Siamese Attribute-missing Graph Auto-encoder

Journal:	*Transactions on Pattern Analysis and Machine Intelligence*
Manuscript ID	TPAMI-2022-01-0177
Manuscript Type:	Regular
Keywords:	I.2.6.g Machine learning < I.2.6 Learning < I.2 Artificial Intelligence < I Computing Methodologies, I.5.1 Models < I.5 Pattern Recognition < I Computing Methodologies

Siamese Attribute-missing Graph Auto-encoder

Wenxuan Tu†, Sihang Zhou†, Xinwang Liu*, *Senior Member, IEEE*, Yue Liu, and Zhiping Cai,

Abstract—Graph representation learning (GRL) on attribute-missing graphs, a common yet challenging problem, has recently attracted considerable attention. We observe that existing literature: 1) isolates the embedding learning of node attributes and graph structures thus fails to take full advantage of the two types of information; 2) imposes too strict a distribution assumption on the latent space variables, leading to less discriminative feature representations. In this paper, based on the idea of introducing intimate information interaction between the two information sources, we propose our Siamese Attribute-missing Graph Auto-encoder (SAGA) to boost the expressive capacity of graph representations for high-quality missing attribute restoration. Specifically, three strategies have been conducted. First, we entangle the attribute embedding and structure embedding by introducing a Siamese network structure to share the parameters learned by both processes, which allows the network training to benefit from more abundant and diverse information. Second, we introduce a K-nearest neighbor (KNN) and structural constraint enhanced learning mechanism to improve the quality of latent features of the missing attributes by filtering unreliable similarities. Third, we manually mask the connections on multiple adjacent matrices and force the structural information embedding sub-network to recover the actual adjacent matrix, thus enforcing the resulting network to be able to selectively exploit more high-order discriminative features for data completion. Extensive experiments on six benchmark datasets demonstrate the superiority of our SAGA against the state-of-the-art methods.

Index Terms—graph representation learning, graph neural network, attribute-missing, Siamese network.

1 Introduction

Graph representation learning (GRL), which aims to learn a graph neural network (GNN) that embeds nodes to a low-dimensional latent space by preserving node attributes and graph structures, has been intensively studied and widely applied into various applications [1], [2], [3], [4], [5], [6]. One underlying assumption commonly adopted by these methods is that all attributes of nodes are complete. However, in practice, this assumption may not hold due to 1) the absence of particular attributes; 2) the absence of all the attributes of specific nodes. These circumstances are usually called attribute incomplete [7] and attribute missing [8], respectively. The existence of the above circumstances makes existing GRL methods unable to effectively handle corresponding learning problems.

To solve the first type of problem, the early methods mainly concentrate on imputation techniques for data completion, such as matrix completion via matrix factorization [9], [10], [11], [12] and generative adversarial learning [13]. Finding that the imputation-based methods disconnect the learning processes of imputation and network optimization, which decreases the diversity and discriminability of the learned representations, some advanced algorithms, e.g., GRAPE [14] and GCNMF [7], merge both processes of data imputation and representation learning into a united graph convolutional network (GCN) [15]-based framework, where they adopt bipartite message passing strategy and Gaussian mixture model (GMM) to restore incomplete values, respectively. The above-mentioned methods could work well when handling attribute-incomplete problems. Nevertheless, they could be hard to produce high-quality data completion when node attributes are entirely missing.

Compared to the first type of methods, the second category aims to tackle a newly proposed problem, which has not been sufficiently studied in the literature and remains an open yet challenging issue. To tackle this issue, SAT [8], makes the first attempt to guide the generation of more meaningful latent embedding by introducing a distribution consistency assumption between attribute and structure embedding sub-networks. Though demonstrating the high quality of attribute restoration in various downstream tasks, SAT suffers from the following non-negligible limitations: 1) adopts two decoupled sub-networks for information extraction, thus isolates the learning of attribute embedding and structure embedding; 2) imposes too strict a distribution assumption on the latent variables, which essentially decreases the discriminative capability of the learned representations; 3) it lacks a structure-attribute information filtering and refining mechanism for data completion, resulting in less robust feature representations.

Motivated by the above observations, we propose a novel graph representation learning network termed **S**iamese **A**ttribute-missing **G**raph **A**uto-encoder (SAGA) to handle attribute-missing graphs, as illustrated in Fig. 1. The core idea of our method is to establish a structure-attribute mutual enhanced learning strategy to 1) allow sufficient interaction between the attribute-missing matrix and the adjacent matrix for information filtering; 2) introduce a hidden structure refinement strategy for high-quality data completion. To facilitate the above ideas, we entangle the attribute information embedding and structural information embedding by introducing a Siamese framework to share the parameters learned by the two processes. Then, we design

W. Tu, X. Liu, Y. Liu, and Z. Cai, are with the College of Computer, National University of Defense Technology, Changsha 410073, China (e-mail: {wenxuantu, yueliu19990731}@163.com, {xinwangliu, zpcai}@nudt.edu.cn).
S. Zhou is with the College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China (e-mail: sihangjoe@gmail.com).
† *Equal contribution.*
* *Corresponding author.*

a dual non-local aggregating (DNA) module to filter unreliable similarities by utilizing K-nearest neighbor (KNN) and structural constraint enhanced learning mechanism. With this mechanism, each node in the latent space could well collect and preserve the most informative information from non-local features. This boosts the quality of latent embedding of the restored attributes. Moreover, to enhance the quality of structural information of attribute-missing nodes, we propose a hidden structure refining (HSR) module. In this module, we manually mask the connections on multiple adjacent matrices and force the structure information embedding sub-network to recover the actual adjacent matrix. As an auxiliary task, the HSR module enables the network to pull closer the representations of neighbor nodes to refine their structure information, especially for that of attribute-missing nodes. This in turn enforces the resulting network to be able to selectively exploit more discriminative features from hidden high-order attributes for data completion. Finally, by iteratively optimizing the two-source information embeddings, the DNA module, and the HSR module on the end-to-end training process, these three parts can benefit from each other to boost the discriminative capability of the graph embedding, thus leading to higher-quality data completion. The main contributions of this paper are listed as follows:

- We propose a novel graph representation learning framework termed as **S**iamese **A**ttribute-missing **G**raph **A**uto-encoder (SAGA) to solve a newly proposed problem, i.e., unsupervised graph representation learning on attribute-missing graphs, which allows us to elegantly achieve powerful data completion without any prior assumption.
- By promoting the learning processes of attribute and structure information to sufficiently interact with each other via the DNA module and the HSR module, we can take full advantage of two-source information to filter unreliable similarities as well as preserve more informative structure-attribute information in the latent space. In this way, the discriminative capacity of resultant latent embedding is improved for better missing attribute restoration.
- Extensive experimental results on six benchmark datasets demonstrate that our proposed method is highly competitive and consistently outperforms the state-of-the-art ones with a preferable margin.

The remainder of this paper is organized as follows. Section 2 reviews related works in terms of Siamese networks, graph representation learning, and graph deep learning with absent data. Section 3 presents the model design and each component of SAGA. Section 4 conducts experiments and discusses the results. Finally, section 5 draws a conclusion.

2 Related Work

Siamese Networks. The Siamese network is a kind of network that contains more than one identical weight-sharing sub-networks, which can naturally introduce inductive biases for invariance modeling [16]. It has wide applications including video super-resolution [17], medical object detection [18], visual tracking [19], etc. Inspired by its success in various visual tasks, researchers have successfully introduced Siamese networks into the field of graph learning [20], [21]. However, it has not been extended to handle incomplete or missing graphs.

Graph Representation Learning. Early graph representation learning (GRL) methods learn the graph embedding by utilizing probability models on the generated random walk paths on graphs [22], [23]. However, these methods overly emphasize the structural information while ignoring the important attribute information. Thanks to the development of graph neural networks (GNNs), GNN-based GRL methods that jointly exploit graph structure information and node attribute information in a spectral [24] or spatial [15] domain have been widely studied in recent years. As one of the most representatives, generative/predictive learning-oriented methods explore abundant information embedded in the data itself via various techniques, such as auto-encoder learning and adversarial learning [25], [26], [27], [28], [29], [30], [31], [32]. Another line pays attention to graph contrastive learning, which aims to maximize the agreement of two jointly sampled positive pairs [4], [33], [34], [35], [36]. One underlying assumption commonly adopted by current GRL methods is that all node attributes are complete. However, in practice, the absent data makes it difficult to utilize the existing methods for satisfactory performance.

Graph Deep Learning with Absent Data. To handle incomplete graph data, one popular way commonly adopted by existing algorithms is data imputation technique, such as matrix completion and generative adversarial learning. For matrix completion, sRGCNN [9], GC-MC [10], NMTR [12], IGMC [37] first formulate the user-item rating matrix, users (or items), and the observed ratings as bipartite graph, nodes, and links, respectively. Then they apply a graph neural network to predict the absent linkages between node pairs for data completion in a transductive or inductive manner. Moreover, by introducing an additional adversarial loss, GINN [13] trains a graph denoising auto-encoder to build intermediate representations of all nodes with the help of a pre-processing graph for data completion. To further improve the quality of attribute restoration, recent efforts combine the processes of data imputation and representation learning into a united GNN-based framework. For instance, GRAPE [14] first formulates the feature imputation as an edge-level prediction task on the graph, and then employs a graph neural network to solve it. After that, GCNMF [7] estimates the incomplete variables by enforcing them to follow the Gaussian mixture distribution using a Gaussian mixture model (GMM).

Although these methods are competent to effectively handle the attribute-incomplete problem, they fail to perform preferably on datasets with missing attributes, i.e., the entire attributes of specific nodes are missing. More recently, an advanced algorithm SAT is proposed to learn on attribute-missing graphs. It adopts two decoupled sub-networks to process node attributes and graph structures, and then utilize structure information to conduct data completion under the guidance of a shared-latent space assumption [8]. Unfortunately, despite its success, SAT not only isolates the learning processes of attribute embedding and structure embedding but also heavily relies on a prior

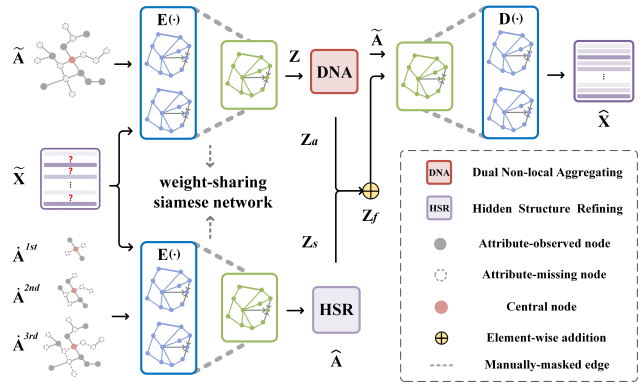

Fig. 1. The overall architecture of SAGA. Our Siamese architecture consists of two branches where the attributes and the affinity matrices are closely entangled. Specially, in the upper branch, the DNA module improves the quality of missing attribute latent feature learning by introducing an unreliable similarity filtering mechanism. While in the bottom branch, the HSR module adopts the multi-order neighbor attentive fusion with a hidden structure recovery strategy to make the network able to exploit intrinsic data structures for information recovery.

TABLE 1
Summary of notations.

Notations	Meaning
$\mathbf{X} \in \mathbb{R}^{N\times D}$	Original attribute matrix
$\mathbf{A} \in \mathbb{R}^{N\times N}$	Original adjacency matrix
$\widetilde{\mathbf{X}} \in \mathbb{R}^{N\times D}$	Initially imputed attribute matrix
$\widetilde{\mathbf{A}} \in \mathbb{R}^{N\times N}$	Normalized adjacency matrix
$\mathbf{A}^{h\text{-}th} \in \mathbb{R}^{N\times N}$	h-th-order adjacency matrix
$\dot{\mathbf{A}}^{h\text{-}th} \in \mathbb{R}^{N\times N}$	Edge-masked h-th-order adjacency matrix
$\mathbf{Z} \in \mathbb{R}^{N\times d}$	Latent embedding matrix
$\mathbf{S} \in \mathbb{R}^{N\times N}$	Affinity matrix
$\mathbf{S}^{\mathcal{N}} \in \mathbb{R}^{N\times N}$	Global-scope indicator matrix
$\mathbf{S}'^{\mathcal{N}} \in \mathbb{R}^{N\times N}$	Multi-order neighbor-scope indicator matrix
$\mathbf{Z}_a \in \mathbb{R}^{N\times d}$	Attribute-enhanced latent embedding matrix
$\mathbf{Z}^{h\text{-}th} \in \mathbb{R}^{N\times d}$	h-th path latent embedding matrix
$\mathbf{C}^{h\text{-}th} \in \mathbb{R}^{d\times N}$	h-th path attention weight matrix
$\mathbf{Z}_s \in \mathbb{R}^{N\times d}$	Structure-enhanced latent embedding matrix
$\mathbf{Z}_f \in \mathbb{R}^{N\times d}$	Fused latent embedding matrix
$\widehat{\mathbf{A}} \in \mathbb{R}^{N\times N}$	Rebuilt adjacency matrix
$\widehat{\mathbf{X}} \in \mathbb{R}^{N\times D}$	Rebuilt attribute matrix

distribution assumption for latent representation learning. Thus it fails to take full advantage of both types of information, leading to less discriminative latent representations. In contrast, our method allows both learned features to sufficiently interact with each other via a structure-attribute mutual enhanced learning strategy. As a result, the learned embedding has the potential to be more discriminative for data completion.

3 The Proposed Method

3.1 Primary Statement

3.1.1 Notations

Given an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with C classes, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$, $\mathcal{E}$, and N are the node set, edge set, and the number of nodes, respectively. In classic graph learning, a graph is usually characterized by its attribute matrix $\mathbf{X} \in \mathbb{R}^{N\times D}$ and normalized adjacency matrix $\widetilde{\mathbf{A}} \in \mathbb{R}^{N\times N}$ [15], where D is the node attribute dimension. Specially, in attribute-missing graph learning, with the existence of missing attributes, we further define $\mathcal{V}^o = \{v_1^o, v_2^o, \ldots, v_{N^o}^o\}$ and $\mathcal{V}^m = \{v_1^m, v_2^m, \ldots, v_{N^m}^m\}$ to be the set of attribute-observed nodes and the set of attribute-missing nodes, respectively. Accordingly, $\mathcal{V} = \mathcal{V}^o \cup \mathcal{V}^m$, $\mathcal{V}^o \cap \mathcal{V}^m = \varnothing$ and $N = N^o + N^m$. In this circumstance, the missing attributes in $\mathbf{X}$ are firstly filled with zero values, random values from a standard Gaussian or observed neighbor values, and the resulting matrix is denoted as $\widetilde{\mathbf{X}} \in \mathbb{R}^{N\times D}$. To introduce high-order adjacent information, we construct a series of adjacent matrices of different orders $\mathcal{A}$ using a random walk-like operation, where $\mathcal{A} = \{\mathbf{A}^{1st}, \mathbf{A}^{2nd}, \ldots, \mathbf{A}^{H\text{-}th}\}$, H is the number of orders. Specially, h-th-order adjacent matrix $\mathbf{A}^{h\text{-}th} \in \mathbb{R}^{N\times N}$ is formulated as:

$$\mathbf{A}^{h\text{-}th} = \mathbf{A}^{1st}\mathbf{A}^{(h-1)\text{-}th}, \tag{1}$$

where $\mathbf{A}^{1st}$ indicates the initial adjacency matrix in original dataset, $\mathbf{A}^{0\text{-}th}$ denotes identity matrix $\mathbf{I} \in \mathbb{R}^{N\times N}$, $1 \le h \le H$. $a_{ij}^{h\text{-}th} = 1$ if node v_i and node v_j are connected, otherwise $a_{ij}^{h\text{-}th} = 0$. It is worth noting that random walk has a drawback, i.e., node revisiting. To alleviate this issue, we first reformulate Eq.(1) as:

$$\mathbf{A}^{h\text{-}th} = \mathbf{A}^{1st}\mathbf{A}^{(h-1)\text{-}th} - diag(\mathbf{A}^{1st}\mathbf{A}^{(h-1)\text{-}th}), \tag{2}$$

where $diag(\mathbf{A}^{1st}\mathbf{A}^{(h-1)\text{-}th})$ denotes the corresponding degree matrix. For $\mathbf{A}^{h\text{-}th}$, we then drop the connections that overlap with the ones in $\mathbf{A}^{(h-1)\text{-}th}, \mathbf{A}^{(h-2)\text{-}th}, \ldots, \mathbf{A}^{1st}$. By doing this, $\mathcal{E}^{h\text{-}th} \cap \mathcal{E}^{(h-1)\text{-}th} \cap \cdots \cap \mathcal{E}^{1st} = \varnothing$, node revisiting issue can be well alleviated. To boost the network learning, we manually mask partial connections (i.e., the linkage relation between attribute-missing nodes) of multi-order adjacent matrices $\mathcal{A}$. After that, $\mathcal{A}$ is redefined as a normalized version $\dot{\mathcal{A}} = \{\dot{\mathbf{A}}^{1st}, \dot{\mathbf{A}}^{2nd}, \ldots, \dot{\mathbf{A}}^{H\text{-}th}\}$, where $\dot{\mathbf{A}}^{h\text{-}th} \in \mathbb{R}^{N\times N}$ refers to the edge-masked h-th-order adjacency matrix. Table 1 summarizes the commonly used notations.

3.1.2 Task Definition

In this paper, we first aim to learn an auto-encoder SAGA without utilizing labeling information. In the encoding phase, given the inputs $\mathcal{G} = \{\widetilde{\mathbf{X}}, \widetilde{\mathbf{A}}\}$ and $\dot{\mathcal{G}}^{h\text{-}th} = \{\widetilde{\mathbf{X}}, \dot{\mathbf{A}}^{h\text{-}th}\}$, our goal is to learn an encoder $E(\cdot)$ that produces graph embedding $\mathbf{Z}_f \in \mathbb{R}^{N\times d}$ such that $d \ll D$, where we have conducted the unreliable similarity filtering and the structure information refining operations for data completion in the latent space. In the decoding phase, the learned graph embedding along with $\widetilde{\mathbf{A}}$ is fed into the graph decoder $D(\cdot)$. The resultant rebuilt attribute matrix $\widehat{\mathbf{X}} \in \mathbb{R}^{N\times D}$ can then be directly employed in downstream tasks, e.g., node classification.

3.2 Overview

This section will introduce and analyze our proposed SAGA framework in detail. Before our work, the existing literature solves the problem of missing attributes with the strategy of latent space alignment [8]. Specifically, they learn the representations of attribute space and structure space independently and conduct information exchange by doing latent space alignment. In contrast, to establish a

structure-attribute mutual enhanced learning strategy, our paper proposes a Siamese network structure where the attributes and the adjacent matrix are closely entangled. As illustrated in Fig. 1, the overall framework mainly consists of two branches. In the upper branch, the encoder $E(\cdot)$ first accepts $\widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{A}}$ as inputs, then outputs $\mathbf{Z}_a \in \mathbb{R}^{N\times d}$ via the proposed dual non-local aggregating (DNA) module (section 3.3.1 for details). The goal of the DNA design is to allow sufficient interaction between the attribute-missing matrix and the indicator matrices in the global and multi-order neighbor spaces for unreliable similarity filtering, as shown in Fig. 2(a). By this means, the network is enabled to discover more hints in the latent space for accurate attribute imputation. While in the bottom branch, the weight-sharing encoder $E(\cdot)$ first accepts $\widetilde{\mathbf{X}}$ and a series of adjacent matrices of different orders $\dot{\mathcal{A}} = \{\dot{\mathbf{A}}^{1st}, \dot{\mathbf{A}}^{2nd}, \dot{\mathbf{A}}^{3rd}\}$, then outputs $\mathbf{Z}_s \in \mathbb{R}^{N\times d}$ and $\widehat{\mathbf{A}} \in \mathbb{R}^{N\times N}$ via the proposed hidden structure refining (HSR) module (section 3.3.2 for details). It is worth noting that HSR is regarded as an auxiliary task to guide the network to automatically exploit complementary information. As shown in Fig. 2(b), by adopting the multi-order neighbor attentive fusion with a hidden structure recovery strategy, the HSR module can make the network able to exploit intrinsic data structures for more accurate latent space construction. After that, by passing the information through the Siamese network and combining the learned structure refinement-oriented latent vectors, the quality of attribute imputation can be further improved. Finally, the fused embedding $\mathbf{Z}_f$ is transferred into the decoder $D(\cdot)$ to tune the model using two reconstruction loss functions for missing attribute restoration $\widehat{\mathbf{X}}$ (section 3.4 and section 3.5 for details). In the following sections, we will provide details on two carefully-designed components and the optimization target, respectively.

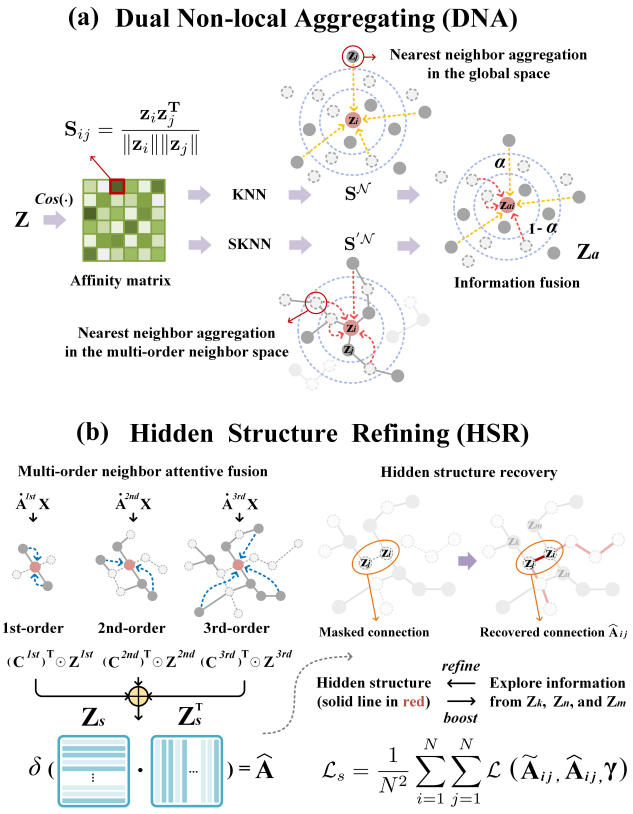

Fig. 2. Illustration of the DNA module and the HSR module. In our work, we propose to solve the attribute-missing problem in a brand new perspective, i.e., 1) the DNA module is designed to aggregate (dotted lines in (a)) more informative features to complement the missing information in the global and multi-order neighbor spaces. By filtering unreliable similarities using indicator matrices (i.e., $\mathbf{S}^{\mathcal{N}}$ and $\mathbf{S}'^{\mathcal{N}}$), the DNA module can help the network to discover more hints for accurate attribute imputation; 2) the HSR module is designed to have the two kinds of information to verify each other by preserving multi-order information and conducting structure information recovery. As an auxiliary task, the HSR module guides the network to automatically exploit complementary information for more accurate latent space construction. By passing the information through the Siamese network and combining two-source latent variables, the quality of attribute imputation can be improved.

3.3 Structure-attribute Mutual Enhancement

In this part, we introduce two proposed components, i.e., the dual non-local aggregating (DNA) module and the hidden structure refining (HSR) module in detail. Both modules are illustrated in Fig. 2. The information of the shared Siamese network will be introduced during the introduction of the two parts.

3.3.1 Dual Non-local Aggregating

For given inputs $\widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{A}}$, a Siamese graph encoder $E(\cdot)$ conducts the following layer-wise propagation to compute l-th latent representations $\mathbf{Z}^{(l)}$, of which i-th row, $\mathbf{z}_i^{(l)}$ denotes the representation for node $v_i \in \mathcal{V}$:

$$\mathbf{Z}^{(l)} = \sigma(\widetilde{\mathbf{A}}\mathbf{Z}^{(l-1)}\mathbf{W}^{(l)}), \tag{3}$$

where $\mathbf{W}^{(l)}$ denotes the learnable parameters of the l-th encoder layer. σ is a non-linear activation function, e.g., ReLU. Note that $\mathbf{Z}^{(0)}$ denotes the initially imputed attribute matrix $\widetilde{\mathbf{X}}$. As seen, the GCN-based encoder conducts neighborhood aggregation in each layer, and it can be viewed as a neighbor imputation operation. Therefore, the missing attributes are gradually imputed during the calculation of the network and become complete in the latent embedding matrix $\mathbf{Z}^{(2)}$, i.e., $\mathbf{Z} \in \mathbb{R}^{N\times d}$. Although the neighborhood aggregation could provide primary imputation, only preserving neighbors without information filtering in the latent space may cause: 1) the representations could be noisy since the initially imputed values with little discriminative capability in $\widetilde{\mathbf{X}}$ would diffuse through the network; 2) to some extent, the representation learning of attribute-missing part may suffer from semantic bias as we do not leverage any supervised information. To overcome these limitations, we try to further refine the learned graph embedding by performing accurate information enhancement to introduce reliable non-local semantic information with few layers. Specially, we introduce an extra operation after the encoder $E(\cdot)$ as follow:

$$\mathbf{Z}_a = \alpha\mathbf{S}^{\mathcal{N}}\mathbf{Z} + (1-\alpha)\mathbf{S}'^{\mathcal{N}}\mathbf{Z}, \tag{4}$$

where α is the learnable weighting coefficient and we set α = 0.5 for initialization. $\mathbf{S}^{\mathcal{N}}, \mathbf{S}'^{\mathcal{N}} \in \mathbb{R}^{N\times N}$ are refined global and multi-order neighbor similarity matrices (i.e., indicator matrices) that are constructed in two different manners. As shown in Fig. 2, to construct $\mathbf{S}^{\mathcal{N}}$ and $\mathbf{S}'^{\mathcal{N}}$, we first

generate the affinity matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$ according to the latent embedding matrix as follow:

$$\mathbf{S}_{ij} = \frac{\mathbf{z}_i \mathbf{z}_j^{\mathbf{T}}}{\|\mathbf{z}_i\|\|\mathbf{z}_j\|}, \quad \forall\, i,j \in [1,N]. \tag{5}$$

Here $\mathbf{z}_i$ indicates the latent embedding of the i-th sample.

To improve the reliability of $\mathbf{S}$, two mechanisms are introduced. On the one hand, we search nearest neighbors for each node v_i from the global scope. The underlying idea is to discover non-adjacent nodes that share similar semantic features with the central node. For instance, in a citation network where each node indicates a specific paper and each edge indicates the citation relationship between two papers. Even though the research content of the two papers belong to the same research field (i.e., two nodes with similar features probably have the same label), they may not cite each other's work or share any citation in the graph since their authors apply the proposed algorithms into different application scenarios. We argue that such semantically similar entities that do not share a connection can be exploited via K-nearest neighbors (KNN) search from the global scope. To this end, we adopt a KNN strategy to filter the less confident similarity estimation in $\mathbf{S}$. Specifically, only the top K largest similarity values of each sample are kept while other values are assigned as -1. We denote the resultant indicator matrix as $\mathbf{S}^{\mathcal{N}}$, where $\mathbf{s}_{ij}^{\mathcal{N}} = 1$ if the embeddings of node v_i and node v_j are semantically similar, otherwise $\mathbf{s}_{ij}^{\mathcal{N}} = 0$. On the other hand, we conduct structure-oriented K-nearest neighbors (SKNN) search to further boost the quality of the latent space from the multi-order neighbor scope. The intuition is that the multi-order neighbor nodes tend to share the same label as the central node. For example, in a citation network, even though two papers are not directly connected, they may be semantically similar and probably be the same category since they have some common citations. Hence, it is worth exploring the informative information and filtering useless one in the range of multi-order neighbor nodes. In this regard, to construct $\mathbf{S}^{'\mathcal{N}}$, we first preserve all the $1st$- to P-th-order neighbor similarity values of each sample in affinity matrix, then filter the non-neighbor elements in $\mathbf{S}$ by setting them to -1. Finally, we conduct a KNN strategy to get the final refined indicator matrix $\mathbf{S}^{'\mathcal{N}}$, similar to the constructing process of $\mathbf{S}^{\mathcal{N}}$.

With the refined similarity matrices, we update the latent embedding $\mathbf{Z}$ with Eq.(4) for reliable information aggregation, so as to improve the quality of missing attribute imputation and the discriminative capability of the latent features simultaneously.

3.3.2 Hidden Structure Refining

As known, graphs contain not only the attribute but also the structure information, which acts as a strong constraint to the node relationship in the graph [38], [39], [40]. Although experimental results (section 4.3.2 for details) have demonstrated that the unreliable similarity filtering mechanism in the DNA module could guarantee the quality of attribute restoration, the structure information of attribute-missing nodes has not been carefully considered. To fill this gap, we propose a hidden structure refining (HSR) module, consisting of two schemes, i.e., the multi-order neighbor attentive fusion and the hidden structure recovery. The intuition is to guide the network to automatically exploit complementary information and have the two kinds of information to verify each other. For example, in the citation network, raw features or detailed descriptions of one paper may be unavailable due to copyright protection. If the citation information (edges) between two papers (nodes) could be well preserved, that unobserved information may be easily inferred from the other accessible paper. We argue that the observed and missing parts should be semantically similar if they are connected via some reliable edges. Specifically, we introduce the hidden structure refining operation as an auxiliary task for more accurate latent space construction. Fig. 2(b) illustrates the design of the HSR module. Likewise, by transferring given inputs, i.e., $\widetilde{\mathbf{X}}$ and $\dot{\mathcal{A}} = \{\dot{\mathbf{A}}^{1st}, \dot{\mathbf{A}}^{2nd}, \ldots, \dot{\mathbf{A}}^{H\text{-}th}\}$, into a weight-sharing graph encoder $E(\cdot)$, we model the h-th path latent representations $\mathbf{Z}^{h\text{-}th(l)}$ as:

$$\mathbf{Z}^{h\text{-}th(l)} = \sigma(\dot{\mathbf{A}}^{h\text{-}th}\mathbf{Z}^{h\text{-}th(l-1)}\mathbf{W}^{(l)}), \tag{6}$$

where we consider $1st$- to $3rd$-order neighbors, i.e., H=3, $h \in [1,H]$. $\mathbf{Z}^{h\text{-}th(0)}$ and $\mathbf{Z}^{h\text{-}th(2)}$ are denoted as $\widetilde{\mathbf{X}}$ and the h-th path latent embedding matrix where each node aggregates h-th-order observed attributes, respectively. Then we transform these embedding matrices though a nonlinear transformation (e.g., one-layer MPL) to estimate the importance of each path. For node v_i, where its embedding in $\mathbf{Z}^{h\text{-}th}$ is $\mathbf{z}_i^{h\text{-}th} \in \mathbb{R}^{1\times d}$, a normalized attention weight $c_i^{h\text{-}th}$ using softmax function is formulated as follows:

$$c_i^{h\text{-}th} = \frac{e^{(\mathbf{W}^{h\text{-}th}(\mathbf{z}_i^{h\text{-}th})^{\mathbf{T}} + \mathbf{b}^{h\text{-}th})}}{\sum_{h=1}^{H} e^{(\mathbf{W}^{h\text{-}th}(\mathbf{z}_i^{h\text{-}th})^{\mathbf{T}} + \mathbf{b}^{h\text{-}th})}}, \tag{7}$$

where $\mathbf{W}^{h\text{-}th} \in \mathbb{R}^{d\times 1}$ denotes the learnable attention parameters and $\mathbf{b}^{h\text{-}th} \in \mathbb{R}^{d\times 1}$ denotes the bias vector of h-th path. Larger $c_i^{h\text{-}th}$ illustrates that the h-th-order observed neighbors could provide more informative information for

Algorithm 1 Training procedure of SAGA

Input: Initially imputed attribute matrix $\widetilde{\mathbf{X}}$; Normalized adjacency matrix $\widetilde{\mathbf{A}}$; A series of edge-masked adjacent matrices of different orders $\dot{\mathcal{A}}$; Iteration number I; Hyper-parameters P, K, γ, λ.
Output: Rebuilt attribute matrix $\widehat{\mathbf{X}}$.

```
Initialize the model parameters θ with an Xavier initialization;
for i = 1 to I do
    Utilize E to encode Z by Eq.(3);
    Construct S^N and S'^N using KNN by Eq.(5);
    Calculate Z_a by Eq.(4);
    Utilize E to encode {Z^1st, Z^2nd, ..., Z^H-th} by Eq.(6);
    Calculate Z_s by Eq.(7) and Eq.(8);
    Rebuilt Â based on Z_s by Eq.(9);
    Calculate L_s by Eq.(10), Eq.(11), and Eq.(12);
    Fuse Z_f using Z_a and Z_s by Eq.(13);
    Utilize D to decode Z_f and output X̂ by Eq.(14);
    Optimize the network with Adam by minimizing Eq.(16);
end for
return X̂
```

node v_i. Next, we combine these latent embedding matrices with attention weights:

$$\mathbf{Z}_s = \sum_{h=1}^{H} (\mathbf{C}^{h\text{-}th})^{\mathbf{T}} \odot \mathbf{Z}^{h\text{-}th}, \tag{8}$$

where $\odot$ means matrix product, $\mathbf{C}^{h\text{-}th} \in \mathbb{R}^{d \times N}$ is denoted as $[\mathbf{c}_1^{h\text{-}th}, \mathbf{c}_2^{h\text{-}th}, \ldots, \mathbf{c}_N^{h\text{-}th}]$ and $\mathbf{c}_n^{h\text{-}th} \in \mathbb{R}^{d \times 1}$ is an attention vector that repeats $c_n^{h\text{-}th}$ with d times. After that, $\mathbf{Z}_s$ is decoded into $\widehat{\mathbf{A}}$ through matmul product with an activation function:

$$\widehat{\mathbf{A}} = \mathit{Sigmoid}(\mathbf{Z}_s \mathbf{Z}_s^{\mathbf{T}}). \tag{9}$$

According to Eq.(9), we can model the linkage relation between node v_i and node v_j by minimizing:

$$l_{ij} = -[\widetilde{\mathbf{A}}_{ij} \ln \widehat{\mathbf{A}}_{ij} + (1 - \widetilde{\mathbf{A}}_{ij}) \ln(1 - \widehat{\mathbf{A}}_{ij})], \tag{10}$$

where there exists a linkage between node i and node j in original graph if $\widetilde{\mathbf{A}}_{ij} \neq 0$, otherwise 0. To enable the network to focus more on attribute-missing nodes to exploit their intrinsic structures, we introduce a pre-defined hyper-parameter γ to balance the hidden structure recovery processes of two types of linkage relations, i.e., the manually-masked part and the manually-preserved part, corresponding to the dotted and solid gray lines in Fig. 2(b). Eq.(10) can be reformulated as:

$$\mathcal{L}_{ij} = \begin{cases} \gamma l_{ij}, & v_i, v_j \in \mathcal{V}^m \\ l_{ij}, & \text{otherwise} \end{cases}. \tag{11}$$

This is very different from the structure preservation fashion of missing nodes as in SAT [8] that we manually mask the edge between attribute-missing nodes on multiple graphs and enforce the network to focus more on predicting these connections. Next, we summarize the merits of the fashion of our edge recovery: 1) more naturally evaluates the overall quality of restored attributes via hidden structure refinement; 2) in turn enforces the resulting network to be able to selectively exploit more high-order discriminative information for data completion. Finally, the average structure reconstruction loss of all node pairs can be written as:

$$\mathcal{L}_s = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathcal{L}_{ij}. \tag{12}$$

3.4 Information Aggregation and Decoding

After obtaining the attribute-enhanced latent embedding matrix $\mathbf{Z}_a$ and the structure-enhanced latent embedding matrix $\mathbf{Z}_s$ from the DNA module and HSR module, we combine both with a learnable weighting coefficient β, where β is initialized as 0.5. Then we directly feed the fused latent embedding matrix $\mathbf{Z}_f$ with $\widetilde{\mathbf{A}}$ into a graph decoder $D(\cdot)$. This process is formulated as:

$$\mathbf{Z}_f = \beta \mathbf{Z}_a + (1 - \beta) \mathbf{Z}_s, \tag{13}$$

$$\mathbf{Z}^{'(l)} = \sigma(\widetilde{\mathbf{A}} \mathbf{Z}^{'(l-1)} \mathbf{W}^{'(l)}), \tag{14}$$

where $\mathbf{W}^{'(l)}$ denotes the learnable parameters of the l-th decoder layer. $\mathbf{Z}^{'(0)}$ and $\mathbf{Z}^{'(2)}$ are denoted as the fused latent embedding matrix $\mathbf{Z}_f$ and the rebuilt attribute matrix $\widehat{\mathbf{X}}$, respectively.

TABLE 2
Summary of datasets.

Dataset	Nodes	Edges	Dimension	Classes
Cora	2708	5278	1433	7
Citeseer	3327	4228	3703	6
Amac	13752	245861	767	10
Amap	7650	119081	745	8
Pubmed	19717	44324	500	3
Cocs	18333	81894	6805	15

3.5 Joint Loss and Optimization

The overall learning objective consists of two parts, i.e., the attribute reconstruction loss of SAGA, and the structure reconstruction loss that is correlated with HSR module:

$$\mathcal{L}_a = \frac{1}{2N^o} \|\widetilde{\mathbf{X}}^o - \widehat{\mathbf{X}}^o\|_F^2. \tag{15}$$

$$\mathcal{L}_{total} = \lambda \mathcal{L}_a + \mathcal{L}_s. \tag{16}$$

In Eq.(16), $\mathcal{L}_a$ denotes the mean square error (MSE) between the observed parts of $\widetilde{\mathbf{X}}$ and $\widehat{\mathbf{X}}$. λ is a pre-defined hyper-parameter that balances the importance of both reconstruction processes. The detailed learning procedure of the proposed SAGA is shown in Algorithm 1. Compared to existing attribute-missing oriented graph representation learning methods, we design a totally different and effective framework to solve the newly proposed problem, i.e., unsupervised graph representation learning on attribute-missing graphs. Here we summarize the merits of our proposed framework with the following factors: our SAGA 1) imposes little distribution assumption on the latent space variables; 2) entangles the learning of attribute embedding and structure embedding to take full advantage of the two types of information; 3) exploits more abundant and robust information, which leads to more discriminative latent representations.

4 Experiments

We evaluate the benefits of SAGA against several state-of-the-art graph representation learning algorithms in the profiling and the node classification tasks. The experiments aim to answer the following research questions:

- **Q1**. How does SAGA perform compared to other baselines in the profiling and the node classification tasks? (see Section 4.2)
- **Q2**. How do the proposed components influence the performance of SAGA? (see Section 4.3)
- **Q3**. How do key hyper-parameters influence the performance of SAGA? (see Section 4.4)
- **Q4**. How about the algorithm convergence on different benchmark datasets? (see Section 4.5)

In the following, we begin with a brief introduction of the experimental setup and then provide detailed experiment results with corresponding analysis.

4.1 Experimental Setup

4.1.1 Benchmark Datasets

We evaluate the proposed SAGA on six benchmark datasets, including four datasets with categorial attributes, i.e., Cora [41], Citeseer [42], Amazon-Computer and Amazon-Photo [43], and two datasets with real-valued attributes, i.e., Pubmed [42] and Coauthor-CS [44]. Table 2 summarizes the brief information of these datasets.

- **Cora**. Cora is a citation network (graph) that consists of 2,708 scientific papers (nodes) classified into one of seven classes. The citation network consists of 10,556 citation links (edges). Each paper in the Cora is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary, where the dictionary contains 1,433 unique words (dimension).
- **Citeseer**. Citeseer is a citation network (graph) that consists of 3,327 scientific papers (nodes) classified into one of six classes. The citation network consists of 9,228 citation links (edges). Each paper in the Citeseer is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary, where the dictionary contains 3,703 unique words (dimension).
- **Amazon Computer (Amac) and Amazon Photo (Amap)**. Amazon Computer and Amazon Photo are segments of the Amazon co-purchase graph, where nodes represent goods, edges indicate that two goods are frequently bought together, node features are bag-of-words encoded product reviews. Amazon Computers consists of 13,752 nodes with dimensions 767 and 245,861 edges. Amazon photo consists of 7650 nodes with dimensions 745 and 119,081 edges. Amazon Computers and Amazon Photo are classified into ten classes and eight classes, respectively.
- **Pubmed**. Pubmed is also a citation network (graph) which consists of 19,717 scientific papers (nodes) classified into one of three classes and 88,651 citation links (edges). Each publication in the dataset is described by a TF/IDF weighted word vector from a dictionary which consists of 500 unique words (dimension).
- **Coauthor-CS (Cocs)**. Coauthor-CS is a co-authorship graph based on the Microsoft Academic Graph from the KDD Cup 2016 challenge. This graph contains 18,333 authors (nodes) classified into fifteen classes and 81,894 connections (edges) that connect two authors if they co-author a paper. Each node consists of 6,805 features (dimensions) representing paper keywords for each author's papers.

4.1.2 Training Procedure

Our experiments are implemented with PyTorch 1.6 platform and run with four NVIDIA Tesla V100S GPU cards. The adjacency matrix is normalized in pre-processing step using NetworkX 2.5.1 and Numpy 1.16.1 packages. The training of the proposed SAGA includes two steps in total. Firstly, we perform the profiling task using Recall@K and NDCG@K as metrics to evaluate the quality of restored attributes. Specifically, we train our SAGA in an unsupervised manner by minimizing the reconstruction loss function $\mathcal{L}_{total}$ for at least 500 iterations until convergence. During the training, we apply the weighted Binary Cross-Entropy loss (BCE) or the Mean Square Error (MSE) loss as our training objective for categorical or real-valued graph data, respectively. To avoid the over-fitting issue, we adopt an early stop strategy when the loss value comes to a plateau. Secondly, we utilize a GCN-based classifier to perform the node classification task over the attribute-restored nodes, where we learn the node embeddings supervised by a cross-entropy loss function with five-fold validation in 10 times, and report the averages evaluated by accuracy (ACC) metric. Note that we strictly follow the classifier settings of SAT [8] in the node classification task.

4.1.3 Implementation Details

For all compared algorithms except for GINN [13] and GCNMF [7], the experimental results are acquired according to the paper of SAT [8]. For GINN and GCNMF algorithms, we first adopt the data and code of SAT to process data before training, then we set the hyper-parameters of both methods by following their original papers. After that, we run their publicly available Pytorch code and report the corresponding performance. For our proposed SAGA, we strictly follow the same data splits as in SAT [8] for performance comparison on all benchmark datasets, including the split of attribute-observed/-missing nodes and the spilt of train/test sets. Specifically, 1) in the profiling task, we randomly sample 40% nodes with attributes as the training set, and manually-mask all attributes of the rest 10% and 50% nodes (i.e., attribute-missing nodes) as the validation set and the test set, respectively. We adopt 4-layer GCNs as our backbone and train it with Adam optimizer, where the learning rate is set to 1e-3. During the training phase, we transfer all samples into our proposed auto-encoder to restore missing attributes by merely reconstructing the attribute-observed nodes (i.e., training set). After training, we directly generate the rebuilt attribute matrix over the well-trained model via a forwarding propagation algorithm. According to the results of parameter sensitivity testing, we set the hyper-parameters P, K as 5 and fix the balanced coefficients γ and λ to 5 and 10, respectively; 2) in the node classification task, the attribute-restored nodes are randomly split into 80% train data and 20% test data with five-fold validation for 1000 iterations in 10 times. Moreover, the learning rate, the latent dimension, the dropout rate, and the weight decay are set to 1e-3, 64, 0.5, and 5e-4, respectively. We adopt early stopping to avoid the over-fitting phenomenon. In the GCN-based classifier, all the compared algorithms consistently utilize the normalized adjacency matrix $\widetilde{\mathbf{A}}^m$ for message passing, where $\widetilde{\mathbf{A}}^m \in \mathbb{R}^{N^m \times N^m}$ describes the linkage relations among attribute-missing nodes.

4.2 Baselines and Comparison Results (Q1)

4.2.1 Baseline Algorithms

In the following, we compare our SAGA with 11 related methods to illustrate its effectiveness. Among these baselines, **NeighAggre** [45] is the representative one of classical profiling algorithms. **VAE** [46] is a well-known auto-encoder-based generative method. **GCN** [15], **GraphSage**

TABLE 3
Profiling performance comparison between the proposed SAGA and nine state-of-the-art algorithms. In this table, two metrics (Recall and NDCG using top 10, 20, 50) of different algorithms on four datasets are reported. ↑ denotes the performance improvement over the SAT algorithm. The boldface and underline values indicate the best and the runner-up results, respectively.

Dataset	Method	Recall@10	Recall@20	Recall@50	NDCG@10	NDCG@20	NDCG@50
Cora	NeighAggre [45]	9.06	14.13	19.61	12.17	15.48	18.50
	VAE [46]	8.87	12.28	21.16	12.24	14.52	19.24
	GCN [15]	12.71	17.72	29.62	17.36	20.76	27.02
	GraphSage [47]	12.84	17.84	29.72	17.68	21.02	27.28
	GAT [48]	13.50	18.12	29.72	17.91	20.99	27.11
	Hers [49]	12.26	17.23	27.99	16.94	20.31	25.96
	GraphRNA [50]	13.95	20.43	31.42	19.34	23.62	29.38
	ARWMF [51]	12.91	18.13	29.60	18.24	21.82	27.76
	SAT [8]	<u>15.08</u>	<u>21.82</u>	<u>34.29</u>	<u>21.12</u>	<u>25.46</u>	<u>32.12</u>
	Ours	**16.86 (1.78↑)**	**23.85 (2.03↑)**	**35.97 (1.68↑)**	**23.49 (2.37↑)**	**28.16 (2.70↑)**	**34.59 (2.47↑)**
Citeseer	NeighAggre [45]	5.11	9.08	15.01	8.23	11.55	15.60
	VAE [46]	3.82	6.68	12.96	6.01	8.39	12.51
	GCN [15]	6.20	10.97	20.52	10.26	14.23	20.49
	GraphSage [47]	6.12	10.97	20.58	10.03	13.93	20.34
	GAT [48]	5.61	10.12	19.57	8.78	12.53	18.72
	Hers [49]	5.76	10.25	19.73	9.04	12.79	19.00
	GraphRNA [50]	<u>7.77</u>	12.72	22.71	12.91	17.03	23.58
	ARWMF [51]	5.52	10.15	19.52	8.59	12.45	18.58
	SAT [8]	7.64	<u>12.80</u>	<u>23.77</u>	<u>12.98</u>	<u>17.29</u>	<u>24.47</u>
	Ours	**9.45 (1.81↑)**	**15.35 (2.55↑)**	**26.74 (2.97↑)**	**16.15 (3.17↑)**	**21.07 (3.81↑)**	**28.58 (4.11↑)**
Amac	NeighAggre [45]	3.21	5.93	13.06	7.88	11.56	19.23
	VAE [46]	2.55	5.02	11.96	6.32	9.70	17.21
	GCN [15]	2.73	5.33	12.75	6.71	10.27	18.24
	GraphSage [47]	2.69	5.28	12.78	6.64	10.20	18.22
	GAT [48]	2.71	5.30	12.78	6.73	10.28	18.30
	Hers [49]	2.73	5.25	12.73	6.76	10.25	18.25
	GraphRNA [50]	3.86	6.90	14.65	9.31	13.33	21.55
	ARWMF [51]	2.80	5.44	12.89	6.94	10.53	18.51
	SAT [8]	<u>3.91</u>	<u>7.03</u>	<u>15.14</u>	<u>9.63</u>	<u>13.79</u>	<u>22.43</u>
	Ours	**4.45 (0.54↑)**	**7.87 (0.84↑)**	**16.51 (1.37↑)**	**10.88 (1.25↑)**	**15.44 (1.65↑)**	**24.63 (2.20↑)**
Amap	NeighAggre [45]	3.29	6.16	13.61	8.13	11.96	19.98
	VAE [46]	2.76	5.38	12.79	6.75	10.31	18.30
	GCN [15]	2.94	5.73	13.24	7.05	10.82	18.93
	GraphSage [47]	2.95	5.62	13.22	7.12	10.79	18.96
	GAT [48]	2.94	5.73	13.24	7.05	10.83	18.92
	Hers [49]	2.92	5.74	13.28	7.14	10.94	19.06
	GraphRNA [50]	3.90	7.03	15.08	9.59	13.77	22.32
	ARWMF [51]	2.94	5.68	13.27	7.27	10.98	19.15
	SAT [8]	<u>4.10</u>	<u>7.43</u>	<u>15.97</u>	<u>10.06</u>	<u>14.50</u>	<u>23.95</u>
	Ours	**4.55 (0.45↑)**	**7.92 (0.49↑)**	**16.33 (0.36↑)**	**11.08 (1.02↑)**	**15.57 (1.07↑)**	**24.51 (0.56↑)**

[47], and **GAT** [48] are typical graph convolutional network (GCN)-based methods, where the node representations are embedded with structure information by GCN. **GraphRNA** [50] and **ARWMF** [51] are representatives of attributed random walk-based methods, which apply random walks on the node-attribute bipartite graphs and can potentially tackle the attribute-missing issue. Since the attribute-missing graph learning is similar to cold-start recommendation task, thus one representative cold-start recommendation method called **Hers** [49] is involved as a baseline. Further, we report the performance of two attribute-incomplete GRL methods, i.e, **GINN** [13], **GCNMF** [7] and a state-of-the-art attribute-missing GRL method **SAT** [8]. Table 3 and Table 4 summarize the performance comparison on the profiling and the node classification tasks.

4.2.2 Profiling Task

In the profiling task, four observations can be obtained from Table 3: 1) SAGA shows the best performance in terms of six metrics against all compared baselines on four datasets. For instance, SAT has been considered as the strongest attribute-missing GRL framework, and our method exceeds it by 1.68%/2.47%, 2.97%/4.11%, 1.37%/2.20%, and 0.36%/0.56% in terms of Recall@50 and NDCG@50 Cora, Citeseer, Amac, and Amap, which verifies the effectiveness of structure-attribute mutual enhanced learning strategy in handling attribute-missing graphs; 2) it can be seen that SAGA consistently outperforms the attributed rand walk-based methods. This is because the operation of random walks may introduce noise to the learning process, which affects the quality of restored attributes; 3) our SAGA also achieves better performance than GCN, GraphSage, and GAT, all of which have been demonstrated the strong representation learning capability on complete graphs, while these methods are not suitable to effectively handle the attribute-missing graphs; 4) since NeighAggre and VAE isolate the processes of data completion and network learning. Thus both algorithms are

TABLE 4
Node classification performance comparison between the proposed SAGA and twelve state-of-the-art algorithms. In this table, the accuracy of different algorithms on six datasets is reported. ↑ denotes the performance improvement over the SAT algorithm. The boldface and underline values indicate the best and the runner-up results, respectively.

Method	Cora	Citeseer	Amac	Amap	Pubmed	Cocs
NeighAggre [45]	64.94	54.13	87.15	90.10	65.64	80.31
VAE [46]	30.11	26.63	40.23	37.81	40.07	23.35
GCN [15]	43.87	40.79	39.74	36.56	42.03	21.80
GraphSage [47]	57.79	42.78	40.19	37.84	42.00	23.35
GAT [48]	45.25	26.88	40.34	37.89	41.96	23.34
Hers [49]	34.05	32.29	40.25	37.94	42.05	23.34
GraphRNA [50]	81.98	63.94	<u>86.50</u>	<u>92.07</u>	**81.72**	<u>88.51</u>
ARWMF [51]	80.25	27.64	74.00	61.46	<u>80.89</u>	83.47
GINN [13]	67.58	55.32	81.72	87.77	54.28	79.74
GCNMF [7]	70.30	63.40	76.43	87.79	62.00	87.53
SAT [8]	<u>83.27</u>	<u>65.99</u>	85.19	91.63	75.37	85.76
Ours	**85.13 (1.86↑)**	**69.25 (3.26↑)**	**88.65 (3.46↑)**	**92.36 (0.73↑)**	80.55 (5.18↑)	**88.90 (3.14↑)**

TABLE 5
Ablation study on initial imputation. Node classification performance of the three designed algorithms on six datasets are reported. GVF-based SAGA, NVF-based SAGA, and ZVF-based SAGA are the algorithms where the missing attributes in $\mathbf{X}$ are filled with random values from a standard Gaussian, observed neighbor values, and zero values. The boldface value indicates the best result.

Method	Cora	Citeseer	Amac	Amap	Pubmed	Cocs
GVF-based SAGA	85.03	67.10	87.82	91.81	70.94	84.63
NVF-based SAGA	**85.31**	69.11	88.21	91.99	**81.36**	88.79
ZVF-based SAGA	85.13	**69.25**	**88.65**	**92.36**	80.55	**88.90**

not comparable to ours.

4.2.3 Node Classification Task

In the node classification task, as reported in Table 4, we can find that 1) our method achieves the best average performance in terms of accuracy on five of six datasets. For instance, SAGA exceeds SAT by 1.86%, 3.26%, 3.46%, 0.73%, and 5.18%, 3.14% accuracy increment. These results well demonstrate that our SAGA could learn a more discriminative latent embedding for data completion by taking full advantage of the attribute and structure information, thus boosting the node classification performance; 2) compared with GINN and GCNMF, our method gains 14.83%, 5.85%, 6.93%, 4.57%, 18.55%, and 1.37% accuracy increment, which indicates that our SAGA is competent to handle attribute-missing graphs, while incomplete GRL methods can not provide effective solutions; 3) it is worth noting that GraphRNA and ARWME achieve slightly better results than that of ours on Pubmed dataset, this is because GraphRNA and ARWME could naturally model the correlation between attribute dimension, especially for handling real-valued graph data.

Overall, the results of both the profiling and the node classification tasks have solidly demonstrated the superiority and effectiveness of SAGA in solving attribute-missing graph representation learning.

4.3 Ablation Comparison (Q2)

4.3.1 Influence of Initial Imputation

Since only partial nodes with observed features exist, it should be considered to assign initial features to the attribute-missing nodes before network training. In Table 5, to illustrate the influence of different fashions of initial imputation, we design three algorithms and compare their performance. GVF-based SAGA, NVF-based SAGA, and ZVF-based SAGA are algorithms where we initialize the missing attributes with random values from a standard Gaussian, observed neighbor values, and zero values. The node classification accuracy performance of three algorithms has been reported in Table 5. From this table, we observe that 1) GVF-based SAGA are not comparable to the other two counterparts on Citesser, Amac, Pubmed, and Cocs. This is because random features contain much semantically irrelevant information that will diffuse through the network, which affects the discriminative capacity of the imputed representations and even causes a distortion of the graph; 2) NVF-based SAGA and ZVF-based SAGA consistently achieve similar performance on six datasets, which indicates that SAGA is insensitive to the initial imputation when setting the missing attributes to zero values or observed neighbor values.

4.3.2 Effectiveness of Each Component

In this section, we design an ablation study to demonstrate the effectiveness of the proposed components. Here we denote a naive graph auto-encoder as the baseline method. +HSR and +DNA refer to the baseline methods with the HSR module and the DNA module, respectively. +PS denotes a pseudo-Siamese counterpart of SAGA. In Fig. 3, we experimentally compare all methods and report their performance on four datasets. We can see that 1) the +HSR method and the +DNA method consistently improve the baseline in terms of six metrics over all datasets. Taking the results on Amazon-C for example, the +HSR method and the +DNA method gain 2.12%/2.21% and 3.43%/3.49% accuracy increment in terms of Recall@50/NDCG@50. These results verify the effectiveness of sufficient interaction between attribute and structure information (i.e., unreliable similarity filtering

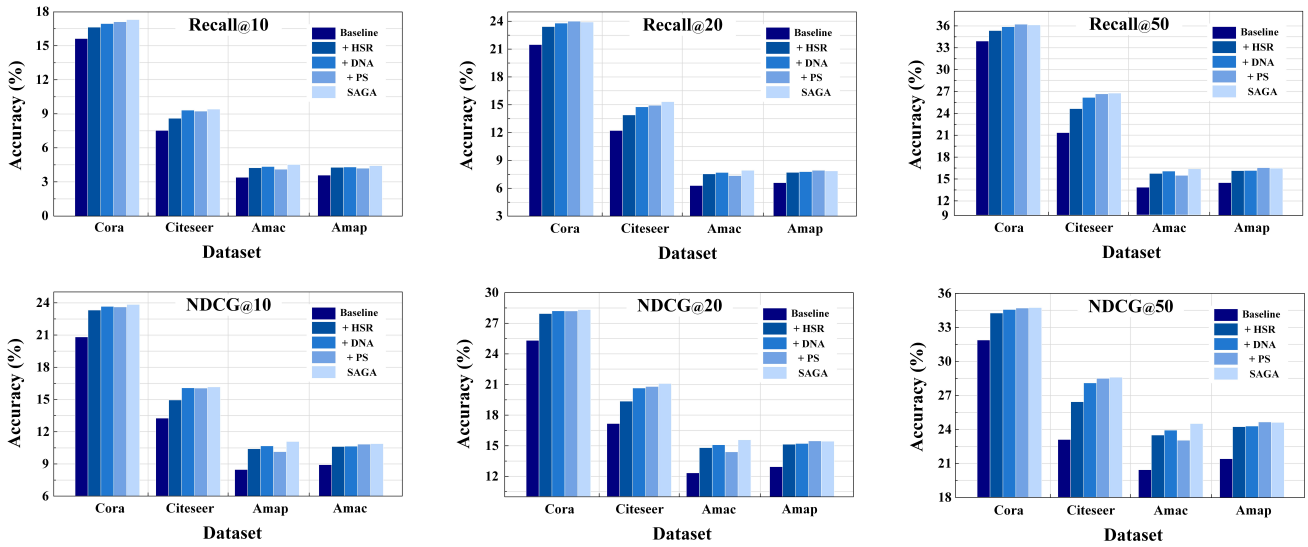

Fig. 3. Ablation study on the effectiveness of each component. In this figure, two metrics (Recall and NDCG using top 10, 20, 50) of the five designed algorithms on four datasets are reported. The baseline is the algorithm that adopts a naive graph auto-encoder without any proposed components. +HSR is the algorithm with the proposed hidden structure refining module. +DNA is the algorithm with the proposed dual non-local aggregating module. +PS is a pseudo-Siamese counterpart of the proposed SAGA. The performance of our SAGA is listed in the last bar.

TABLE 6
Ablation study on the dual non-local aggregating mechanism of the DNA module. In this table, two metrics (Recall and NDCG using top 10, 20, 50) of the three designed algorithms on four datasets are reported. DNA-KNN is the algorithm where we merely utilize K-nearest neighbor (KNN) to filter unreliable similarities upon the affinity matrix. DNA-KNN is the algorithm where we merely utilize structure-oriented K-nearest neighbor (SKNN) to filter unreliable similarities upon the affinity matrix. DNA indicates our proposed SAGA using a dual non-local aggregating mechanism. ↑ denotes the performance improvement. The boldface value indicates the best result.

Dataset	Method	Recall@10	Recall@20	Recall@50	NDCG@10	NDCG@20	NDCG@50
Cora	DNA-KNN	16.72	23.66	35.82	23.26	28.01	34.43
	DNA-SKNN	16.55	23.38	35.73	23.02	27.87	34.14
	DNA	**16.86 (0.31↑)**	**23.85 (0.53↑)**	**35.97 (0.24↑)**	**23.49 (0.47↑)**	**28.16 (0.29↑)**	**34.59 (0.45↑)**
Citeseer	DNA-KNN	9.30	15.04	26.32	16.12	20.91	28.28
	DNA-SKNN	9.22	14.94	26.48	15.71	20.49	28.03
	DNA	**9.45 (0.23↑)**	**15.35 (0.41↑)**	**26.74 (0.42↑)**	**16.15 (0.44↑)**	**21.07 (0.58↑)**	**28.58 (0.55↑)**
Amac	DNA-KNN	4.29	7.61	16.23	10.46	15.21	24.40
	DNA-SKNN	4.22	7.50	16.11	10.31	14.92	24.24
	DNA	**4.45 (0.23↑)**	**7.87 (0.37↑)**	**16.51 (0.40↑)**	**10.88 (0.57↑)**	**15.44 (0.52↑)**	**24.63 (0.39↑)**
Amap	DNA-KNN	4.38	7.75	16.03	10.75	15.26	24.26
	DNA-SKNN	4.30	7.59	16.27	10.70	15.27	24.38
	DNA	**4.55 (0.25↑)**	**7.92 (0.33↑)**	**16.33 (0.30↑)**	**11.08 (0.38↑)**	**15.57 (0.31↑)**	**24.51 (0.25↑)**

and graph structure refinement) for data completion. We can obtain similar observations from the results on other metrics and datasets; 2) this ablation study also reveals the advantage of our Siamese architecture, which can help to better exploit the two-source information for data imputation. As seen, the proposed SAGA demonstrates slightly better performance than the pseudo-Siamese counterpart. According to these observations, the effectiveness of the proposed components in SAGA has been clearly verified.

4.3.3 Analysis of the DNA Mechanism

To further reveal the effect of the dual non-local aggregating mechanism, we present the profiling performance of three SAGA variants, i.e., DNA-KNN, DNA-SKNN, and DNA (our proposed method) in Table 6. DNA-KNN and DNA-SKNN are the algorithms where we utilize K-nearest neighbor (KNN) and structure-oriented K-nearest neighbor (SKNN) to filter unreliable similarities in the latent space, respectively. From this table, we can find that 1) taking the results on Cora for instance, two proposed correlation aggregating strategies can achieve promising performance in terms of six metrics, it is obvious that filtering unreliable similarities and preserving more informative information could boost the discriminative capacity of the learned latent embedding. We can obtain similar observations from the results on other datasets; 2) our dual non-local aggregating mechanism consistently achieves better performance than other counterparts. These results illustrate that although there exists some overlapping information between the learning processes of DNA-KNN and DNA-SKNN, combining both together does help each other to exploit more informative information to some extent.

4.3.4 Analysis of the DNA and HSR Structure Designs

In this subsection, we conduct additional experiments to show the effect of different DNA and HSR structure designs.

TABLE 7
Ablation study on different structures of the DNA module and the HSR module. In this table, two metrics (Recall and NDCG using top 10, 20, 50) of the two designed algorithms on four datasets are reported. DNA-HSR-H is the algorithm where the DNA module and the HSR module are structured in a hierarchical design (DNA after HSR). DNA-HSR-P is the algorithm where the DNA module and the HSR module are structured in a parallel design. ↑ denotes the performance improvement. The boldface value indicates the best result.

Dataset	Method	Recall@10	Recall@20	Recall@50	NDCG@10	NDCG@20	NDCG@50
Cora	DNA-HSR-H	16.65	23.55	35.75	23.39	28.03	34.47
	DNA-HSR-P	**16.86 (0.21↑)**	**23.85 (0.30↑)**	**35.97 (0.22↑)**	**23.49 (0.10↑)**	**28.16 (0.13↑)**	**34.59 (0.12↑)**
Citeseer	DNA-HSR-H	8.44	13.59	24.21	14.61	18.93	25.89
	DNA-HSR-P	**9.45 (1.01↑)**	**15.35 (1.76↑)**	**26.74 (2.53↑)**	**16.15 (1.54↑)**	**21.07 (2.14↑)**	**28.58 (2.69↑)**
Amac	DNA-HSR-H	4.21	7.42	16.04	10.33	14.98	24.11
	DNA-HSR-P	**4.45 (0.24↑)**	**7.87 (0.45↑)**	**16.51 (0.47↑)**	**10.88 (0.55↑)**	**15.44 (0.45↑)**	**24.63 (0.52↑)**
Amap	DNA-HSR-H	4.27	7.71	16.31	10.53	15.10	24.26
	DNA-HSR-P	**4.55 (0.28↑)**	**7.92 (0.21↑)**	**16.33 (0.02↑)**	**11.08 (0.55↑)**	**15.57 (0.47↑)**	**24.51 (0.25↑)**

TABLE 8
Profiling performance comparison between SAT and our proposed SAGA with different observed ratio settings. In this table, two metrics (Recall and NDCG using top 10, 20, 50) of the two algorithms on two datasets are reported. ↑ denotes the performance improvement. The boldface value indicates the best result.

Dataset	Ratio	Method	Recall@10	Recall@20	Recall@50	NDCG@10	NDCG@20	NDCG@50
Cora	20%	SAT	13.49	19.75	31.02	19.33	23.51	29.61
		Ours	**14.60** (1.11↑)	**21.10 (1.35↑)**	**32.57 (1.55↑)**	**20.95 (1.62↑)**	**25.27 (1.76↑)**	**31.36 (1.75↑)**
	30%	SAT	14.89	21.30	33.46	20.87	25.15	31.59
		Ours	**15.93 (1.04↑)**	**22.52 (1.22↑)**	**34.83 (1.37↑)**	**22.40 (1.53↑)**	**26.82 (1.67↑)**	**33.32 (1.73↑)**
	40%	SAT	14.71	21.76	34.33	20.60	25.18	31.87
		Ours	**16.86 (2.15↑)**	**23.85 (2.09↑)**	**35.97 (1.64↑)**	**23.49 (2.89↑)**	**28.16 (2.98↑)**	**34.59 (2.72↑)**
	50%	SAT	15.72	22.46	34.82	21.69	26.14	32.73
		Ours	**17.22 (1.50↑)**	**24.59 (2.13↑)**	**36.94 (2.12↑)**	**23.96 (2.27↑)**	**28.81 (2.67↑)**	**35.39 (2.66↑)**
	60%	SAT	15.59	22.64	35.74	21.79	26.45	33.44
		Ours	**17.90 (2.31↑)**	25.21 **(2.57↑)**	**38.26 (2.52↑)**	**24.65 (2.86↑)**	**29.53 (3.08↑)**	**36.40 (2.96↑)**
	70%	SAT	15.68	22.75	36.63	21.77	26.47	33.78
		Ours	**18.13 (2.45↑)**	**26.10 (3.35↑)**	**39.53 (2.90↑)**	**25.33 (3.56↑)**	**30.61 (4.14↑)**	**37.78 (4.00↑)**
	80%	SAT	17.43	25.00	37.95	24.01	29.11	36.05
		Ours	**18.46 (1.03↑)**	**26.72 (1.72↑)**	**41.20 (3.25↑)**	**25.87 (1.86↑)**	**31.45 (2.34↑)**	**39.10 (3.05↑)**
Citeseer	20%	SAT	6.30	10.74	20.82	10.63	14.33	20.92
		Ours	**8.07 (1.77↑)**	**13.29 (2.55↑)**	**23.73 (2.91↑)**	**13.67 (3.04↑)**	**18.03 (3.70↑)**	**24.90 (3.98↑)**
	30%	SAT	7.08	11.87	22.46	12.18	16.20	23.13
		Ours	**8.80 (1.72↑)**	**14.27 (2.40↑)**	**25.30 (2.84↑)**	**15.05 (2.87↑)**	**19.64 (3.44↑)**	**26.91 (3.78↑)**
	40%	SAT	7.78	12.77	23.43	13.41	17.58	24.58
		DNA	**9.45 (1.67↑)**	**15.35 (2.58↑)**	**26.74 (3.31↑)**	**16.15 (2.74↑)**	**21.07 (3.49↑)**	**28.58 (4.00↑)**
	50%	SAT	8.16	13.37	24.36	13.93	18.28	25.48
		Ours	**10.04 (1.88↑)**	**15.92 (2.55↑)**	**27.46 (3.10↑)**	**17.07 (3.14↑)**	**22.00 (3.72↑)**	**29.57 (4.09↑)**
	60%	SAT	8.32	13.60	24.68	14.35	18.75	26.01
		Ours	**10.33 (2.01↑)**	**16.55 (2.95↑)**	**28.66 (3.98↑)**	**17.51 (3.16↑)**	**22.74 (3.99↑)**	**30.70 (4.69↑)**
	70%	SAT	8.54	14.02	25.38	14.72	19.30	26.76
		Ours	**11.01 (2.47↑)**	**17.40 (3.38↑)**	**29.56 (4.18↑)**	**18.56 (3.84↑)**	**23.96 (4.66↑)**	**31.94 (5.18↑)**
	80%	SAT	8.37	13.52	25.58	14.41	18.70	26.58
		Ours	**10.71 (2.34↑)**	**17.10 (3.58↑)**	**29.39 (3.81↑)**	**18.13 (3.72↑)**	**23.45 (4.75↑)**	**31.54 (4.96↑)**

Specifically, we develop two algorithms (i.e., DNA-HSR-H and DNA-HSR-P) and compare their performance. DNA-HSR-H and DNA-HSR-P mean that the DNA module and the HSR module are structured in a hierarchical and a parallel design, respectively. From the results in Table 7, DNA-HSR-P (our proposed SAGA) consistently achieves better performance than that of DNA-HSR-H. Taking the results on Citeseer for example, DNA-HSR-P exceeds DNA-HSR-H by 1.01%, 1.76%, 2.53%, 1.54%, 2.14%, and 2.69% in terms of six metrics. In addition, the observations on other datasets are similar. We attribute the superiority of DNA-HSR-P as the following aspect, in our Siamese architecture (the parallel design), the HSR module could be regarded as an auxiliary task to guide the network to automatically exploit intrinsic data structures for more accurate latent space construction. By combining the preserved latent variables of two proposed components, two-source information could negotiate with each other in a complementary manner, leading to higher-quality attribute restoration.

4.3.5 Analysis of the Observed Ratio

In this part, to further verify the effectiveness of the proposed SAGA, we compare it with the strongest baseline SAT on Cora and Citeseer by varying the observed ratio from 20% to 80%. Table 8 presents the Recall and NDCG using the top 10, 20, 50 comparisons of both algorithms. From this table, we have the following observations: 1) taking the results of NDCG@10 on both datasets for example, our proposed SAGA learned with only 20% observed attributes still can achieve comparable or slightly better performance

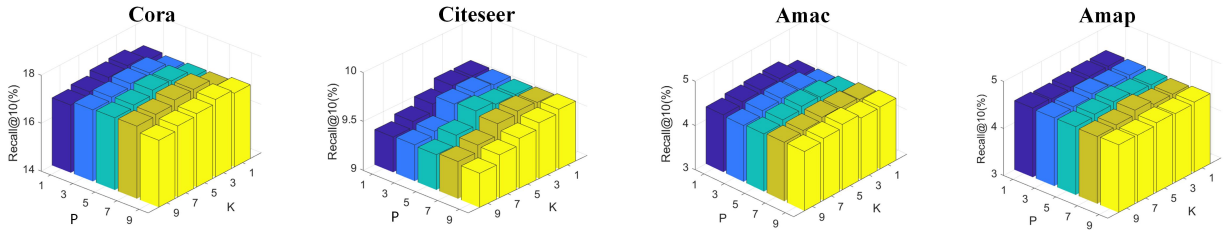

Fig. 4. Parameter sensitivity with P and K. In this figure, one metric (Recall using top 10) of the proposed SAGA on four datasets is reported. We explore the performance variation of our algorithm by varying hyper-parameters P and K from one to nine.

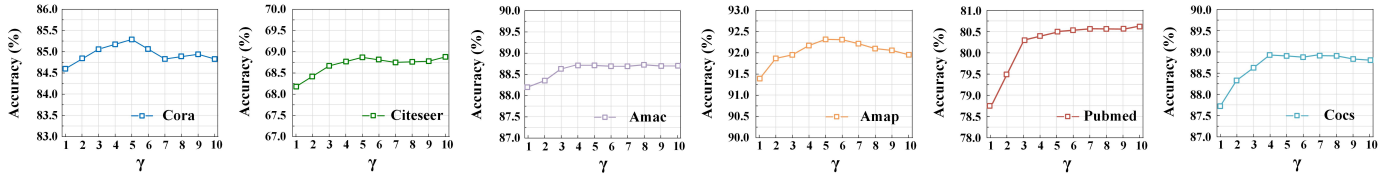

Fig. 5. Parameter sensitivity with γ. In this figure, the accuracy performance of the proposed SAGA on six datasets is reported. We explore the performance variation of our algorithm by varying the balanced coefficient γ from one to ten.

than SAT learned with 40% observed attributes. The results in terms of the other five metrics are similar; 2) the proposed SAGA outperforms SAT by 1.55%, 1.37%, 1.64%, 2.12%, 2.52%, 2.90%, 3.25% in terms of Recall@50 with the variation of observed ratio from 20% to 80% on Cora. The improvement is more significant with the increase of the observed ratio. The above observations have solidly demonstrated the effectiveness and robustness of our proposed SAGA.

4.4 Analysis of Hyper-parameters (Q3)

In this section, we study the algorithm sensitivity by setting different hyper-parameters (i.e., P, K, γ, and λ) and observing the changes in algorithm performance.

4.4.1 Parameter Analysis of P and K

Here we present the analysis of two hyper-parameters P and K on four datasets. As seen, Fig. 4 presents the performance variation of SAGA when P and K vary from 1 to 9, where we can see that 1) for a certain P, K value between 3 and 5 achieves better performance; 2) for a certain K, SAGA obtains stable results when P varies from 3 to 7. These indicate that our SAGA needs proper hyper-parameters to establish a structure-attribute mutual enhanced learning. Overall, SAGA performs well by setting P and K to 5 across both datasets.

4.4.2 Parameter Analysis of γ

We conduct experiments on six datasets to investigate the effect of hyper-parameter γ in Eq.(11). Fig. 5 presents the node classification performance with different γ, i.e., γ varies from 1 to 10. We can observe that 1) γ is effective in improving the performance; 2) the ACC metric first increases to a higher value and then keeps stable on Citeseer, Amac, Pubmed, and Cocs when γ varies from 0 to 10; 3) with the increasing value of γ, the performance on Cora and Amap tends to drop but still keeps better than that of the baseline (i.e., γ is set to 1); 4) SAGA performs well by setting γ to 5 across all datasets.

4.4.3 Parameter Analysis of λ

We also investigate the effect of hyper-parameter λ, which balances the importance of two reconstruction losses of SAGA. We vary λ from 2 to 20 and report the corresponding results. As shown in Fig. 6, we can find that 1) the performance can be improved with tuning the hyper-parameter λ; 2) increasing the value of λ from 2 to 8 slightly increases the performance, and continually increasing λ to a higher value obtains relatively stable performance; 3) the proposed SAGA tends to perform well across four datasets when setting λ to 10.

4.5 Algorithm Convergence (Q4)

To illustrate the convergence of the proposed SAGA, we record the metric reflected by Recall@10 on four datasets and show the performance variation as the training iteration goes. The results are illustrated in Fig. 7, we can see that 1) the performance of our method gradually increases to a plateau with an obvious tendency; 2) SAGA could converge within 1000 iterations. These results clearly verify the suitable convergence property of our proposed SAGA.

5 Conclusion

In this paper, we propose the SAGA method to handle attribute-missing graphs. In our network, two core components, i.e., DNA and HSR modules, take full advantage of structures and attributes, and allow both types of information to sufficiently interact with each other in a Siamese framework. In this way, unreliable similar information is filtered while more informative information can be well collected and preserved, which effectively enhances the discriminative capacity of the learned latent embedding for data completion. Moreover, our Siamese design can assist in boosting the learning capability of SAGA. Extensive experiments on six benchmark datasets have demonstrated the effectiveness and superiority of the proposed method. Future work may aim to extend SAGA to handle multi-view

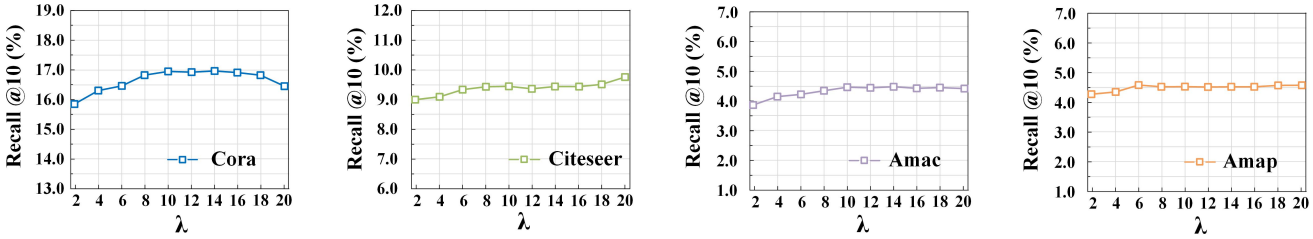

Fig. 6. Parameter sensitivity with λ. In this figure, one metric (Recall using top 10) of the proposed SAGA on four datasets is reported. We explore the performance variation of our algorithm by varying the balanced coefficient λ from two to twenty.

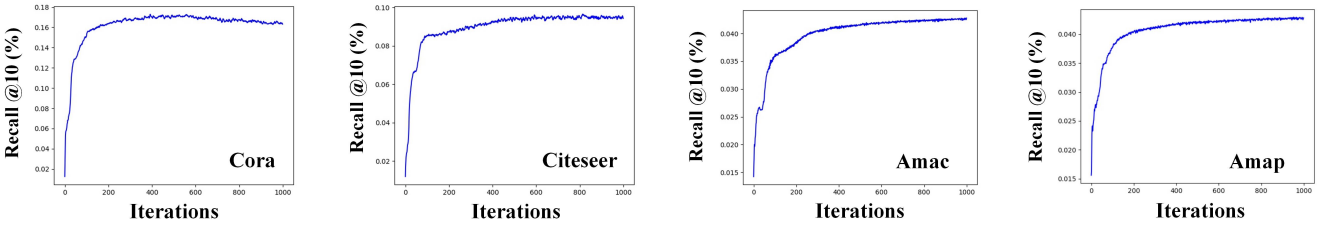

Fig. 7. Algorithm convergence of the proposed SAGA. X-axis and Y-axis refer to the number of iterations and the corresponding value of Recall@10 metric, respectively.

attribute-missing GRL, where various observed information from different views can be exploited to further improve the performance.

Acknowledgment

This work was supported by the National Key Research and Development Program of China (project no. 2020AAA0107100) and the National Natural Science Foundation of China (project no. 62006237).

References

[1] P. Hu, K. C. C. Chan, and T. He, "Deep graph clustering in social network," in *WWW*, 2017, pp. 1425–1426.
[2] Y. Shen, N. Ding, H.-T. Zheng, Y. Li, and M. Yang, "Modeling relation paths for knowledge graph completion," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 11, pp. 3607–3617, 2021.
[3] Z. Wang, L. Zheng, Y. Li, and S. Wang, "Linkage based face clustering via graph convolution network," in *CVPR*, 2019, pp. 1117–1125.
[4] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, "Graph representation learning via graphical mutual information maximization," in *WWW*, 2020, pp. 259–270.
[5] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 249–270, 2022.
[6] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.
[7] H. Taguchi, X. Liu, and T. Murata, "Graph convolutional networks for graphs containing missing features," *Future Generation Computer Systems*, vol. 117, pp. 155–168, 2021.
[8] X. Chen, S. Chen, J. Yao, H. Zheng, Y. Zhang, and I. W. Tsang, "Learning on attribute-missing graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 2, pp. 740–757, 2022.
[9] F. Monti, M. M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *NeurIPS*, 2017, pp. 3697–3707.
[10] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," in *ArXiv abs/1706.02263*, 2017.
[11] L. Zheng, C.-T. Lu, F. Jiang, J. Zhang, and P. S. Yu, "Spectral collaborative filtering," in *NeurIPS*, 2018, pp. 311–319.
[12] C. Gao, X. He, D. Gan, X. Chen, F. Feng, Y. Li, T. Chua, and D. Jin, "Neural multi-task recommendation from multi-behavior data," in *ICDE*, 2019, p. 1554–1557.
[13] I. Spinelli, S. Scardapane, and A. Uncini, "Missing data imputation with adversarially-trained graph convolutional networks," *Neural Networks*, vol. 129, pp. 249–260, 2020.
[14] J. You, X. Ma, D. Y. Ding, M. J. Kochenderfer, and J. Leskovec, "Handling missing data with graph representation learning," in *NeurIPS*, 2020.
[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[16] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a siamese time delay neural network," in *NeurIPS*, 1993, pp. 737–744.
[17] H. Zhang, D. Liu, and Z. Xiong, "Two-stream action recognition-oriented video super-resolution," in *ICCV*, 2019, pp. 8798–8807.
[18] H. Chen, Y. Wang, K. Zheng, W. Li, C.-T. Chang, A. P. Harrison, J. Xiao, G. D. Hager, L. Lu, C.-H. Liao, and S. Miao, "Anatomy-aware siamese network: Exploiting semantic asymmetry for accurate pelvic fracture detection in x-ray images," in *ECCV*, 2020, pp. 239–255.
[19] Z. Fu, Q. Liu, Z. Fu, and Y. Wang, "Stmtrack: Template-free visual tracking with space-time memory networks," in *CVPR*, 2021, pp. 13 774–13 783.
[20] E. Krivosheev, M. Atzeni, K. Mirylenka, P. Scotton, C. Miksovic, and A. Zorin, "Business entity matching with siamese graph convolutional networks," in *AAAI*, 2021, pp. 16 054–16 056.
[21] M. Jin, Y. Zheng, Y.-F. Li, C. Gong, C. Zhou, and S. Pan, "Multi-scale contrastive siamese networks for self-supervised graph representation learning," in *IJCAI*, 2021, pp. 1477–1483.
[22] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *KDD*, 2014, pp. 701–710.
[23] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks." in *KDD*, 2016, pp. 855–864.
[24] Z. Wang, Z. Li, R. Wang, F. Nie, and X. Li, "Large graph clustering with simultaneous spectral embedding and discretization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 12, pp. 4426–4440, 2021.
[25] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *ArXiv abs/1611.07308*, 2016.

[26] Z. Tao, H. Liu, J. Li, Z. Wang, and Y. Fu, "Adversarial graph embedding for ensemble clustering," in *IJCAI*, 2019, pp. 3562–3568.
[27] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, and P. Cui, "Structural deep clustering network," in *WWW*, 2020, pp. 1400–1410.
[28] W. Tu, S. Zhou, X. Liu, X. Guo, Z. Cai, E. Zhu, and J. Cheng, "Deep fusion clustering network," in *AAAI*, 2021, pp. 9978–9987.
[29] Y. Liu, W. Tu, S. Zhou, X. Liu, L. Song, X. Yang, and E. Zhu, "Deep graph clustering via dual correlation reduction," in *ArXiv abs/2112.14772*, 2021.
[30] R. A. Rossi, R. Zhou, and N. K. Ahmed, "Deep inductive graph representation learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 3, pp. 438–452, 2020.
[31] F. Feng, X. He, J. Tang, and T.-S. Chua, "Graph adversarial training: Dynamically regularizing based on graph structure," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 6, pp. 2493–2504, 2021.
[32] T. Q. Dinh, Y. Xiong, Z. Huang, T. Vo, A. Mishra, W. H. Kim, S. N. Ravi, and V. Singh, "Performing group difference testing on graph structured data from gans: Analysis and applications in neuroimaging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 2, pp. 877–889, 2022.
[33] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *ICLR*, 2019.
[34] K. Hassani and A. H. K. Ahmadi, "Contrastive multi-view representation learning on graphs," in *ICML*, 2020, pp. 4116–4126.
[35] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *NeurIPS*, 2020.
[36] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *WWW*, 2021, pp. 2069–2080.
[37] M. Zhang and Y. Chen, "Inductive matrix completion based on graph neural networks," in *ICLR*, 2020.
[38] J. Chang, L. Wang, G. Meng, Q. Zhang, S. Xiang, and C. Pan, "Local-aggregation graph networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 11, pp. 2874–2886, 2020.
[39] G. Ciano, A. Rossi, M. Bianchini, and F. Scarselli, "On inductive-transductive learning with graph neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 2, pp. 758–769, 2022.
[40] S. Wang, J. Arroyo, J. T. Vogelstein, and C. E. Priebe, "Joint embedding of graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 4, pp. 1324–1336, 2021.
[41] A. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.
[42] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *Information Retrieval*, vol. 29, no. 3, pp. 93–106, 2008.
[43] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," in *ArXiv abs/1811.05868*, 2018.
[44] G. Namata, B. London, L. Getoor, and B. Huang, "Query-driven active surveying for collective classification," in *NeurIPS*, 2012.
[45] Özgür Simsek and D. D. Jensen, "Navigating networks by using homophily and degree," *National Academy of Sciences*, vol. 105, no. 35, pp. 12 758–12 762, 2008.
[46] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.
[47] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017, pp. 1024–1034.
[48] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.
[49] L. Hu, S. Jian, L. Cao, Z. Gu, Q. Chen, and A. Amirbekyan, "Hers: Modeling influential contexts with heterogeneous relations for sparse and cold-start recommendation," in *AAAI*, 2019, pp. 3830–3837.
[50] X. Huang, Q. Song, Y. Li, and X. Hu, "Graph recurrent networks with attributed random walks," in *KDD*, 2019, pp. 732–740.
[51] L. Chen, S. Gong, J. Bruna, and M. M. Bronstein, "Attributed random walk as matrix factorization," in *NeurIPS Workshop*, 2019.

Wenxuan Tu is pursuing his Ph.D. degree in College of Computer, National University of Defense Technology (NUDT), China. His research interests include unsupervised graph learning, deep graph clustering, and image semantic segmentation. He has published several papers in highly regarded journals and conferences such as AAAI, ICML, MM, IEEE T-IP, Information Sciences, etc.

Sihang Zhou received his Ph.D. degree from the National University of Defense Technology (NUDT), China in 2019. He received his M.S. degree in computer science from the same school in 2014 and his bachelor's degree in information and computing science from the University of Electronic Science and Technology of China (UESTC) in 2012. He is now a lecturer of the College of Intelligence Science and Technology, NUDT. His current research interests include machine learning, pattern recognition, and medical image analysis.

Xinwang Liu received his Ph.D. degree from the National University of Defense Technology (NUDT), China. He is now a full professor of the College of Computer, NUDT. His current research interests include kernel learning and unsupervised feature learning. Dr. Liu has published 60+ peer-reviewed papers, including those in highly regarded journals and conferences such as IEEE T-PAMI, IEEE T-KDE, IEEE T-IP, IEEE T-NNLS, IEEE T-MM, IEEE T-IFS, ICML, NeurIPS, ICCV, CVPR, AAAI, IJCAI, etc. He is a senior member of IEEE. More information can be found at https://xinwangliu.github.io.

Yue Liu is pursuing his M.E. degree in College of Computer, National University of Defense Technology (NUDT), China. Her research interests include unsupervised graph learning and multi-view deep graph clustering.

Zhiping Cai received the B.S. M.S. and Ph.D. degrees in computer science and technology from the National University of Defense Technology (NUDT), Changsha, China, in 1996, 2002, and 2005, respectively. He is a Full Professor at the College of Computer, NUDT. His current research interests include artificial intelligence, network security, and big data. Prof. Cai is a Senior Member of CCF.