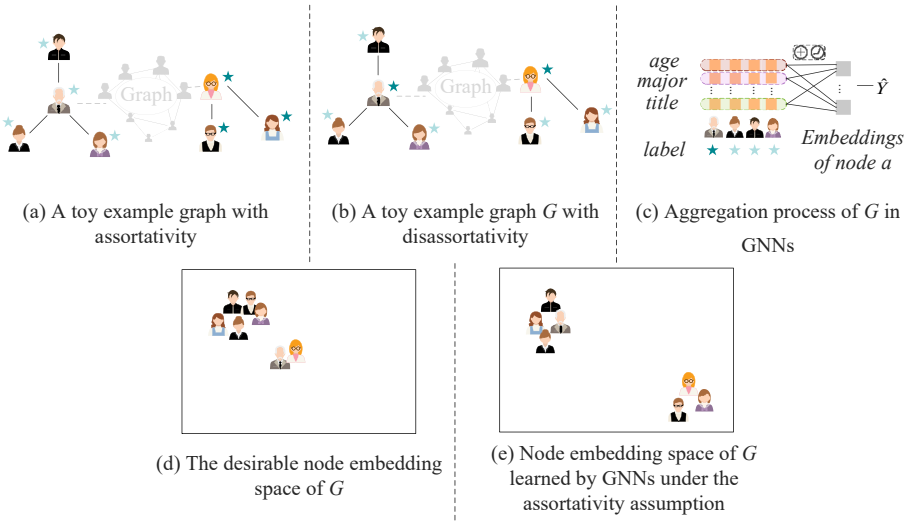
inconsistent with the actual situation, since the distance between neighbors with different labels should be far from each other (the desirable node embedding space of G is as Fig. 1(d)). Separating self-embedding from neighbor-embedding is a solution to solve the dilemma and has been adopted by some recent studies, as it avoids introducing more noises during the aggregate process. As the example shown in Fig. 1(d), the separated representations can reasonably capture the disassortative structure presented in the original graph data (Table 1).

In this paper, the authors explore advanced graph neural networks to learn representations that embody both separation and aggregation. This is motivated by many real-world graphs are having complex relations, such as a mixture of assortative and disassortative structures. Learning only separation or aggregation information is therefore insufficient to satisfy discriminative ability, nor compensate for the scarce supervision signals for training a model, and hard to infer the neighbor's label for the node with label-given, which also creates more headaches to mine implicit supervision signals furtherly. Moreover, in many real networks, misconnections or missing connections exist frequently, leading to noise during the aggregation process of GNNs.

To this end, the authors propose a novel approach, termed Graph neural networks via contrast between separation and aggregation for self and neighborhood (COSA). It pursues the optimal friendly neighbor space, contrasts the separated signals with the friendly aggregated signals. Simultaneously, it introduces the variational expectation maximization algorithm (VEM) (Neal & Hinton, 1998) to infer node without label-given approximate the intractable posterior distribution and alternately optimize the whole model. COSA synthesizes two novel modules: the contrast separation and aggregation (CSA) module (GCNq_φ) and the structure adapter (Struc-Adapter) module (including the search adapter (SA) module and the GCNp_θ layers). More specifically, the COSA approach first obtains a friendly neighbor space incorporating the feature semantic and topological semantic which are complementary to each other. To derive deeper semantic correlation, in the sequential set of GCNq_φ layers, the CSA module aggregates information in the original neighbor space and the friendly neighbor space respectively, and produces the customized aggregated signals by fusing properly. In addition, in the sequential set of GCNq_φ layers, the CSA module also produces the separated signals by separating the self-embedding and the neighbor-embedding. In the last GCNq_φ layer, CSA contrasts the customized aggregated signals and the separated signals to make up for the scarce supervision signals implicitly, and is jointly optimized with structure prediction and specific downstream graph learning task to leverage different supervisory signals for learning the potential label distribution. Meanwhile, researchers exploit the SA

Table 1
Symbols and notations.

Symbol	Explanation
G	An undirected graph $G = (V, E, X, Y)$.
V	The nodes set of G with $\|V\| = n$.
E	The edge set of G with $\|E\| = e$.
$\boldsymbol{X}$	The node feature matrix with $\|\boldsymbol{X}\| = m$.
$\boldsymbol{Y}$, $\hat{\boldsymbol{Y}}$, $\boldsymbol{Y_L}$, $\boldsymbol{Y_U}$	The set of ground truth labels of nodes with $\|\boldsymbol{Y}\| = n$, the predicted probabilities, the given and ungiven labels, respectively.
$\boldsymbol{Z}$	$\boldsymbol{Z}\in\mathbb{R}^{n\times r}$, the matrix of node embeddings, and the i-th row of $\boldsymbol{Z}$ is the embeddings of node i.
r	The dimension of node embeddings, and $r \ll (m, n)$.
$\boldsymbol{A}$	The adjacency matrix of G, where $\boldsymbol{A}_{ij}$ is the i-th row and j-th column of $\boldsymbol{A}$.
$\boldsymbol{D}$, d_i	The degree matrix of G, and d_i is the degree of node i.
$\bar{G}$, $\bar{\boldsymbol{A}}$, $\bar{\boldsymbol{D}}$	Adding the self-loop, G is changed to $\bar{G}$. $\bar{\boldsymbol{A}}$ is the adjacency matrix of $\bar{G}$, and $\bar{\boldsymbol{D}}$ is the degree matrix of $\bar{G}$.
G', $\tilde{\boldsymbol{A}}'$, $\tilde{\boldsymbol{D}}'$	Adding the self-loop, $\tilde{\boldsymbol{A}}'$ is the adjacency matrix of the customized neighbor space G', and $\tilde{\boldsymbol{D}}'$ is the degree matrix of G'.
$\boldsymbol{\Gamma}_f$, $\boldsymbol{\Gamma}_s$, $\boldsymbol{\Gamma}_m$, $\boldsymbol{X}_s$, $\boldsymbol{X}_n$, $\boldsymbol{X}_\varphi$, $\boldsymbol{X}_\phi$, $\boldsymbol{X}_\omega$, $\boldsymbol{X}_\pi$, $\boldsymbol{X}_\theta$	The features embeddings in the feature semantic space, the structure semantic space, the structure and feature mixed semantic space, the self, neighborhood, GCNq_φ, GCNq_ϕ, GCNq_ω, GCNq_π, and GCNp_θ, respectively.
$\boldsymbol{W}_f$, $\boldsymbol{W}_s$, $\boldsymbol{W}_\phi$, $\boldsymbol{W}_\omega$, $\boldsymbol{W}_\pi$, $\boldsymbol{W}_\theta$	The trainable weight matrix.
Y_q, Y_p	The potential label distribution learned by GCNq_φ and GCNp_θ, respectively.
k	The number of layers in GCNq_φ, GCNq_ϕ, GCNq_ω, GCNq_π, and GCNp_θ, respectively.
α, β	The weights of L_c and L_s.
λ_f, λ_s	The weight for $\boldsymbol{\Gamma}_f$ and $\boldsymbol{\Gamma}_s$.
L_t, L_c, L_s, L_φ, L_θ	The loss of semi-supervised related to the specific task, the loss of contrastive learning, the loss of reconstructing the degree of nodes, the loss in GCNq_φ, and the loss in GCNp_θ, respectively.
$\|\|$, $/^{\mathrm{h}}$, $\odot$, $diag^{\rho}(\cdot)$, $sim(\bullet,\bullet)$, σ	The operator of concatenation, Hadamard division, Hadamard product, multiplying the diagonal elements of the matrix by the regulatory factor ρ, the similarity function measures the difference between nodes, and the activation function, respectively.
ξ	The weight of $\boldsymbol{X}_\omega$ when fused.
ε, τ	The population in a generation, and the generation.
$K_{i,j}$, $O_{i,j}$, $U_{i,j}$	The i-th initial individual, mutation individual, and the trial individual of j-th generation, respectively.
t, t_φ, t_θ	The t-th iteration in the COSA model, GCNq_φ, and GCNp_θ, respectively.
sp_{ai}	The length of the shortest path between two nodes v_a and v_i.
ζ	The weight in contrastive learning.
μ_1, μ_2	The random numbers ranging in [0,1].



(a) A toy example graph with assortativity

(b) A toy example graph G with disassortativity

(c) Aggregation process of G in GNNs

(d) The desirable node embedding space of G

(e) Node embedding space of G learned by GNNs under the assortativity assumption

Fig. 1. (a) A toy graph with an assortative structure. (b) A social graph G with a disassortative structure. (c) The simplified version of the aggregation and propagation process of GNNs on G, where the nodes have different types of labels and features. GNNs are used to exploit both node features and graph structure information to learn vectorized graph representations. Learning algorithms are performed on the resulting representations. (d) The desirable node embedding space of G learned by the models under the disassortative assumption. (e) The node embedding space of G graph representations is typically used in existing GNNs under the assortative assumption inconsistent with the actual situation.

module of Struc-Adapter that utilizes the global search strategy (researchers apply the simple reliable but effect algorithm – differential evolution (DE) as demonstration in this paper) to pursue the optimal friendly neighbor space to mitigate the unreasonable of connection. In the sequential set of GCNp_θ layers, together with the proposed CSA to model the approximated posterior according to variational EM for learning reasonable representations on complex graphs.

In summary, the contributions of this work include:

1) The authors propose a novel graph neural network framework termed COSA to learn reasonable graph representations for real-world graphs with disassortativity. COSA mitigates the unreasonable of connection, fully utilizes the extra supervision information, and weakens the impact of irrelevant information. It offers a novel perspective on learning complex graphs with disassortativity.
2) The authors introduce the CSA module, in which the customized aggregated signals incorporating the feature semantic and topological semantics are dynamically synthesized. Meanwhile, it learns the separated signals between self and neighborhood, extracts intrinsic interrelationship by contrastive learning, makes up for the scarce supervision signals, and is jointly optimized with structure prediction and specific downstream graph learning tasks.
3) The authors further introduce the Struc-Adapter module that performs differential evolution to search the optimal customized neighbor space for mitigating the unreasonable of connection. Together with the CSA, the Struc-Adapter model approximated posterior via the variational EM for learning reasonable representations on complex graphs.
4) The authors conduct extensive experiments on eight real-world benchmark datasets from diverse application domains and justify the efficacy of the proposed COSA approach in multiple popular graph learning tasks – node classification and link prediction. The empirical results show that the model COSA achieves new state-of-the-art performance on these tasks, compared to eight state-of-the-art models.

Recently, some other studies have been dedicated to learning graph neural networks for disassortative networks. This work uniquely differs from these prior studies in three main aspects: 1) the CSA module learns the intrinsic interrelationship between the customized aggregated signals and the separated signals which originate from self and neighborhood, enabling the learning of more diverse and complex structures, 2) the Struc-Adapter module learns optimal synthesis of the feature semantic and topological semantic according to the downstream graph learning task, and 3) joint the CSA and Struc-Adapter module via variational EM frame to model the approximated posterior for learning reasonable representations on complex graphs with disassortativity.

The rest of the paper is organized as follows. Section 2 discusses the related studies. Section 3 gives the preliminaries. Section 4 presents the proposed COSA model. Section 5 provides experimental results and analyses. Finally, Section 6 concludes this paper.

2. Related work

2.1. Graph neural networks

Recently, GNNs have achieved great success in solving machine learning problems on graph data (Cai et al., 2018; Wu et al., 2021; Zhang et al., 2020). A pioneering work of GNNs – GNN (Scarselli et al., 2009) – extends neural network methods to graph domains and opens the era of graph neural networks. Defferrard, Bresson, and Vandergheynst (2016) propose the spectral approach ChebNet which removes the computationally expensive Laplacian eigen decomposition and leverages the Chebyshev polynomials to approximate convolution filters. Kipf and Welling (2017) develop a widely used spectral method – Graph convolutional network (GCN) – which learns the node representations by the first-order approximation. Different from the spectral methods, spatial methods are simpler. Hamilton, Ying, and Leskovec (2017) exploit GraphSAGE which learns aggregators by sampling and aggregating neighbor information. Velickovic et al. (2018) develop GAT which can assign different weights to different neighborhood for target node while not require to know the graph structure beforehand. Wu et al. (2019) propose SGC which removes nonlinearities and collapsing weight matrices to reduce the complexity while retaining state-of-the-art performance. Wang, Zhu, Bo, Cui, Shi, and Pei (2020) develop AM-GCN which offers different weights for extracting and fusing adaptively from node features, topological structures, and their combinations simultaneously. To balance the information between graph structure and node features, Jin et al. (2021) exploit SimPGCN which adaptively integrates graph structure and node features, and offers a feature similarity preserving aggregation solution. All these GNN approaches are focused on assortative graph modeling only.

2.2. Disassortative graph representation learning

Recently, some efforts have been dedicated to generalizing GCN to heterophilic networks. Zhu et al. (2020) propose H2GCN which designs ego- and neighbor-embedding separation, higher-order neighborhoods, and combination of intermediate representations. From the perspective of frequency signals, Bo, Wang, Shi, and Shen (2021) provide FAGCN a self-gating mechanism so that the model can integrate different signals in the process of message passing adaptively. Zhu et al. (2021) provide CPGNN which exploits the compatibility matrix for modeling the homophily level. He et al. (2021) develop BM-GCN which incorporates block modeling into the aggregation process according to the homophily degree. Wang, Wang, Jin, He, and Huang (2022) develop HOG which provides two measurements of homophily degree and incorporates a learnable homophily degree matrix into graph convolution framework. Pei, Wei, Chang, Lei, and Yang (2020) propose GeomGCN which consists of node embedding, structural neighborhood, and bi-level aggregation for learning on graphs.

Different from these works, this approach incorporates the representations in the original neighbor space and the customized neighbor space which dynamically synthesis the feature semantic and topological semantic according to the downstream graph learning task, and learn the intrinsic interrelationship between the customized aggregated signals and the separated signals manipulating the contrastive learning. Such Fusing effective information meticulously can help to reduce the risk of introducing noise and irrelevant information, capture the rational neighbors, and avoid negative impact on the prediction of downstream tasks. In addition, this approach incorporates the variational EM frame to model the approximated posterior for further offsetting the scarce supervision signals and making representations more distinguishable.

2.3. Contrastive learning on graphs

Originating from self-supervision learning (SSL), contrastive learning methods have mushroomed these years, which learn the discrimination of representations with contrasting positive and negative samples by some form of data augmentation techniques. Zhu et al. (2021) develop GCA which designs augmentation schemes based on node centrality measures and adds noise to unimportant node features. Wan, Pan, Yang, and Gong (2021) produce CG3 which maximizes the agreement between the representations learned from global and local views, and leverages the underlying relationship between the input graph topology and data feature. Xia et al. (2022) provide ProGCL which devises two schemes to estimate the probability of a negative being true one and boost the performance of GCL. Yu et al. (2022) develop QRec which discards the augmentations and instead adds uniform noises to the embedding space for creating contrastive views. Zhu, Guo, Wu, and Tang (2022) develop RoSA which leverages the earth mover's distance to model the distribution transform and introduces adversarial training

to enhance the robustness.

Unlike these approaches, the proposed CSA module performs contrastive learning in diverse receptive fields in different layers, including the separated signals and the customized neighbor space which dynamically synthesis the feature semantic and topological semantic. It is jointly optimized with structure prediction and downstream graph learning objectives to learn representations.

3. Preliminaries

3.1. Symbols and notations

3.1.1. Problem and related techniques

Problem. Let $G = (V, E, \boldsymbol{X}, \boldsymbol{Y})$ be an undirected graph, V is the set of n nodes, E is the set of e edges, $\boldsymbol{X}$ is the set of m features for all nodes, and $\boldsymbol{Y}$ is the set of node labels. $\boldsymbol{A}$ is an adjacency matrix of G. The problem is to learn reasonable r-dimensional representations for each node from an undirected graph G, and $r \ll \min(m, n)$.

General EM Algorithm. Finding the maximum likelihood solution for a model containing latent variables is the basic objective of EM algorithm which contains two steps – E-step and M−step. The authors denote the set of observed data as x, the set of latent variables as h and the set of parameters as θ. In the E-step, the posterior distribution of latent variables h is derived as $p(h|x; \theta_{t\text{-}1})$, employing the observed data x and the parameters $\theta_{t\text{-}1}$ estimated by the last iteration. While in the M-step, maximize the log-likelihood obtained in E-step to calculate the value of parameter θ_t as follows:

$$\theta_t = \underset{\theta}{\operatorname{argmax}}\left(\mathbb{E}_{p(h|x,\theta_{t-1})}[\log p(x, h|\theta)]\right). \tag{1}$$

The estimated values of the parameters θ_t are used in the next E-step, and the process is carried out alternately.

Differential Evolution Algorithm. There have many excellent optimization algorithms, such as the model in Belciug (2022) and the modified version of the swarm algorithm (Jovanovic, Jovanovic, Bacanin, Jovancai Stakic, Antonijevic, Magd, & Zivkovic, 2022). Differential evolution (DE) algorithm which is one of the optimization algorithms is developed by by Storn and Price (1997) and Wu et al. (2022). DE is a stochastic global search optimization algorithm including four operations, i.e., initialization, mutation, crossover, and selection. DE starts with choosing a population of NP candidate solutions (individuals) in the searching space uniformly and randomly. In the mutant step (three most frequently adopted mutation strategies are as following: Rand/1, Best/1, and RandToBest/1), DE generates a mutant vector for individuals in the current population. Next, to increase population diversity, in the crossover step (two most frequently schemes: arithmetic and binominal), a new trial vector is produced by the current vector and its corresponding mutant vector. Then, in the selection step, DE must decide which individual between the current vector X_i and the trial vector U_i will be survived (when $g(X_i)$ is better than $g(U_i)$, the authors choose X_i; otherwise, U_i is retained. Here, $g(\bullet)$ is the objective function).

4. Proposed approach

4.1. The overall architecture

Overall, the proposed approach COSA aims to learn reasonable representations for nodes of networks with disassortativity. As illustrated in Fig. 2, COSA contains two major modules: the CSA module (the blue dash frame in Fig. 2) and the Struc-Adapter module (the purple dash frame in Fig. 2).

Particularly, given a network G with n nodes, e edges, and m features as input, each GCNq_φ layer in CSA contains GCNq_ϕ, GCNq_ω and GCNq_π which learn the separation signals between self-embedding and the neighbor-embedding, aggregated signals in the original neighbor space, and aggregated signals in the friendly neighbor space G', respectively, and the latter two kind of aggregated signals are fused properly. Further, in the last GCNq_φ layer, CSA contrasts the customized aggregated signals and the separated signals by contrastive learning for making up the scarce supervision signals, and jointly optimizes with the structure

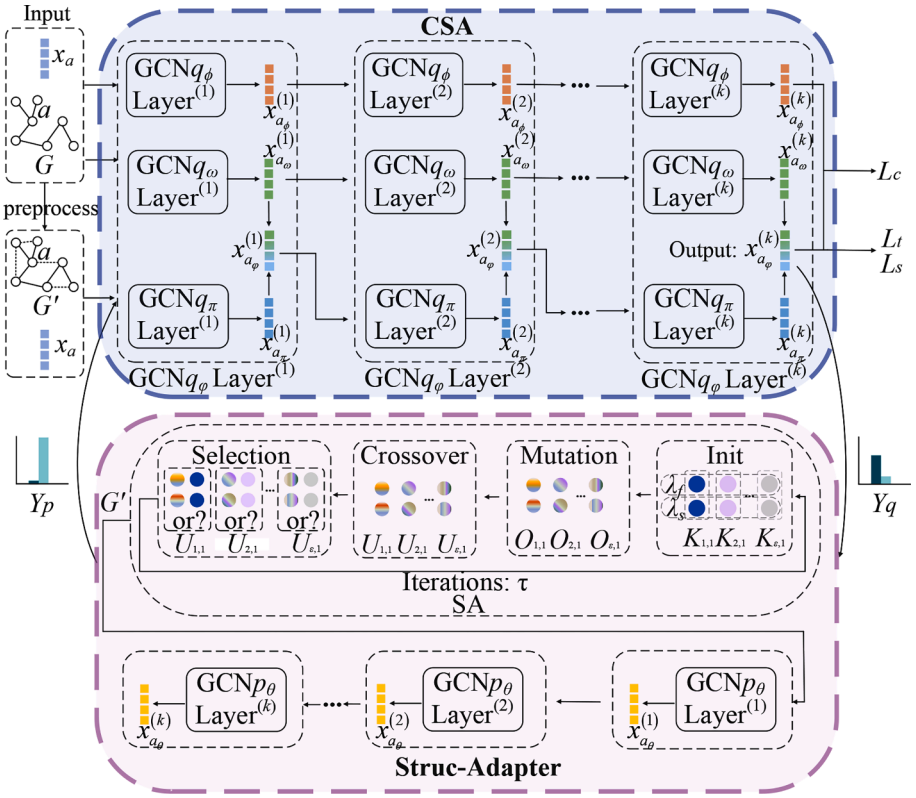


Fig. 2. The overall architecture of the COSA model deriving from the variational EM. The inputs of the whole model are the network G and features (the feature of node a is x_a). The model is two-fold: a) contrast separation and aggregation modeling (CSA module which composes of a GCNq_φ layer which consisting three GCNs – GCNq_ϕ, GCNq_ω and GCNq_π): each GCNq_ϕ layer learns the separated embeddings for nodes between self-embedding and the neighbor-embedding; each GCNq_ω layer learns aggregated signals in the original network G; each GCNq_π layer learns aggregated signals in the customized network G', then fuses with the previous signals properly; and finally in the last GCNq_φ layer, contrasts the customized aggregated signals and the separated signals, jointly optimizes with the structure prediction and the downstream graph learning objectives, and learns the potential label distribution; and b) the optimal structure modeling (Struc-Adapter module which includes the SA module and the GCNp_θ layers): performs the differential evolution algorithm to optimize G' and learns the potential label distribution in the GCNp_θ layers, then models the approximated posterior together with CSA according to variational EM. The output of representation for node a is $x_{a\varphi}$.

prediction and the downstream graph learning objectives to learn the potential label distribution. The objective function L_φ of the CSA module is:

$$\min(L_t + \alpha L_c + \beta L_s), \tag{2}$$

where L_t denotes the loss of the specific downstream task, L_c is the self-supervised loss function of the separation and aggregation between self-embeddings and neighborhood-embeddings, L_s is the structure reconstruction loss function, and α, β denote the corresponding weights (α, β are the hyper-parameters, and the ranges are 0 to 1. In this experiment, please refer *Section 5.1*).

The Struc-Adapter module optimizes the friendly neighbor space G' according to the differential evolution algorithm and utilizes GCNp_θ to update the potential label distribution. The objective function L_θ of the Struc-Adapter module is:

$$\min L_t, \tag{3}$$

where L_t denotes the loss of the specific downstream task. Together with CSA, the Struc-Adapter module model the approximated posterior according to variational EM (Neal & Hinton, 1998) for learning reasonable representations $\boldsymbol{Z}$ on complex networks. The authors elaborate each module of the approach in the following sections.

4.2. The variational EM processing

Generally, the disassortative network has usually complex structures, since the features or labels of neighbors in many real-world networks are often different from the target node'. It is quite hard to infer the neighbor's label for the node with label-given which also creates more headaches to mine implicit supervision signals furtherly. To infer the labels of the unlabeled nodes Y_U, the authors need to estimate the posterior distribution $p_\theta(\mathrm{Y}_U \mid \mathrm{X}, \mathrm{A}, \mathrm{Y}_L)$, where Y_L is the given labels, and θ is the parameter. To approximate the intractable posterior distribution, the variational EM algorithm simultaneously to and alternately optimize the whole model. To this end, we introduce a distribution $q_\varphi(\mathrm{Y}_U \mid \mathrm{A}, \mathrm{X}, \mathrm{Y}_L)$ parameterized by φ to approximate the true posterior the true posterior $p_\theta(\mathrm{Y}_U \mid \mathrm{X}, \mathrm{A}, \mathrm{Y}_L)$. Afterwards, we can write the Evidence Lower Bound (ELBO) as:

$$\begin{aligned} \log p_\theta(Y_L|X) &\geqslant \log p_\theta(Y_U, Y_L|X) - D_{KL}(q_\varphi(Y_U|A,X,Y_L)\|p_\theta(Y_U|A,X)), \\ &= \mathbb{E}_{q_\varphi(\mathrm{Y_L}|\mathrm{X})}[\log \mathrm{p}_\theta(\mathrm{Y_U}, \mathrm{Y_L}|\mathrm{X}) - \log \mathrm{q}_\varphi(\mathrm{Y_U}|\mathrm{X})], \end{aligned} \tag{4}$$

where $\mathrm{D}_{KL}(\cdot||\cdot)$ represents the Kullback-Leibler divergence between two distributions. In this approach, $q_\varphi(\mathrm{Y_U}|\mathrm{A}, \mathrm{X}, \mathrm{Y_L})$ is composed of a series GCNq_φ layers, and $p_\theta(\mathrm{Y}|\mathrm{A}, \mathrm{X})$ is composed of a series GCNp_θ layers. The details of each GNN are presented as follows.

4.3. Preprocess

This component is responsible for developing the customized neighbor space G' incorporating the feature semantic $\boldsymbol{\Gamma}_f$ and structure semantic $\boldsymbol{\Gamma}_s$ which are complementary to each other.

The feature semantic space collection. The authors adopt the multi-layer perceptron (MLP) to capture the feature semantic space $\boldsymbol{\Gamma}_f$ as follows:

$$\boldsymbol{\Gamma}_f = \mathrm{softmax}\big(\sigma\big(XW_f\big)\big), \tag{5}$$

where σ is the activation function; $\boldsymbol{X}$ is the features; $\boldsymbol{W}_f$ is the learned weight matrix.

The structure semantic space collection. To capture the structure semantic space $\boldsymbol{\Gamma}_s$, this article collects the connection patterns or subgraph of the target node around the neighbors. To this end, for a target node a, the subgraph G_a consists of the m layer of neighbors and a is the centre. The function of subgraph patterns calculated as follows:

$$\Delta_a = d_a \| d_i \| sp_{ai}, \quad \text{for every neighbors } i \text{ of target node } a, \tag{6}$$

where d_a, d_i denote the degree of nodes a and I; sp_{ai} is the length of the shortest path between two nodes v_a and v_i, $v_i \in Nm\ a \backslash \{v_a\}$, $Nm\ a = \{v_j \in V |\ sp_{ai} \leq m\}$; $||$ is the operator of concatenation.

Then, the method simulates random walks on the subgraph of each node, and utilizes the Skip-Gram (Perozzi, Al-Rfou, & Skiena, 2014) model with negative sampling to learn the structure semantic $\boldsymbol{\Gamma}_s$.

The customized neighbor space collection. First, the authors initialize the weights λ_f for $\boldsymbol{\Gamma}_f$ and λ_s for $\boldsymbol{\Gamma}_s$ utilizing the normalized accuracy of corresponding accuracy of semantic, and incorporate the feature semantic $\boldsymbol{\Gamma}_f$ and structure semantic $\boldsymbol{\Gamma}_s$ as follows:

$$\boldsymbol{\Gamma}_m = \lambda_f \boldsymbol{\Gamma}_f + \lambda_s \boldsymbol{\Gamma}_s, \tag{7}$$

where λ_f and λ_s are from 0 to 1. Then, the corresponding parts of $\boldsymbol{\Gamma}_s$ are replaced by the available ground-truth labels to obtain $\boldsymbol{\Gamma}'_s$, and the possibility of an edge between nodes can be computed as follows:

$$Q = \big(\boldsymbol{\Gamma}'^{\mathrm{T}}_m A \boldsymbol{\Gamma}'_m\big) /^h \big(\boldsymbol{\Gamma}'^{\mathrm{T}}_m A I\big), \tag{8}$$

where I is a matrix with all one elements; $/^h$ is Hadamard or component-wise division. In addition, the adjacency matrix A' of G' can be calculated as follows:

$$A' = \big(\boldsymbol{\Gamma}_m diag^\rho\big(QQ^{\mathrm{T}}\big)\boldsymbol{\Gamma}^{\mathrm{T}}_m\big) \odot A, \tag{9}$$

where $diag^\rho(\cdot)$ is a function of multiplying the diagonal elements of the matrix by the regulatory factor ρ; $\odot$ denotes the Hadamard or component-wise product.

4.4. Contrast separation and aggregation modeling

The CSA module composes of the GCNq_φ layer which consisting three GCNs – GCNq_ϕ, GCNq_ω and GCNq_π – for learning the intrinsic interrelationship and the extra supervision information between the optimal customized aggregated information and the separated information originating from self and neighborhood, and is jointly optimized with structure prediction and specific downstream graph learning task. Meanwhile, the module learns the potential label distribution. The technical details are presented as follows.

4.4.1. GCNq_ϕ

GCNq_ϕ is responsible for separating the self-embedding and the neighborhood embedding, and assembling the two embeddings together of the representation $\boldsymbol{X}_\phi$ for subsequent as the reference and contrast. The self-embeddings $\boldsymbol{X}_s$ can be calculated as follows:

$$\boldsymbol{X}_s = \sigma(XW_s), \tag{10}$$

where σ is the activation function; $\boldsymbol{W}_s$ is the learned weight matrix. The neighborhood embedding $\boldsymbol{X}_n$ is composed and concatenated with the first-order and second-order neighbor embeddings:

$$\boldsymbol{X}_n = \big(((A - I_E)X)\|\big((AA^{\mathrm{T}} - A - I_E)X\big)\big), \tag{11}$$

where I_E is an identity matrix; $||$ is the operator of concatenation.

Then, the embedding $\boldsymbol{X}_\phi$ is concatenated by $\boldsymbol{X}_s$ and $\boldsymbol{X}_n$ as 'sandwich' to increase own proportion:

$$\boldsymbol{X}_\phi = \sigma\big((\boldsymbol{X}_s \| \boldsymbol{X}_n \| \boldsymbol{X}_s)\boldsymbol{W}_\phi\big), \tag{12}$$

where σ is the activation function; $\boldsymbol{W}_\phi$ is the learned weight matrix; $||$ is the operator of concatenation.

4.4.2. GCNq_ω

GCNq_ω is a GCN learning the aggregating embeddings $\boldsymbol{X}_\omega$ in the original neighbor space G. The embeddings calculated in the *k-th* GCN$_\omega$

layer are as follows:

$$\boldsymbol{X}_{\omega}^{(k)} = \sigma(\widetilde{\boldsymbol{D}}^{-1/2}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-1/2}\boldsymbol{X}_{\omega}^{(k-1)}\boldsymbol{W}_{\omega}^{(k)}), \tag{13}$$

where

$$\widetilde{\boldsymbol{D}}_{ii} = \sum_{j}\widetilde{\boldsymbol{A}}_{ij}, \widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_{E}, \tag{14}$$

where σ is the activation function; $\boldsymbol{W}_{\omega}^{(k)}$ is the trainable weight matrix; $\boldsymbol{I}_{E}$ is the identity matrix; $\tilde{\boldsymbol{A}}$ is the adjacency matrix of G adding the self-loop; $\widetilde{\boldsymbol{D}}$ is the corresponding degree matrix.

4.4.3. GCNq_{π}

GCNq_{π} first learns the aggregating embeddings $\boldsymbol{X}_{\pi}$ in the customized neighbor space G'. The embeddings can be calculated in the first GCN$_{\pi}$ layer are as follows:

$$\boldsymbol{X}_{\pi}^{(1)} = \sigma(\widetilde{\boldsymbol{D}}^{'-1/2}\widetilde{\boldsymbol{A}}'\widetilde{\boldsymbol{D}}^{'-1/2}\boldsymbol{X}\boldsymbol{W}_{\pi}^{(1)}), \tag{15}$$

where

$$\widetilde{\boldsymbol{D}}'_{ii} = \sum_{j}\widetilde{\boldsymbol{A}}'_{ij}, \widetilde{\boldsymbol{A}}' = \boldsymbol{A}' + \boldsymbol{I}_{E}, \tag{16}$$

where σ is the activation function; $\boldsymbol{W}_{\pi}^{(1)}$ is the trainable weight matrix; I_E is the identity matrix; $\tilde{\boldsymbol{A}}'$ is the adjacency matrix of G' adding the self-loop; $\widetilde{\boldsymbol{D}}'$ is the corresponding degree matrix. Then the embeddings $\boldsymbol{X}_{\omega}$ and $\boldsymbol{X}_{\pi}$ are fused properly:

$$\boldsymbol{X}_{\varphi}^{(k)} = \xi\boldsymbol{X}_{\omega}^{(k)} + \boldsymbol{X}_{\pi}^{(k)}, \tag{17}$$

where $k > 1$; ξ is the weight.

In the last GCNq_{φ} layer, we obtain $\boldsymbol{Z} = \boldsymbol{X}(k)\ \varphi$. The task-specific loss function L_t is as follows:

$$L_t = CE(\boldsymbol{Z}, \boldsymbol{Y}), \tag{18}$$

where CE denotes the cross-entropy loss for node classification, or the Bayesian personalized ranking loss for link prediction.

To make up the scarce supervision signals, in the last GCNq_{φ} layer, CSA contrasts the customized aggregated signals $\boldsymbol{X}_{\varphi}$ and the separated signals $\boldsymbol{X}_{\phi}$ by contrastive learning, the objective L_c as in Eqs. (19) and (20):

$$L_c^{(k)} = -\frac{1}{n}\log\frac{\sum_{i\in V}e^{sim\left(\boldsymbol{X}_{\varphi_i},\boldsymbol{X}_{\phi_i}\right)/\zeta}}{\sum_{i,j\in V,i\neq j}e^{sim\left(\boldsymbol{X}_{\varphi_i},\boldsymbol{X}_{\phi_j}\right)/\zeta} + \sum_{i\in V}e^{sim\left(\boldsymbol{X}_{\varphi_i},\boldsymbol{X}_{\phi_i}\right)/\zeta}}, \tag{19}$$

where

$$sim\left(\boldsymbol{X}_{\varphi_i},\boldsymbol{X}_{\phi_j}\right) = \frac{\boldsymbol{X}_{\varphi_i}\cdot\boldsymbol{X}_{\phi_j}}{|\boldsymbol{X}_{\varphi_i}|\left|\boldsymbol{X}_{\phi_j}\right|}, \tag{20}$$

where ζ is recommended to be set as 1 empirically; $sim(\bullet,\bullet)$ is the similarity function. The node self is the positive sample, while different nodes are the negative samples.

To explore whether the model can maintain the connection patterns of the target node, we achieve the goal by reconstructing the degree of nodes, via leveraging MLP, and the loss function L_s is as follows:

$$L_s = MSE\left(d_v, \sigma\left(\mathrm{MLP}\left(\boldsymbol{X}_{\varphi_v}^{(k)}\right)\right)\right), \tag{21}$$

where MSE denotes the mean squared error or L2 loss.

Therefore, the overall loss function L_{φ} in CSA is defined as follows:

$$\begin{aligned}&\min L_t + \alpha L_c + \beta L_s\\ &= \min CE(\boldsymbol{Z},\boldsymbol{Y}) - \frac{\alpha}{n}\log\frac{\sum_{i\in V}e^{sim\left(\boldsymbol{X}_{\varphi_i},\boldsymbol{X}_{\phi_i}\right)/\zeta}}{\sum_{i,j\in V,i\neq j}e^{sim\left(\boldsymbol{X}_{\varphi_i},\boldsymbol{X}_{\phi_j}\right)/\zeta} + \sum_{i\in V}e^{sim\left(\boldsymbol{X}_{\varphi_i},\boldsymbol{X}_{\phi_i}\right)/\zeta}}\\ &+\beta MSE\left(d_v, \sigma\left(\mathrm{MLP}\left(\boldsymbol{X}_{\varphi_v}^{(k)}\right)\right)\right),\end{aligned} \tag{22}$$

aiming to optimize the GCNq_{φ} layers by jointly minimizing simultaneously from the task-specific loss function, the contrastive loss function, and the L2 loss function of degree reconstruction. Meanwhile, infer the potential label distribution Y_q.

4.5. The optimal structure modeling

In this section, the authors introduce the Struc-Adapter module which consists SA module and the GCNp_{θ} layers, to optimize the customized neighbor space G' and mitigate the unreasonable of connections, and learn the potential label distribution.

4.5.1. The SA module

The SA module performs differential evolution to search the optimal customized neighbor space G'. Firstly, we initialize a population consisting with ε individuals which is a set of λ_f and λ_s (as shown in *Section 4.3.1*) actually. One of the individual $K_{i,j}$ is denoted by [*K*1 *i,j*, *K*2 *i,j*]. *K1 *,** stands for λ_f; *K2 *,** stands for λ_s; *i* stands for the *i-th* individual and the scope is {1, 2,…, ε}; *j* stands for the *j-th* generation and the scope is {1, 2,…, τ}; *K*1 *i,j* stands for λ_f; *K*2 *i,j* stands for λ_s.

Mutation. The mutation operation is to generate a series of new mutant individuals $O_{i,j}$. Since popularity and robustness, we choose the Rand/1 strategy as demonstration (Bilal, Pant, Zaheer, Garcia-Hernandez, & Abraham, 2020). The mutation individuals $O_{i,j}$ are generated as follows:

$$O_{i,j} = K_{a_1,j} + \mu_1\cdot\left(K_{a_2,j} - K_{a_3,j}\right), \tag{23}$$

where $K_{a1,j}$, $K_{a2,j}$, and $K_{a3,j}$ are different individuals randomly selected from τ individuals in generation j; μ_1 is a random number ranging in [0,1].

Crossover. The crossover operation is to generate a series of new trial individuals $U_{i,j}$ furtherly leveraging the original individual $K_{i,j}$ and its corresponding mutant individual $O_{i,j}$. This paper adopts the arithmetic crossover strategy (Bilal et al., 2020) as a demonstration, and the trial individuals $U_{i,j}$ are generated as follows:

$$U_{i,j} = K_{i,j} + \mu_2\cdot\left(O_{i,j} - K_{i,j}\right), \tag{24}$$

where μ_1 is a random number from 0 to 1. Here, we should check whether the value of $U_{i,j}$ is out of range. Specifically, if $U_{i,j}$ is out of upper bound or below lower bound, we need to do truncate.

Selection. In the selection operation, a series of new individuals $U_{i,j}$ need to be decided which individual can survive in the next generation $j + 1$ according to the objective function value. Specifically, the first item of $U_{i,j}$ is actually λ_f, and the second item is λ_s. According to Eqs. (7)–(9) and (27), the customized neighbor space G' and the loss of GCNp_{θ} can be obtained. Furthermore, we get the new individual $U_{i,j}$ as follows:

$$U_{i,j} = \begin{cases} K_{i,j} & \text{if } \mathrm{L}_{\theta}\left(\mathrm{K_{i,j}}\right) < \mathrm{L}_{\theta}\left(\mathrm{F_{i,j}}\right) \\ F_{i,j} & \text{otherwise} \end{cases}. \tag{25}$$

As a result, the performance of the next generation $j + 1$ is always better than, or at least the same as, the current generation j. After τ generation, we obtain an excellent population. Then pick out the best individual F_{best} to acquire the minimum of L_{θ} in the population.

4.5.2. $GCNp_\theta$

After the optimal customized neighbor space G' is obtained, in the sequential set of GCNp_θ layers, the potential label distribution is learned as follows:

$$\boldsymbol{X}_\theta^{(k)} = \sigma(\widetilde{\boldsymbol{D}}'^{-1/2}\widetilde{\boldsymbol{A}}'\widetilde{\boldsymbol{D}}'^{-1/2}\boldsymbol{X}\boldsymbol{W}_\theta^{(k)}), \tag{26}$$

and the detailed explanations Eq. (26) refer to Eq. (16). The task-specific loss function L_θ is as follows:

$$L_\theta = CE(\boldsymbol{Z}, \boldsymbol{Y}), \tag{27}$$

and the detailed explanations Eq. (27) refer to Eq. (18). Meanwhile, infer the potential label distribution Y_p.

The Struc-Adapter module is together with the CSA module to model the approximated posterior according to variational EM for learning reasonable representations on complex graphs.

4.6. Algorithm of COSA and its complexity analysis

Algorithm design. The full COSA algorithm is presented in Table 2.

The adjacency matrix $\boldsymbol{A}$ of graph G, $\boldsymbol{X}$, $\boldsymbol{Y}$, r (the dimension of embeddings), k (the number of the layer), h_s (the hidden size), t (the number of COSA iterations), t_φ (the number of CSA iterations), t_θ (the number of Struc-Adapter iterations), and τ (the generation in SA) are the inputs of the algorithm.

After initializing all the parameters (including all the trainable weight matrices), steps 2–27 are the main training steps. More specifically, after the embeddings $\boldsymbol{X}_\varphi$ in GCNq_φ (steps 5–7) and the loss L_φ (steps 8–11) are computed, we can obtain the potential label distribution Y_q (step 12). It then learns how to revise the customized neighbor space G' in steps 16–19. After the embeddings $\boldsymbol{X}_\theta$ in GCNq_θ (step 21) and the loss L_θ (step 22) are computed, we can obtain the potential label distribution Y_p (step 23). After several iterations, we can get the embeddings matrix $\boldsymbol{Z}$ (step 26).

Complexity analysis. Computing $\boldsymbol{X}_\phi$, $\boldsymbol{X}_\omega$, $\boldsymbol{X}_\pi$, and $\boldsymbol{X}_\theta$ are the main time cost steps. For computing $\boldsymbol{X}_\phi$, the time complexity is $O(c_1 e)$ which is linear in the number of edges; for computing $\boldsymbol{X}_\omega$, the time complexity is $O(c_2 e)$; for computing $\boldsymbol{X}_\pi$, the time complexity is $O(c_3 e)$; for computing $\boldsymbol{X}_\theta$, the time complexity is $O(c_4 e)$ (where c_1, c_2, c_3 and c_4 are constants). Thus, the overall computation cost of COSA is $O(c_1 e + c_2 e + c_3 e + c_4 e) \rightarrow O(ce)$ (where c is a constant).

Table 2
Algorithm COSA.

Algorithm: COSA
Input: the adjacency matrix $\boldsymbol{A}$ of graph G, $\boldsymbol{X}$, $\boldsymbol{Y}$, r, k, h_s, t, t_φ, t_θ, τ
Output: the nodes embeddings matrix $\boldsymbol{Z}$ for each node

Step:
1. initialize k, r, h_s, t, t_φ, t_θ, τ
2. preprocess: compute the customized neighbor space collection by Eqs. (5)–(9)
3. for $i \le t$:
4. for $j \le t_\varphi$:
5. compute $\boldsymbol{X}_\phi$ by Eqs. (10)–(12)
6. compute $\boldsymbol{X}_\omega$ by Eqs. (13) and (14)
7. compute $\boldsymbol{X}_\pi$ and $\boldsymbol{X}_\varphi$ by Eqs. (15)–(17)
8. compute L_t by Eq. (18)
9. compute L_c by Eqs. (19) and (20)
10. compute L_s by Eq. (21)
11. compute L_φ by Eq. (22)
12. infer Y_q
13. end for
14. for $l \le t_\theta$:
15. for $s \le \tau$:
16. mutation operater by Eq. (23)
17. crossover operate by Eq. (24)
18. selection operate by Eq. (25)
19. update G'
20. end for
21. compute $\boldsymbol{X}_\theta$ by Eq. (27)
22. compute L_θ by Eq. (22)
23. infer Y_p
24. end for
25. end for
26. $\boldsymbol{Z} \leftarrow \boldsymbol{X}_\varphi$
27. return $\boldsymbol{Z}$

5. Experiments

In the experiments, the authors aim at answering the following three research questions:

- RQ.1: How effective is the proposed COSA model in learning diverse graph structures in real-world data?
- RQ.2: How does each module of COSA contribute to its overall performance?
- RQ.3: How robust is the performance COSA model w.r.t. hyper-parameter settings?

5.1. General settings

Datasets. Eight popular graph datasets from different application domains are used in the experiments, including Squirrel (Wang, Mou, Wang, Xiao, Ju, Shi, & Xie, 2021), Actor (Zhu et al., 2020), Chameleon (Kim & Oh, 2021), UAI (Wang et al., 2020), AIR-USA (Zhang, Wang, Shi, Liu, & Song, 2021) (USA for short), MS Academic (Kim & Oh, 2021) (MS for short), Disease (Zhang et al., 2021), and ACM (Wang et al., 2020). Their key statistics are shown in Table 3, with the introduction of each dataset given as follows. Particularly, to evaluate whether these real-world datasets have intricate graph structure, the authors use a graph complexity measure – homophily – to quantify some hidden yet important characteristics of the datasets used. In Table 3, $\alpha \in [0,1]$ denotes the average homophily of a graph which is introduced and used in Kim and Oh (2021). A larger α value implies that nodes in given graph tend to have a stronger positive association with their neighbors.

- **Chameleon** and **Squirrel** are two networks in Wikipedia with specific topics. In the two datasets, nodes are informative nouns in pages, and labels correspond to the monthly traffic of the pages.
- **Actor** is a part of film-director-actor-writer network, each node belongs to one of five roles, and the edges are co-occurances on the same Wikipedia page.
- **UAI** is a dataset for community detection with 2067 nodes and 28,311 edges.
- **USA** is an air-traffic network collected from the Statistical Office of the European Union from January to November 2016, respectively. The nodes are airports and the edges are commercial flights between airports. The nodes' labels are the levels of activity of the airports.
- **MS** is a co-authorship Microsoft Academic network. The nodes represent authors, edges are co-authorships, and features are the keywords from authors' papers.
- **Disease** contains relation between the states of being infected or not by SIR diseases, in which each node is a state. The full dataset

Table 3
Datasets and Their Key Statistics.

Dataset	α	Nodes	Edges	Features	Classes
Squirrel	0.21	5,201	217,073	2,089	5
Actor	0.23	7600	26,752	932	5
Chameleon	0.27	2,277	36,101	2,325	5
UAI	0.44	3067	28,311	4973	19
USA	0.55	1,190	13,599	238	4
MS	0.83	18,333	81,894	6,805	15
Disease	0.88	1,044	1,043	1,000	2
ACM	0.89	3,025	13,128	1,870	3

simulates the propagation of the disease. The hyperbolicity (how hierarchical or tree-like structure contains in the graph) of this dataset is stronger (Zhang et al., 2021).

- **ACM** is a research paper citation network. The nodes in the network are publications, and the edges are citation links. The features of nodes are bag-of-words representations of the papers. In ACM, all the papers are divided into 3 classes.

Evaluation metrics. To evaluate the proposed COSA model, two types of graph learning tasks are tested, i.e., node classification and link prediction. Micro-F1 and Macro-F1 are adopted as the node classification metrics (Zhang et al., 2020). Micro-F1 needs to calculate metrics globally by counting the total true positives, false negatives, and false positives. While Macro-F1 needs to calculate metrics for each class and find their unweighted mean. AUC (area under the receiver operating characteristic curve) and AP (average precision) focus on measuring the accuracy of the algorithm on the whole, which are used as link prediction metrics (Zhang et al., 2020).

Baselines. In the experiments, the authors compare the proposed COSA model with eight recent state-of-the-art graph embedding models. These models include popular GNN models: ChebNet (Defferrard et al., 2016), GCN (Kipf & Welling, 2017), SAGE (Hamilton et al., 2017), and SGC (Wu et al., 2019); GNNs for disassortative graphs: H2GCN (Zhu et al., 2020), FAGCN (Bo et al., 2021), and HOG (Wang et al., 2022); and self-supervised learning-enhanced GNNs: SimPGCN (Jin et al., 2021). These models are chosen because they are closely related to one or more modules of COSA. A detailed introduction of the models is given in Table 4.

Implementation details. As for ChebNet, GCN, SGC, and SAGE, the authors use the implementations of the PyTorch Geometric library in all experiments. For H2GCN, HOG, FAGCN, and SimGCN, the authors use the source codes provided by the authors. Generally, for all the methods, the authors set the embedding dimension $r = 16$, the number of layers $k = 2$, the hidden size $h_s = 500$, the number of iterations $t = 300$, the optimal L_2 regularization with weight decay is in {5e-3, 5e-4, 5e-5}, learning rate is in [0.01–0.1], and dropout rate is in [0.0–0.6]. For the proposed method, the authors set the hyper-parameters: the weight of $\boldsymbol{X}_\omega$ $\xi = 0.01$, the generation $\tau = 5$, the population in a generation $\varepsilon = 10$, the weights of L_c and L_s: α and β are in [0.001, 1]. All the experiments are performed on a computer that has a 2.39 GHz E5-2680 CPU with 8 cores and 64 GB RAM, and 11G GTX1080ti GPUs.

5.2. Efficacy in graph modeling (RQ1)

To have a thorough performance evaluation of the COSA model, the authors examine the effectiveness of the learned representations in enabling two key graph learning tasks, including node classification and link prediction. Further, COSA is compared with eight state-of-the-art GNN models on eight diverse real-world graph datasets. In addition, the authors also compare their computational efficiency on each dataset. The authors repeat the experiment 20 times and report the average results.

Table 4
Eight competing models used in the experiments.

Method	Description
ChebNet (NIPS16′)	It leverages the Chebyshev polynomials to approximate the convolution filter to learn the node embeddings.
GCN (ICLR17′)	Using the first-order approximation of the Chebyshev filter, it learns node representations by aggregating information from neighbors.
SAGE (NIPS17′)	It learns aggregators by sampling and aggregating neighbor information.
SGC (ICML19′)	It is a kind of GCN which removes nonlinearities and collapsing weight matrices to reduce the complexity.
H2GCN (NIPS20′)	It captures ego- and neighbor-embedding separation, higher-order neighborhoods, and combination of intermediate representations.
FAGCN (AAAI21′)	It integrates different signals in the process of message passing adaptively by a self-gating mechanism.
SimPGCN (WSDM21′)	It offers a feature similarity preserving aggregation solution that adaptively integrates graph structure and node features.
HOG (AAAI22′)	It provides two measurements of homophily degree and incorporates a learnable homophily degree matrix into graph convolution framework.

Node classification. In this experiment, the Disease dataset is divided into three parts randomly: 30% for training (training ratio), 10% for validation, and 60% for test (Chami, Ying, Re, & Leskovec, 2019). For Squirrel, Actor and Chameleon datasets, 48%/32%/20 of nodes per class (L/C for short) are for train/validation/test (Zhu et al., 2020). Other datasets are divided into three parts randomly, with each dataset having the fixed training/validation/test set, where the label rate – 20 L/C – is used for training and 500/1000 nodes are used for validation/test. The Micro-F1 and Macro-F1 scores are reported in Table 5 (bold numbers represent the best average results). To better understand the comparison results, we also conduct a statistical significance test analysis. The Friedman test (Demsar, 2006) is carried out because it is effective in validating the performance of several models on multiple datasets. In addition, to check whether COSA has significantly better performance than every single model, the Wilcoxon signed-ranks test (Demsar, 2006) is also carried out. The F-rank scores of the Friedman test and p-value scores of the Wilcoxon signed-ranks test are summarized in the last four columns on the bottom right in Table 5, respectively.

From Table 5, there have the following observations. First, COSA achieves the best performance on all eight datasets except the Actor dataset where COSA performs on par with the recent method H2GCN. As a result, COSA achieves the highest F-rank value, and significantly outperforms all the competing methods at the 95% confidence level in both Micro-F1 and Macro-F1. Based on the complexity indicator α in Table 3, most of these datasets contain complex graph structure information, e.g., weak homophily relation as indicated by a small α such as Squirrel, or mixed of homophily and disassortativity as indicated by a middle α, such as UAI. General GNNs perform well on homophily situations, but they work ineffectively on the graphs with weak homophily. The consistent superiority of the proposed model in this research across the datasets demonstrates its capability in modeling those complex graph structures.

Second, the improvement of COSA is particularly significant on some strong homophily datasets, such as MS and Disease, where the values of α of these datasets are larger, indicating the presence of a stronger homophily graph structure in these datasets. General GNNs perform well on homophily situations, but they work even better on the graphs with strong homophily, too. Jointing the different aggregation embeddings in CSA helps to capture the associations and similarities among the adjacent nodes, resulting in a good performance on these two graph datasets that have strong homophily relation.

Link prediction. In this task, use 85/5/10 percent edge splits for training, validation, and testing on all datasets (Chami et al., 2019). AUC and AP are adopted as the evaluation criteria. The results are given in Table 6.

From Table 6, it can observe that COSA achieves the highest F-rank value and significantly outperforms all the competing methods at the 95% confidence level in both AUC and AP results, except SAGE and HOG on AP results; and compared with SAGE and HOG, COSA obtains generally better performance on most of the datasets. Although the improvement is not statistically significant over them, it recall that COSA significantly outperforms it at the 99% confidence level in the node classification task, as shown in Table 5. These link prediction results reinforce the effectiveness and the significance of the method proposed in this research.

Particularly, COSA achieves the best performance on the Chameleon, UAI, USA, MS, and Disease datasets, but it is less effective than the competing methods HOG, FAGCN, and SGC on the other datasets, such as ACM, Actor, and Squirrel in this task. This may be due to the fact that

Table 5
Micro-F1 and Macro-F1 scores (%) for node classification, including F-rank scores of Friedman test and *p*-value scores of Wilcoxon signed-ranks test.

Method	Squirrel (α:0.21)		Actor (α:0.23)		Chameleon (α:0.27)		UAI (α:0.44)		USA (α:0.55)	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
ChebNet	$44.03_{\pm 1.05}$	$40.15_{\pm 0.77}$	$34.03_{\pm 2.15}$	$30.17_{\pm 0.98}$	$55.01_{\pm 0.80}$	$56.90_{\pm 0.72}$	$50.03_{\pm 1.05}$	$40.15_{\pm 0.75}$	$53.87_{\pm 1.09}$	$52.89_{\pm 1.64}$
GCN	$36.03_{\pm 1.45}$	$35.98_{\pm 1.35}$	$30.05_{\pm 1.07}$	$30.22_{\pm 0.77}$	$59.10_{\pm 1.10}$	$58.57_{\pm 0.53}$	$49.03_{\pm 1.36}$	$40.91_{\pm 1.21}$	$54.97_{\pm 3.16}$	$55.82_{\pm 3.06}$
SAGE	$41.20_{\pm 2.00}$	$40.75_{\pm 1.88}$	$34.13_{\pm 1.25}$	$30.15_{\pm 0.58}$	$53.72_{\pm 2.50}$	$51.81_{\pm 3.1}$	$49.90_{\pm 1.05}$	$41.01_{\pm 0.92}$	$55.73_{\pm 0.9}$	$55.97_{\pm 1.25}$
SGC	$44.90_{\pm 1.50}$	$45.11_{\pm 2.00}$	$34.23_{\pm 1.05}$	$31.55_{\pm 1.12}$	$59.98_{\pm 0.97}$	$58.89_{\pm 1.05}$	$53.83_{\pm 0.95}$	$43.05_{\pm 0.77}$	$56.50_{\pm 1.57}$	$56.93_{\pm 1.67}$
H2GCN	$41.79_{\pm 1.08}$	$40.87_{\pm 1.01}$	$\mathbf{36.06}_{\pm 1.53}$	$29.87_{\pm 0.36}$	$62.66_{\pm 0.31}$	$62.31_{\pm 0.23}$	$23.45_{\pm 0.25}$	$10.91_{\pm 0.89}$	$51.87_{\pm 0.19}$	$50.28_{\pm 0.17}$
FAGCN	$45.98_{\pm 0.82}$	$45.61_{\pm 0.61}$	$35.83_{\pm 2.15}$	$33.01_{\pm 2.71}$	$64.27_{\pm 0.68}$	$62.65_{\pm 0.55}$	$59.12_{\pm 1.07}$	$43.25_{\pm 0.72}$	$55.60_{\pm 0.64}$	$54.85_{\pm 0.50}$
SimPGCN	$39.10_{\pm 0.51}$	$38.67_{\pm 0.95}$	$34.08_{\pm 1.66}$	$32.35_{\pm 0.99}$	$59.78_{\pm 2.26}$	$57.33_{\pm 3.71}$	$54.05_{\pm 0.62}$	$41.33_{\pm 0.67}$	$53.03_{\pm 0.69}$	$51.94_{\pm 0.88}$
HOG	$38.60_{\pm 0.10}$	$39.15_{\pm 0.27}$	$35.03_{\pm 1.05}$	$32.65_{\pm 1.27}$	$38.25_{\pm 0.45}$	$36.87_{\pm 0.13}$	$64.10_{\pm 2.01}$	$43.71_{\pm 0.41}$	$51.03_{\pm 1.01}$	$50.38_{\pm 0.14}$
COSA	$\mathbf{46.19}_{\pm 2.58}$	$\mathbf{45.93}_{\pm 3.57}$	$35.92_{\pm 1.88}$	$\mathbf{33.14}_{\pm 1.27}$	$\mathbf{66.39}_{\pm 1.92}$	$\mathbf{66.32}_{\pm 1.61}$	$\mathbf{68.62}_{\pm 2.99}$	$\mathbf{46.19}_{\pm 1.59}$	$\mathbf{56.92}_{\pm 0.68}$	$\mathbf{57.18}_{\pm 0.57}$

Method	**MS (α:0.83)**		**Disease (α:0.88)**		**ACM (α:0.89)**		**Micro-F1**		**Macro-F1**	
	Micro-F1	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**	**F-rank***	***p*-value #**	**F-rank ***	***p*-value #**
ChebNet	$88.83_{\pm 0.31}$	$87.99_{\pm 1.11}$	$76.10_{\pm 5.12}$	$63.45_{\pm 3.55}$	$79.53_{\pm 0.80}$	$34.86_{\pm 3.10}$	2.87	0.003	2.87	0.003
GCN	$92.17_{\pm 0.47}$	$90.40_{\pm 1.90}$	$78.30_{\pm 8.70}$	$67.87_{\pm 4.09}$	$87.80_{\pm 2.10}$	$40.82_{\pm 1.65}$	3.12	0.003	4.25	0.003
SAGE	$91.53_{\pm 0.63}$	$89.57_{\pm 0.65}$	$82.95_{\pm 0.55}$	$70.08_{\pm 1.33}$	$81.80_{\pm 2.38}$	$39.12_{\pm 1.58}$	3.87	0.003	4.12	0.003
SGC	$93.40_{\pm 0.51}$	$92.42_{\pm 0.99}$	$78.97_{\pm 4.21}$	$70.44_{\pm 2.07}$	$88.36_{\pm 4.97}$	$42.95_{\pm 2.18}$	6.25	0.003	6.75	0.003
H2GCN	$92.59_{\pm 0.55}$	$92.77_{\pm 1.62}$	$86.23_{\pm 0.05}$	$72.98_{\pm 0.08}$	$87.20_{\pm 0.04}$	$31.05_{\pm 0.08}$	5.25	0.007	4.25	0.003
FAGCN	$92.20_{\pm 0.08}$	$90.95_{\pm 0.26}$	$78.39_{\pm 8.56}$	$65.25_{\pm 8.1}$	$45.20_{\pm 3.55}$	$20.86_{\pm 1.45}$	5.62	0.003	5.62	0.003
SimPGCN	$91.93_{\pm 1.59}$	$90.67_{\pm 0.58}$	$91.57_{\pm 3.92}$	$47.80_{\pm 2.17}$	$80.20_{\pm 5.75}$	$31.32_{\pm 1.46}$	4.12	0.003	3.60	0.003
HOG	$92.30_{\pm 0.30}$	$90.96_{\pm 0.84}$	$92.05_{\pm 0.47}$	$67.27_{\pm 2.66}$	$89.03_{\pm 5.69}$	$39.95_{\pm 0.88}$	5.0	0.003	4.62	0.003
COSA	$\mathbf{93.61}_{\pm 0.51}$	$\mathbf{93.23}_{\pm 0.88}$	$\mathbf{93.00}_{\pm 2.69}$	$\mathbf{76.11}_{\pm 3.91}$	$\mathbf{90.11}_{\pm 3.90}$	$\mathbf{43.81}_{\pm 3.10}$	**8.87**	/	**9.0**	/

* A higher F-rank value indicates a higher result.
\# The accepted hypotheses with a significance level of 0.05 are highlighted.

Table 6
AUC and AP scores (%) for link prediction, including F-rank scores of Friedman test and *p*-value scores of Wilcoxon signed-ranks test.

Method	**Squirrel (α:0.21)**		**Actor (α:0.23)**		**Chameleon(α:0.27)**		**UAI (α:0.44)**		**USA (α:0.55)**	
	AUC	**AP**	**AUC**	**AP**	**AUC**	**AP**	**AUC**	**AP**	**AUC**	**AP**
ChebNet	$61.92_{\pm 0.75}$	$60.27_{\pm 1.29}$	$72.30_{\pm 0.85}$	$73.91_{\pm 0.21}$	$81.22_{\pm 0.48}$	$84.78_{\pm 0.46}$	$70.55_{\pm 1.71}$	$70.39_{\pm 1.95}$	$86.10_{\pm 0.89}$	$86.82_{\pm 1.05}$
GCN	$93.02_{\pm 0.13}$	$91.56_{\pm 0.18}$	$72.50_{\pm 0.85}$	$74.03_{\pm 0.91}$	$82.87_{\pm 0.70}$	$83.61_{\pm 0.46}$	$70.01_{\pm 0.75}$	$71.27_{\pm 1.29}$	$85.90_{\pm 1.12}$	$82.75_{\pm 1.35}$
SAGE	$84.71_{\pm 0.29}$	$78.89_{\pm 2.36}$	$75.13_{\pm 0.11}$	$77.51_{\pm 0.35}$	$83.73_{\pm 0.36}$	$86.35_{\pm 0.39}$	$78.75_{\pm 1.81}$	$80.13_{\pm 2.11}$	$86.63_{\pm 0.95}$	$87.17_{\pm 0.75}$
SGC	$\mathbf{93.09}_{\pm 0.15}$	$91.49_{\pm 0.23}$	$72.02_{\pm 0.76}$	$74.67_{\pm 0.82}$	$81.83_{\pm 1.42}$	$83.14_{\pm 0.33}$	$80.12_{\pm 2.16}$	$81.51_{\pm 2.62}$	$79.21_{\pm 2.48}$	$76.76_{\pm 2.19}$
H2GCN	$90.36_{\pm 1.62}$	$90.69_{\pm 1.88}$	$70.32_{\pm 1.60}$	$70.65_{\pm 1.62}$	$82.02_{\pm 1.10}$	$83.51_{\pm 2.31}$	$81.56_{\pm 3.77}$	$81.72_{\pm 3.71}$	$80.55_{\pm 3.11}$	$81.71_{\pm 2.72}$
FAGCN	$88.80_{\pm 0.79}$	$87.80_{\pm 1.01}$	$75.88_{\pm 1.11}$	$\mathbf{77.38}_{\pm 1.25}$	$76.03_{\pm 7.31}$	$79.14_{\pm 7.36}$	$71.06_{\pm 1.61}$	$75.29_{\pm 1.56}$	$85.80_{\pm 0.70}$	$85.37_{\pm 1.12}$
SimPGCN	$88.54_{\pm 0.83}$	$84.97_{\pm 1.52}$	$73.92_{\pm 1.32}$	$76.82_{\pm 1.33}$	$76.63_{\pm 0.12}$	$78.55_{\pm 0.24}$	$53.43_{\pm 6.42}$	$53.64_{\pm 7.67}$	$86.53_{\pm 0.60}$	$85.79_{\pm 0.84}$
HOG	$89.37_{\pm 2.25}$	$88.92_{\pm 2.83}$	$70.36_{\pm 1.60}$	$72.15_{\pm 1.77}$	$79.66_{\pm 1.62}$	$80.36_{\pm 1.77}$	$80.59_{\pm 1.60}$	$81.51_{\pm 1.75}$	$87.12_{\pm 2.61}$	$88.52_{\pm 2.62}$
COSA	$92.51_{\pm 3.59}$	$\mathbf{91.70}_{\pm 2.99}$	$\mathbf{78.73}_{\pm 3.15}$	$76.28_{\pm 4.07}$	$\mathbf{85.70}_{\pm 1.95}$	$\mathbf{86.50}_{\pm 1.15}$	$\mathbf{87.65}_{\pm 1.65}$	$\mathbf{87.82}_{\pm 1.36}$	$\mathbf{90.85}_{\pm 3.99}$	$\mathbf{90.63}_{\pm 4.51}$

Method	**MS (α:0.83)**		**Disease (α:0.88)**		**ACM (α:0.89)**		**AUC**		**AP**	
	AUC	**AP**	**AUC**	**AP**	**AUC**	**AP**	**F-rank***	***p*-value #**	**F-rank ***	***p*-value #**
ChebNet	$64.34_{\pm 0.86}$	$64.85_{\pm 0.81}$	$89.22_{\pm 1.66}$	$86.92_{\pm 2.94}$	$59.23_{\pm 0.37}$	$58.41_{\pm 0.82}$	3.37	0.003	3.75	0.003
GCN	$77.58_{\pm 0.61}$	$76.87_{\pm 0.71}$	$87.29_{\pm 1.09}$	$84.58_{\pm 2.06}$	$84.27_{\pm 1.21}$	$83.67_{\pm 1.49}$	5.12	0.007	5.0	0.003
SAGE	$63.58_{\pm 2.42}$	$61.39_{\pm 2.82}$	$75.05_{\pm 7.78}$	$73.62_{\pm 3.95}$	$74.94_{\pm 0.30}$	$70.03_{\pm 1.42}$	4.50	0.003	4.75	0.011
SGC	$77.95_{\pm 0.59}$	$77.79_{\pm 0.68}$	$87.88_{\pm 1.15}$	$84.03_{\pm 2.08}$	$77.39_{\pm 1.67}$	$77.07_{\pm 1.49}$	4.87	0.007	4.68	0.003
H2GCN	$82.51_{\pm 6.76}$	$81.33_{\pm 6.51}$	$88.66_{\pm 3.51}$	$81.31_{\pm 3.76}$	$81.66_{\pm 3.11}$	$85.71_{\pm 3.76}$	5.25	0.003	4.87	0.003
FAGCN	$74.07_{\pm 1.12}$	$75.43_{\pm 0.98}$	$84.71_{\pm 4.0}$	$84.11_{\pm 4.8}$	$62.02_{\pm 1.95}$	$59.72_{\pm 1.25}$	3.50	0.003	4.0	0.007
SimPGCN	$81.41_{\pm 6.76}$	$79.68_{\pm 6.75}$	$50.00_{\pm 0.00}$	$50.00_{\pm 0.00}$	$72.98_{\pm 3.03}$	$66.31_{\pm 2.85}$	3.50	0.003	3.37	0.007
HOG	$83.56_{\pm 3.58}$	$81.51_{\pm 3.76}$	$89.35_{\pm 2.76}$	$85.66_{\pm 3.71}$	$\mathbf{89.31}_{\pm 6.76}$	$87.71_{\pm 6.76}$	6.25	0.007	6.06	0.011
COSA	$\mathbf{85.11}_{\pm 2.73}$	$\mathbf{84.12}_{\pm 1.71}$	$\mathbf{90.57}_{\pm 1.01}$	$\mathbf{87.45}_{\pm 2.31}$	$88.16_{\pm 0.75}$	$\mathbf{85.80}_{\pm 0.77}$	**8.62**	/	**8.5**	/

* A higher F-rank value indicates a higher result.
\# The accepted hypotheses with a significance level of 0.05 are highlighted.

self-embedding and neighborhood embedding similarities learned by contrastive learning are not as helpful in link prediction as that in node classification.

Comparison of computational efficiency. Finally, the authors compare the computational efficiency among all the models, and the results are presented in Fig. 3 (some results are truncated for the sake of readability). COSA often runs slower than other GNN models, such as ChebNet, GCN, SAGE, SGC, and SimPGCN, as it is enforced to model both separation and aggregation embeddings. However, it is more efficient than GNN models that capture similarly complex graph structures, such as H2GCN.

5.3. Ablation experiment (RQ2)

To evaluate the influences of each component, the authors conduct the ablation experiment. To this end, the authors design three variants of COSA, i.e., COSA0, COSA1, and COSA2. Specifically, COSA0 (COSA1) is strictly focused on learning separated (aggregated) embeddings only, while COSA2 learns both separated and aggregated representations. Finally, by adding the Struc-Adapter module to COSA2, obtain our full model COSA. The authors compare the three variants with COSA in the task of node classification. Table 7 presents the results (%) of Micro-F1 when the settings are as usual.

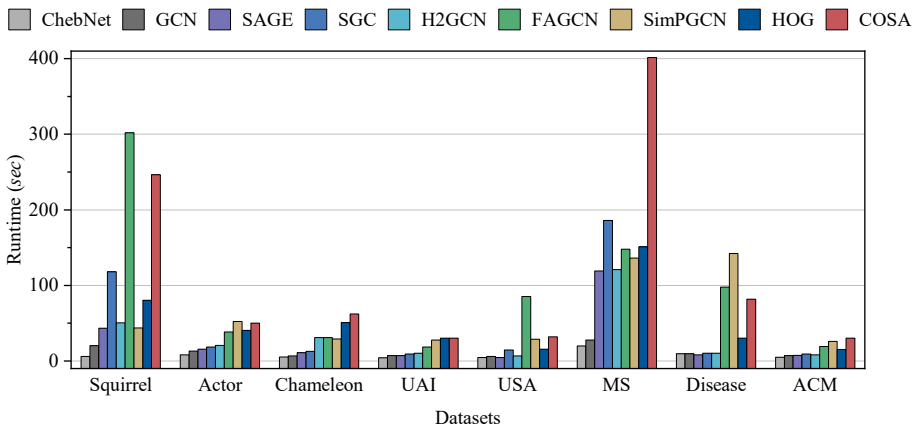


Fig. 3. Runtime on different datasets. Note that, for the sake of readability, the results that reach the top *y*-axis are truncated.

Table 7
Micro-F1 results (%) for ablation study, including F-rank scores of Friedman test and *p*-value scores of Wilcoxon signed-ranks test.

	COSA0	COSA1	COSA2	COSA
Separation	√		√	√
Aggregation		√	√	√
Struc-Adapter				√
Squirrel	41.55	39.98	43.62	**46.21**
Actor	28.33	27.63	33.37	**35.91**
Chameleon	58.41	54.56	62.51	**66.42**
UAI	63.91	63.52	66.11	**68.6**
USA	52.13	53.61	55.25	**56.92**
MS	87.99	90.02	92.33	**93.65**
Disease	88.21	89.55	91.62	**93.01**
ACM	86.36	87.12	89.51	**90.1**
F-rank*	1.50	1.50	3.0	**4.0**
p-value#	0.003	0.003	0.003	/

* A higher F-rank value indicates a higher result.
\# The accepted hypotheses with a significance level of 0.05 are highlighted.

It can see from the results that both COSA0 and COSA1 perform fairly well across the eight datasets, indicating the presence of both assortativity and disassortativity (the former is captured by COSA0 while the latter is captured by COSA1). The aggregated representation learner COSA1 performs better than the separated representation learner COSA0 on the datasets with large homophily (the value of α is larger), while COSA0 performs better than COSA1 on the datasets with smaller homophily (the value of α is smaller), indicating the importance of the aggregated representation learning on datasets with assortativity structure, and separated representation learning on datasets with disassortativity. More importantly, the joint separated and aggregated representation learner COSA2 consistently outperforms both COSA0 and COSA1. These results also demonstrate the effectiveness of the model in synthesizing both types of representations. Lastly, COSA outperforms COSA2 on all the datasets, showing the consistently positive contribution of the optimal customized neighbor space learning component Struc-Adapter in COSA.

5.4. Hyper-parameter analysis (RQ3)

In this section, the authors investigate the sensitivity of the performance of COSA w.r.t. three key hyper-parameters, including the dimension of nodes embeddings r, the hidden size h_s, and the generation τ. The experiments are performed on the Disease and Chameleon datasets using the node classification task. Similar observations can be found in other datasets. In this set of experiments, the authors vary one of the tested hyper-parameters, with the other ones set as default values. The authors vary r in {8, 16, 32, 64, 128, 256, 512}, h_s in {50, 100, 200, 300, 400, 500, 1000}, and τ in {1, 2, 3, 4, 5, 10}. The settings are as usual, and the test results are reported in Fig. 4.

It can see that: a) the Micro-F1 scores in Fig. 4(a-b) roughly increase firstly and then decrease, which is probably because adding dimensions appropriately can carry more useful distinguish information while too high dimension would introduce noisy information to the representation space; b) the hidden size h_s controls the capacity of the hidden layer, with the changing trend of the Micro-F1 scores in Fig. 4(c-d) swings back and forth at the beginning and then gradually becomes saturated at around $h_s = 500$, and so too large h_s does not help enhance the performance of the model; and c) as for the generation τ, the trend of the Micro-F1 scores in Fig. 4(e-f) is relatively stable, and the values of τ cannot influence the performance of the model. Overall, the Micro-F1 scores are stable within certain ranges of settings of the key hyper-parameters, showing that COSA has a relatively robust performance regarding its hyper-parameters, and so they can be well-tuned using the validation dataset in practice.

In addition, to analyze the convergence of COSA intuitively, Fig. 5 presents the changes of loss. As illustrated in Fig. 5, we find that COSA converges quickly demonstrating the good stability of COSA during training.

6. Conclusion

This paper proposes to jointly the separated and the optimal friendly aggregated information. This enables the learning of expressive representations of complex graph structures with disassortativity, for which current state-of-the-art GNN approaches fail to do so. To this end, we introduce a novel graph neural network approach called COSA that learns the intrinsic interrelationship and the extra supervision information between the optimal customized aggregated information and the separated information originating from self and neighborhood, while at the same time pursues the optimal friendly neighbor space and integrates the variational inference to train the whole framework for exploring the approximated posterior. The efficacy of COSA and its modules is well supported by extensive empirical results on eight real-world benchmark datasets using two main downstream graph learning tasks, including node classification and link prediction.

The structure and properties of real-world networks are often complex, and there may have class-imbalance problem. For example, in the fraud detection task, since the quantity of the fraudulent data is relatively rare, the class-imbalance problem is especially obvious. The solution for this problem in the disassortative networks will be explored in the future work.

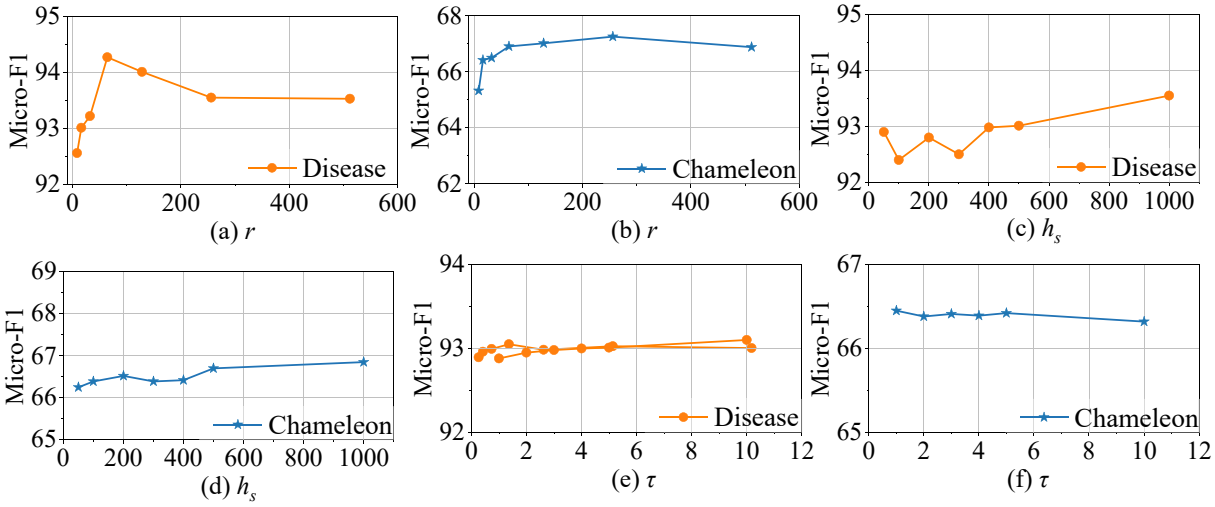


Fig. 4. Micro-F1 results w.r.t. different settings of three hyper-parameters.

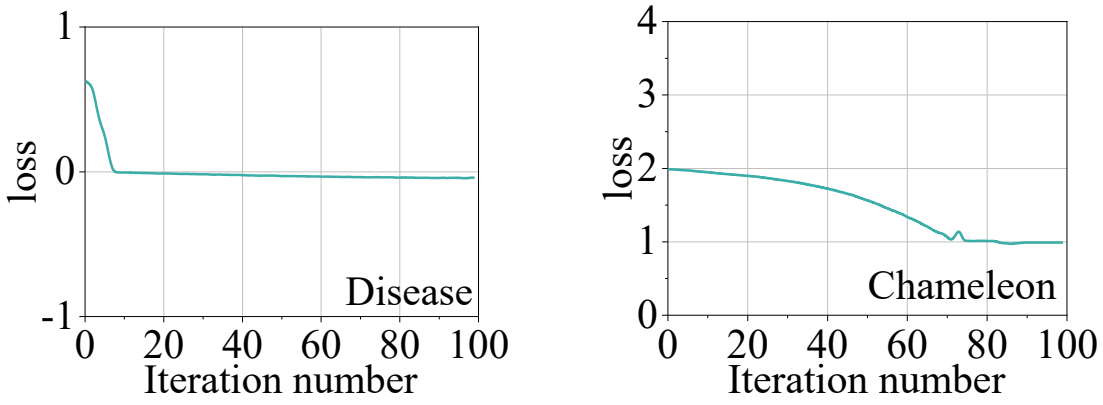


Fig. 5. Changes of loss as iteration number increases on Disease and Chameleon datasets.

CRediT authorship contribution statement

Xiaoyu Xu: Investigation, Conceptualization, Methodology, Formal analysis, Data curation, Writing – original draft, Writing – review & editing, Visualization, Project administration. **Xiaoyu Shi:** Conceptualization, Methodology, Formal analysis, Writing – review & editing, Funding acquisition. **Mingsheng Shang:** Conceptualization, Methodology, Resources, Supervision, Writing – review & editing, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China under Grant 62072429, in part by the Key Cooperation Project of Chongqing Municipal Education Commission under Grant HZ2021017 and Grant HZ2021008, and the "Fertilizer Robot" project of Chongqing Committee on Agriculture and Rural Affairs.

References

Belciug, S. (2022). Learning deep neural networks' architectures using differential evolution. Case study: Medical imaging processing. *Computers in Biology and Medicine 146*, article.105623.

Bilal, Pant, M., Zaheer, H., Garcia-Hernandez, L., & Abraham, A. (2020). Differential evolution: A review of more than two decades of research. *Engineering Applications of Artificial Intelligence, 90*, 1012–1036.

Bo, D., Wang, X., Shi, C., & Shen, H. (2021). Beyond low-frequency information in graph convolutional networks. In *Proceedings of the 35th AAAI* (pp. 3950–3957).

Cai, H., Zheng, V. W., & Chang, K.-C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge & Data Engineering, 30*(09), 1616–1637.

Chami, I., Ying, R., Re, C., & Leskovec, J. (2019). Hyperbolic graph convolutional neural networks. In *Processing of the 33rd NIPS*, Vancouver, CANADA. pp. 4869–4880.

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Processing of the 30th NIPS*, Barcelona, SPAIN. pp. 3837–3845.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

Dornaika, F. (2022). On the use of high-order feature propagation in graph convolution networks with manifold regularization. *Information Sciences, 584*, 467–478.

Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems, 151*, 78–94.

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Proceedings of the 31st NIPS*, Long Beach, CA. pp. 1024–1034.

He, D., Liang, C., Liu, H., Wen, M., Jiao, P., & Feng, Z. (2021). Block modeling-guided graph convolutional neural networks. In *Proceedings of the 36th AAAI* (pp. 1–8).

Jin, W., Derr, T., Wang, Y., Ma, Y., Liu, Z., & Tang, J. (2021). Node similarity preserving graph convolutional networks. In *Proceedings of the 21th WSDM* (pp. 148–156).

Jovanovic, L., Jovanovic, D., Bacanin, N., Jovancai Stakic, A., Antonijevic, M., Magd, H., … Zivkovic, M. (2022). Multi-step crude oil price prediction based on lstm approach tuned by salp swarm algorithm with disputation operator. *Sustainability 14*(21), article.14616.

Kim, D., & Oh, A. H. (2021). How to find your friendly neighborhood: Graph attention design with self-supervision. In *Proceedings of the 9th ICLR* (pp. 1–25).

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th ICLR* (pp. 1–14).

Neal, R. M., & Hinton, G. E. (1998). A view of the em algorithm that justifies incremental, sparse, and other variants. *Learning in Graphical Models*, 355–368.

Pei, H., Wei, B., Chang, K.-C.-C., Lei, Y., & Yang, B.-G. (2020). Geometric graph convolutional networks. In *Proceedings of the 8th ICLR* (pp. 1–10).

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th SIGKDD*, New York, USA. pp. 701–710.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks, 20*(1), 61–80.

Storn, R., & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*(4), 341–359.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio', P., & Bengio, Y. (2018). Graph attention networks. In *Proceedings of the 6th ICLR* (pp. 1–12).

Wan, S., Pan, S., Yang, J., & Gong, C. (2021). Contrastive and generative graph convolutional networks for graph-based semi-supervised learning. In *Proceedings of the 35th AAAI* (pp. 10049–10057).

Wang, X., Zhu, M., Bo, D., Cui, P., Shi, C., & Pei, J. (2020). Am-gcn: Adaptive multi-channel graph convolutional networks. In *Proceedings of the 26th SIGKDD*, Virtual Event, CA, USA. pp. 1243–1253.

Wang, R., Mou, S., Wang, X., Xiao, W., Ju, Q., Shi, C., & Xie, X. (2021). Graph structure estimation neural networks. In *Proceedings of the 30th WWW*, Ljubljana, Slovenia. pp. 342–353.

Wang, J., Liang, J., Cui, J., & Liang, J. (2021). Semi-supervised learning with mixed-order graph convolutional networks. *Information Sciences, 573*, 171–181.

Wang, T., Wang, R., Jin, D., He, D., & Huang, Y. (2022). Powerful graph convolutioal networks with adaptive propagation mechanism for homophily and heterophily. In *Proceedings of the 36th AAAI* (pp. 1–9).

Wu, D., He, Q., Luo, X., Shang, M., He, Y., & Wang, G. (2022). A posterior-neighborhood-regularized latent factor model for highly accurate web service qos prediction. *IEEE Transactions on Services Computing, 15*(2), 793–805.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems, 32*(1), 1–21.

Wu, F., Jr., Souza, A. H., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. Q. (2019). Simplifying graph convolutional networks. In *Proceedings of the 36th ICML* (pp. 6861–6871).

Xia, J., Wu, L., Wang, G., Chen, J., & Li, S. Z. (2022). Progcl: Rethinking hard negative mining in graph contrastive learning. In *Proceedings of the 39th ICML* (pp. 1–11).

Yu, J., Yin, H., Xia, X., Chen, T., Cui, L., & Nguyen, Q. V. H. (2022). Are graph augmentations necessary? Simple graph contrastive learning for recommendation. In *Proceedings of the 28th SIGKDD* (pp. 1–10).

Zhang, Y., Wang, X., Shi, C., Liu, N., & Song, G. (2021). Lorentzian graph convolutional networks. In *Proceedings of the 30th WWW*, Ljubljana, Slovenia. pp. 1249–1261.

Zhang, D., Yin, J., Zhu, X., & Zhang, C. (2020). Network representation learning: A survey. *IEEE Transactions on Big Data, 6*(1), 3–28.

Zhu, Y., Guo, J., Wu, F., & Tang, S. R. (2022). A robust self-aligned framework for node-node graph contrastive learning. In *Proceedings of the 31th IJCAI* (pp. 1–12).

Zhu, J., Rossi, R. A., Rao, A., Mai, T., Lipka, N., Ahmed, N. K., & Koutra, D. (2021). Graph neural networks with heterophily. In *Proceedings of 35th AAAI* (pp. 11168–11176).

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., & Wang, L. (2021). Graph contrastive learning with adaptive augmentation. In *Proceedings of the 21th WWW* (pp. 2069–2080).

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., & Koutra, D. (2020). Beyond homophily in graph neural networks: Current limitations and effective designs. In *Proceedings of the 33rd NIPS* (pp. 1–12).