Information Fusion 131 (2026) 104190

Contents lists available at ScienceDirect

Information Fusion

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

Adaptive virtual anchors for efficient and stable clustering over large multi-view attributed graphs

Mengyao Li [a], Zhibang Yang [b,*], Xu Zhou [a], Joey Tianyi Zhou [c], Quanqing Xu [d], Chuanhui Yang [d], Kenli Li [a], Keqin Li [a,e]

[a] Hunan University, Changsha, 410082, China
[b] Hunan Province Key Laboratory of Industrial Internet Technology and Security, Changsha University, Changsha, 410022, China
[c] A*STAR Centre for Frontier AI Research, 138632, Singapore
[d] OceanBase and the Research Institute of Ant Group, Hangzhou, 310000, China
[e] State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Keywords:
Attributed graph clustering
Community detection
Multi-view attributed graph

ABSTRACT

Multi-view attributed graphs (MVAG) are well-known for their ability to model complex networks and relationships, which can provide diverse yet complementary information for finding a consensus partition suitable for all views. There have been abundant methods for clustering over multi-view attributed graphs. However, most of them are not suitable for large-scale graphs due to high complexity. Moreover, while existing anchor-based methods can effectively accelerate clustering, they mainly focus on either attribute information or graph structure during anchor selection, and some suffer from stability issues. Inspired by this, in this paper, we propose the adaptive virtual anchor clustering method (AVAC) to boost clustering performance and keep stable results. In particular, we first introduce adaptive virtual anchors for multi-view attributed graphs, which are learned and generated from graphs adaptively. After that, we connect anchor learning and anchor graph construction closely and cyclically to learn virtual anchors dynamically and make them capture real data distribution and topology information more accurately. Last but not least, we design a five-block coordinate descent method with proven convergence to further optimize our virtual anchors more representative of existing nodes. Extensive experiments over both real and synthetic datasets demonstrate the effectiveness, efficiency, and stability of our method. Compared to state-of-the-art approaches, the AVAC algorithm always gains stable results with a significant improvement in accuracy, and achieves a speedup of 1.8 times on public large-scale datasets. The source code is available at https://github.com/lmyfree/AVAC.

1. Introduction

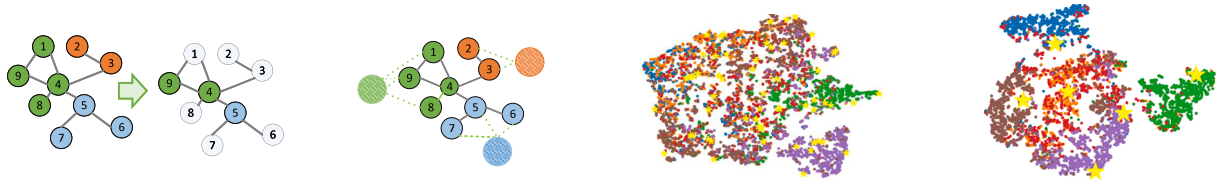
As a prevalent type of graph, attributed graphs [1–4] contribute to modeling complex relationships between various entities with related attributes in social networks [5], citation networks [6], biological networks [7], and so on. Recently, as the network data in real-world applications become typically multi-modal and multi-relational, multi-view attributed graphs [8,9] have garnered significant attention. Multi-view attributed graph clustering [10,11], which aims to seek a unified partition to divide nodes into several disjoint clusters, plays a critical role in many applications.

Application. Clustering on multi-view attributed graphs plays a critical role in recommendation systems [12], social network analysis [11] and other applications. For instance, multi-view attributed graphs in movie recommendation systems can showcase various relationships among movies such as co-actor and co-director relationships, along with related keywords, types, and other attribute features. The integration of diverse views provides more comprehensive information for clustering which can contribute to accurate movie recommendation.

Prior Work and Limitations. Faced with complex multi-view attributed graphs, how to fuse different views is the most imperative problem. At present, existing approaches can be categorized into two classes. The first class endeavors to fuse the attributed graphs of all views into a consistent graph, such as subspace-based methods [10,11,13]. Multi-view attributed graph clustering (MAGC) [11] leverages the self-expressiveness property to combine diverse views into a consistent graph. Multi-view contrastive graph clustering (MCGC) [13] learns a consensus graph regularized by graph contrastive loss. The other class

* Corresponding author.
E-mail addresses: lmy835@hnu.edu.cn (M. Li), yangzb@ccsu.edu.cn (Z. Yang), zhxu@hnu.edu.cn (X. Zhou), joey.tianyi.zhou@gmail.com (J.T. Zhou), xuquanqing.xqq@oceanbase.com (Q. Xu), rizhao.ych@oceanbase.com (C. Yang), lkl@hnu.edu.cn (K. Li), likq@hnu.edu.cn (K. Li).

https://doi.org/10.1016/j.inffus.2026.104190
Received 11 February 2025; Received in revised form 30 December 2025; Accepted 26 January 2026
Available online 28 January 2026



(a) Sampled Anchors (b) Adaptive Virtual Anchors (c) Sampled Anchors in the BlogCatalog Dataset (d) Adaptive Virtual Anchors in the BlogCatalog Dataset

Fig. 1. Traditional anchors **vs** Virtual anchors. Fig. 1(a) shows traditional anchors that are sampled from existing nodes only considering topology. Fig. 1(b) shows adaptive virtual anchors learned and generated from data dynamically. Fig. 1(c) and (d) show anchor embeddings obtained by MvAGC and our method in the BlogCatalog dataset, where yellow stars are anchors.

[14–16] applies graph embedding techniques to transform graph data into low-dimensional feature representations and then clusters on these representations to obtain final results. Fan et al. [17] introduces an innovative autoencoder framework that encompasses an encoder shared across all views for extracting shared representations, along with multiple decoders for reconstructing multi-view graphs.

Although there have been some works [11,13] on clustering multi-view attributed graphs, they also encounter the following limitations:

- **They are not suitable for large-scale graphs due to their high complexity.** In real-life scenarios, graph data becomes increasingly complex and large-scale. For example, the number of nodes in user networks on social media platforms such as Facebook[1] and QQ[2] has already exceeded millions by 2018. However, most methods fail to deal with large-scale graphs due to long running time or memory limitations, as shown in Section 4.4.
- **Existing anchor-based methods can not effectively deal with multi-view attributed graphs.** Although numerous anchor-based methods [18–21] have been developed to reduce running time, they are mainly designed for Euclidean data and focus solely on attributes, making them unsuitable for multi-view attributed graphs, as demonstrated in Section 4.2. Recently, MvAGC [10] introduces an anchor-based method for multi-view attributed graphs, which selects anchors from existing nodes via a sampling algorithm. However, MvAGC only considers graph structure and ignores attribute information during sampling, which directly affects the quality of anchors and clustering accuracy. Moreover, due to its dependence on random sampling algorithms, MvAGC introduces uncertainty in the anchor selection process, leading to unstable anchors that adversely impact model stability. Section 4.5 depicts that the clustering results of MvAGC fluctuate significantly, with evaluation indicators varying by approximately 15%. These unstable results mean that MvAGC requires multiple repeated experiments and additional computational time to achieve optimal outcomes, making it unsuitable for real-world applications.

Challenges. Consequently, how to obtain efficient and stable anchors adaptive for multi-view attributed graphs has become a challenging problem. We summarize posed grim challenges as two aspects. First, it is crucial to design efficient and stable anchors adaptive for multi-view attributed graphs. The quality of anchors directly affects the accuracy of clustering, so how to obtain high-quality anchors that can accurately capture real features and topology structures becomes a core concern. Second, it is significant to design an effective and convergent solution to solve the multivariate optimization problem. Multi-view attributed graph clustering usually can be modeled as a multivariate optimization problem. Different optimization strategies directly affect the optimized results, which can influence the clustering accuracy.

Our Solution. We first introduce adaptive virtual anchors specifically designed for multi-view attributed graphs, more efficient than existing methods. Moreover, to solve the multivariate optimization problem proposed by our method, we design an optimization solver, derived from the Alternating Direction Method of Multipliers (ADMM) [22], specifically customized to address the unique challenges of our method and update adaptive virtual anchors dynamically.

Example 1. Fig. 1 delineates the differences between traditional anchors and our proposed adaptive virtual anchors, using MvAGC as a representative example. Fig. 1(a) depicts the anchors of traditional approach, which are sampled from given nodes only considering topology. For instance, given an attributed graph with nine nodes categorized into three classes marked with different colors, the traditional methods only consider the structural information and randomly sample three nodes v_4, v_5 and v_9 from the nine nodes as anchors that remain fixed in other processes. It fails to fully leverage both attribute and structural information, and the random sampling process is entirely decoupled from the other steps, resulting in unrepresentative anchors that adversely affect accuracy. Fig. 1(b) showcases our adaptive virtual anchors, which are adaptively learned by considering both structure and attribute information and are dynamically updated to capture the true data distribution accurately. Fig. 1(c) and (d) visualize the sampled anchors obtained by MvAGC and the adaptive virtual anchors produced by our method in the BlogCatalog dataset [23] using t-SNE, where “yellow stars” denote the anchors. It is clear that our method provides a sharper distinction between clusters than traditional approaches, with adaptive virtual anchors primarily positioned at cluster centers, showing them more representative than sampled anchors.

Contributions. We propose the adaptive virtual anchor clustering method (AVAC) to maintain high accuracy stably while reducing the running time. AVAC generates adaptive virtual anchors through alternate optimization. In summary, we highlight the contributions as follows:

- We first introduce adaptive virtual anchors for multi-view attributed graph clustering which can be learned and generated adaptively from graphs.
- We propose the adaptive virtual anchor clustering method (AVAC) which generates adaptive virtual anchors via optimization and connects anchor generation and anchor graph construction closely and cyclically to dynamically learn adaptive virtual anchors and capture the real data distribution and graph structure more accurately.
- We design a five-block coordinate descent method to solve the multivariate optimization problem proposed by AVAC, and prove its convergence.
- Extensive experimental results demonstrate that AVAC can obtain stable results with a significant improvement in accuracy while further reducing the time cost of clustering. In particular, on the public large-scale dataset, our algorithm has achieved a speedup of 1.8 times over the advanced method.

The rest of this paper is organized as follows. In Section 2, we present preliminaries and related works. Section 3 describes our method and optimization while analyzing the time complexity and convergence. Then,

[1] https://investor.fb.com/home/default.aspx
[2] https://www.tencent.com/zh-cn/investors/financial-reports.html

we conduct extensive experiments and analyze results in Section 4. Finally, Section 5 concludes this paper.

2. Preliminaries

In this section, we display the definition of multi-view attributed graphs (MVAG) clustering and then introduce related works including attributed graph clustering methods, multi-view subspace methods, multi-view deep learning methods, and anchor-based methods. We summarize frequently used notations in Table 1.

Table 1
Notations and descriptions.

Notation	Description	Notation	Description
n	Number of nodes	$\boldsymbol{X}$	Attribute matrix
λ	Weighting factor	$\boldsymbol{A}$	Adjacency matrix
d	Consensus dimension	h	h-order filter
V	Number of views	m	Number of anchors
K	Number of clusters	α_i	View coefficient
$\boldsymbol{P}$	Consensus anchors	$\boldsymbol{Z}$	Fused anchor graph
$\boldsymbol{W}_v^{(1)}$	Attribute projecting matrix in the v-th view	$\boldsymbol{W}_v^{(2)}$	Topology projecting matrix in the v-th view
$\boldsymbol{S}$	Self-representation matrix	d_v	Dimension in the v-th view

2.1. Problem statement

Definition 1. (Multi-view attributed graph clustering [11])
A multi-view attributed graph can be defined as $G = \{\Phi, E_1, E_2, \dots, E_V, \boldsymbol{X_1}, \boldsymbol{X_2}, \dots, \boldsymbol{X_V}\}$, where Φ represents the node set and $e_{i,j}^v \in E_v$ denotes the relationship between node v_i and v_j in the v-th view. $\boldsymbol{X_v} \in \mathbb{R}^{d_v \times n}$ denotes the attributed matrix for v-th view with d_v dimensions. Furthermore, the structural information can be represented by V adjacency matrices $\{\boldsymbol{A_v}\}_{v=1}^V$, where $\boldsymbol{A_v} = \{a_{i,j}^v\} \in \mathbb{R}^{n \times n}$ and if there exists an edge between nodes v_i and v_j in the v-th view, $a_{i,j}^v = 1$, otherwise, $a_{i,j}^v = 0$. In addition, clustering on the multi-view attributed graph aims to find a unified partition fitting all views to divide the nodes of the graph G into K clusters $(C_1, C_2, \dots, C_K)$.

2.2. Related work

2.2.1. Attributed graph clustering

Attributed graph clustering has garnered sustained attention in recent years. From a task-oriented perspective, it can be broadly divided into graph-level clustering and node-level clustering. The former focuses on partitioning multiple graphs into distinct clusters. Recently, Graph Prompt Clustering (GPC) [24] proposed a "pretraining-prompt-finetuning" framework specifically designed for graph-level clustering. However, in this paper, we focus on node-level clustering, which aims to group nodes into disjoint clusters. Hu et al. [25] constructed proximity matrices from topological connections and employed a Double Visible-Hidden Feature Extraction mechanism for multi-view node clustering. Up to now, existing multi-view attributed graph clustering methods can be divided into several classes: multi-view subspace clustering methods, multi-view deep-learning methods and so on.

2.2.2. Multi-view subspace clustering

The subspace-based methods [11,13,26] hold an assumption that each data sample can be represented as a linear combination of other data samples in the same subspace. Therefore, subspace-based methods for multi-view can be summarized as follows:

$$\min_{\boldsymbol{S}, \alpha_v} \sum_{v=1}^{V} \alpha_v \left(||\boldsymbol{X_v} - \boldsymbol{X_v} \cdot \boldsymbol{S}||_F^2 + \lambda ||\boldsymbol{S} - \boldsymbol{A_v}||_F^2 \right), \tag{1}$$

where $\boldsymbol{X_v}$ and $\boldsymbol{A_v}$ denote the attribute matrix and adjacency matrix for v-th view respectively. α_v is the weight parameter for v-th view. $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ is the self-representation matrix and $|| * ||_F$ represents the Frobenius norm. Eq. (1) attempts to reconstruct the attribute matrix and minimizes the self-reconstruction error. In subspace methods [13], they always cost $\mathcal{O}(n^3)$, a high time complexity, to acquire $\boldsymbol{S}$ via optimization, bringing a grim challenge to large-scale graphs.

2.2.3. Multi-view deep-learning methods

Numerous methods based on deep learning [27–29] have achieved promising results in clustering attributed graphs, however, they fail to deal with multi-view attributed graphs that integrate information from multiple views. Recently, DIAGC [30] and MAGAF [31] introduced the innovative autoencoder framework to cluster multi-view graphs. However, these complex models involve many parameters that require optimization, making them unsuitable for large-scale attributed graphs.

2.2.4. Anchor-based methods

Anchor-based methods aim to reduce computational cost and shorten runtime by introducing representative anchors. In recent years, Zhang et al. [32] found the anchors that satisfy the desired conditions by maximizing the Mahalanobis distance between them. Qin et al. [33] proposed the discriminative anchor learning for multi-view clustering.

Limitations of Anchor-based methods.
Although many Euclidean anchor-based methods [19,20,32,34,35], such as DAGF [36] and DALMC [33], have achieved promising results on Euclidean data, they primarily rely on attributes while overlooking structural information. In contrast, MvAGC [10] selected anchors solely based on graph topology and suffered from instability issues.

In attributed graphs, topology and attributes convey distinct yet complementary information, as shown in Fig. 2. For instance, node 538 and node 1022 exhibit a high cosine similarity of 0.72 but are not linked, whereas node 15 and node 667 are linked yet share a low attribute similarity of only 0.06. These examples highlight anchors for multi-view attributed graphs should be specifically designed which can fuse both topological information and node attributes.

Comparison with Anchor-based methods. In this paper, we propose adaptive virtual anchors that jointly fuse structural and attribute information to generate higher quality anchors. Moreover, by connecting anchor generation and anchor graph construction closely, the optimization problem of our model is more complex compared with other methods [10,19,32,35], for it has five variables with more constraints to optimize. Thus, we design an optimization solver named the five-block coordinate descent optimization to solve it.

3. The proposed methodology

In this section, we first introduce the overall framework of our method (AVAC). Then, we present the important components in our method. Finally, we depict our method in algorithm and provide its time complexity, convergence analysis, and memory optimization discussion.

3.1. Overall framework

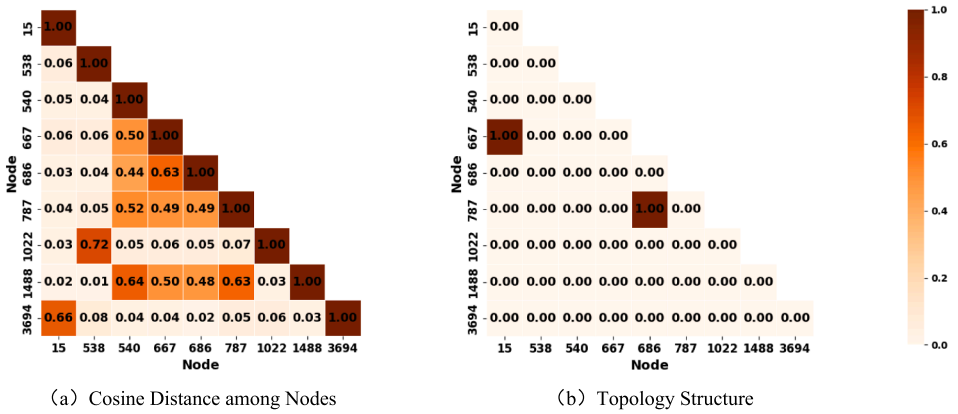
AVAC includes four stages: data preprocessing, adaptive virtual anchor generation, adaptive virtual anchor graph construction, and clustering, as shown in Fig. 3.

Data Preprocessing. In Stage A, in the data preprocessing, we adopt the h-order graph filter [11] to preprocess the multi-view attributed graph and obtain smooth representations,

$$\hat{\boldsymbol{X}} = \left(1 - \frac{1}{2}\boldsymbol{L}\right)^h \boldsymbol{X}, \tag{2}$$

where h is a non-negative integer and $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{A}$ denotes the normalized graph Laplacian.

Adaptive virtual Anchor Generation.



(a) Cosine Distance among Nodes (b) Topology Structure

Fig. 2. Inconsistency between graph attributes and topology information.

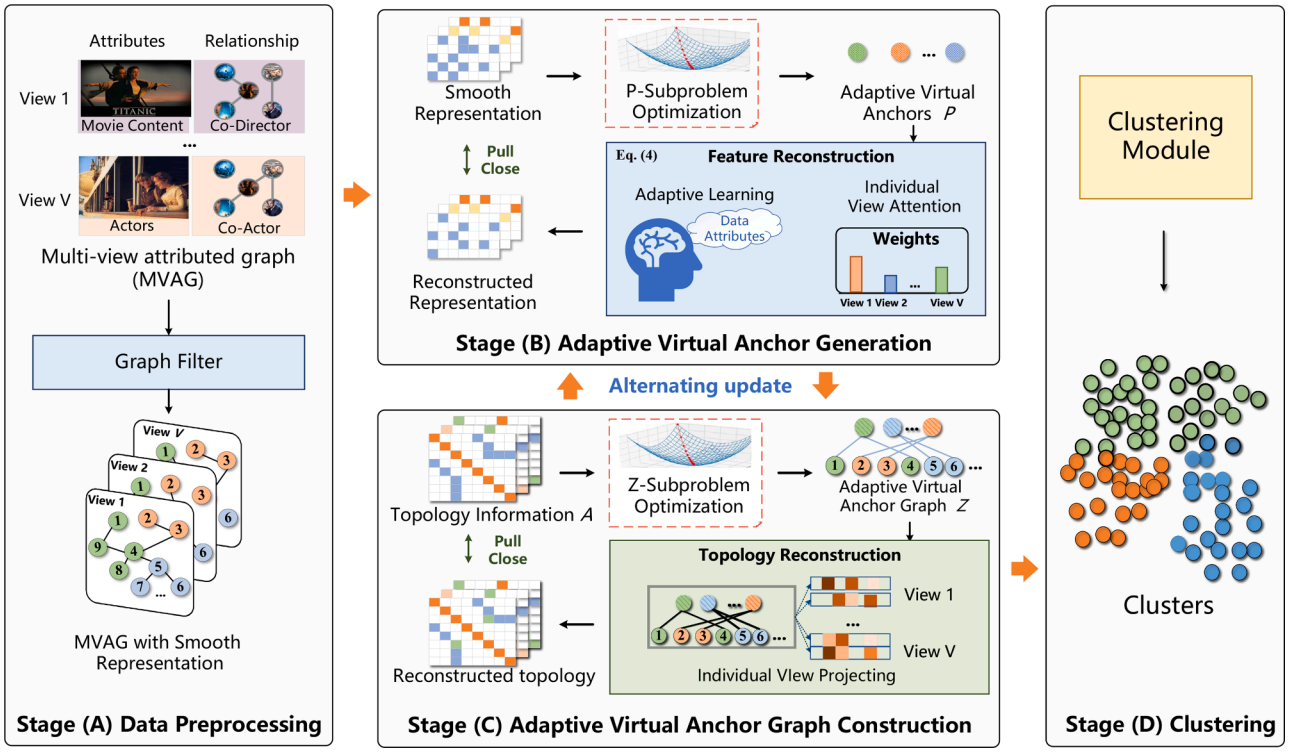


Fig. 3. Framework of the AVAC method. Our method includes four stages: data preprocessing, adaptive virtual anchor generation, adaptive virtual anchor graph construction, and clustering. AVAC generates adaptive virtual anchors and connects anchor generation and anchor graph construction closely and cyclically to capture topology and real distribution of attributes.

In Stage B, we generate adaptive virtual anchors to effectively represent all nodes. To ensure that these anchors accurately reflect the real distribution of the given graph, we design an optimization solver (Section 3.2.3) to update the anchors.

Adaptive virtual Anchor Graph Construction. In Stage C, we construct an adaptive virtual anchor graph that captures the similarities between nodes and anchors. To ensure that the anchor graph accurately captures the true topology of the original graph, an optimization solver (Section 3.2.3) is designed to optimize it.

Clustering. After obtaining the adaptive virtual anchors $\boldsymbol{P}$ and the virtual anchor graph $\boldsymbol{Z}$ from the optimization solver, we perform SVD on the virtual anchor graph $\boldsymbol{Z}$ to get the left singular vector $\boldsymbol{U}$ as the subspace representation. Finally, we perform the clustering algorithm like K-means on the subspace representation $\boldsymbol{U}$ to get the clustering results.

Overall, we depict the general framework in brief, as shown in Algorithm 1. We perform Stage B and Stage C repeatedly to update the adaptive virtual anchors and the adaptive virtual anchor graph dynamically.

3.2. Core components of the proposed method

In this section, we will introduce three important components: adaptive virtual anchor generation, adaptive virtual anchor graph construction, and the five-block coordinate descent optimization. In AVAC, we adaptively learn virtual anchors by projecting virtual anchors to different views to reconstruct attribute matrices and adjacency matrices, thereby obtaining a consensus anchor graph with attribute information and topology information of all views. The details are as follows.

Algorithm 1: Overall framework of AVAC.

Input: Multi-view attributed graph, the number of cluster K.
Output: K clusters.

Stage A: Data Processing.
Perform the graph filter to obtain smooth representations.
while *not converged* **do**
  Stage B: Adaptive Virtual Anchor Generation.
  Utilize the individual view attention mechanism to adaptively learn virtual anchors. (Subsection 3.2.1).
  Design the solver optimization to obtain the updated adaptive virtual anchors (Subsection 3.2.3).
  Stage C: Adaptive Virtual Anchor Graph Construction.
  Construct the adaptive virtual anchor graph via individual view projecting (Subsection 3.2.2). Design the optimization solver to obtain the updated adaptive virtual anchor graph (Subsection 3.2.3).
end
Stage D: Clustering.
Perform the clustering algorithm to obtain clusters. **return** K *clusters.*

3.2.1. Adaptive virtual anchor generation

The adaptive virtual anchor generation process, illustrated in Stage B of Fig. 3, comprises two main components: adaptive learning and individual view attention.

- **Adaptive learning.** To generate adaptive virtual anchors, we project adaptive virtual anchors $\boldsymbol{P} \in \mathbb{R}^{d\times m}$ to the real feature space $\hat{\boldsymbol{X}}_v \in \mathbb{R}^{d_v\times n}$ of each view, where d and m are consensus dimension and number of anchors. By minimizing the distance between reconstructed features and original features, we can adaptively learn representative virtual anchors that can effectively capture the real distribution. We formulate this objective function as:

$$\begin{aligned} &\min_{\boldsymbol{Z},\boldsymbol{P},\boldsymbol{W}_v^{(1)}} ||\hat{\boldsymbol{X}}_v - \boldsymbol{W}_v^{(1)}\boldsymbol{P}\boldsymbol{Z}||_F^2, \\ &s.t. \quad \boldsymbol{W}_v^{(1)^T}\boldsymbol{W}_v^{(1)} = \boldsymbol{I}, \boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{I} \end{aligned} \tag{3}$$

  where $\boldsymbol{P}$ is the consistent anchor and $\boldsymbol{W}_v^{(1)} \in \mathbb{R}^{d_v\times d}$ denotes a projecting matrix that can map the consensus dimension d to the dimensions of attributes in the v-th view. $\boldsymbol{Z}$ is the anchor graph that indicates the similarity among original nodes and anchors.
- **Individual View Attention.** Since different views play various roles in the clustering task, we introduce the view-attention mechanism [18] in our AVAC method. In view-attention mechanism, we set a series of parameters $\{\alpha_v\}_{v=1}^V$ to pay different attention to individual views, and then optimize them through the optimization solver. Thus, Eq. (3) can be further developed as:

$$\begin{aligned} &\min_{\boldsymbol{W}_v^{(1)},\boldsymbol{Z},\boldsymbol{P},\alpha_v} \sum_{v=1}^{V} \alpha_v^2 \left(||\hat{\boldsymbol{X}}_v - \boldsymbol{W}_v^{(1)}\boldsymbol{P}\boldsymbol{Z}||_F^2 \right), \\ &s.t. \sum_{v=1}^{V} \alpha_v = 1, \boldsymbol{W}_v^{(1)^T}\boldsymbol{W}_v^{(1)} = \boldsymbol{I}, \boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{I} \end{aligned} \tag{4}$$

  where α_v is the attention coefficient of the v-th view. $\boldsymbol{Z} \in \mathbb{R}^{m\times n}$ denotes the anchor graph which reflects the similarity relationships among anchors and nodes and m is the number of anchors. $\boldsymbol{P} \in \mathbb{R}^{d\times m}$ represents the consensus anchors for all views, and d is the consensus dimension. The individual view attention mechanism can selectively absorb useful information from each view and combine the complementarity and consensus information among these views more effectively.

3.2.2. Adaptive virtual anchor graph construction

We project the anchor graph into the real topological space in each view to reconstruct the topological information and update the anchor graph by reducing the distance between the reconstructed adjacency matrix and the original adjacency matrix, as shown in the Stage C of Fig. 3. Meanwhile, we can reconstruct the original feature by projecting the adaptive virtual anchor to the real feature space via the anchor graph. Furthermore, through the individual view attention, we can assign different weights to each view to combine these views effectively. In summary, we obtain the final formulation of our problem:

$$\begin{aligned} &\min_{\substack{\boldsymbol{W}_v^{(1)},\boldsymbol{W}_v^{(2)} \\ \boldsymbol{Z},\boldsymbol{P},\alpha_v}} \sum_{v=1}^{V} \alpha_v^2 \Big(\underbrace{||\hat{\boldsymbol{X}}_v - \boldsymbol{W}_v^{(1)}\boldsymbol{P}\boldsymbol{Z}||_F^2}_{\text{Feature Reconstruction}} + \lambda \underbrace{||\boldsymbol{A}_v - \boldsymbol{W}_v^{(2)}\boldsymbol{Z}||_F^2}_{\text{Topology Reconstruction}} \Big), \\ &s.t. \sum_{v=1}^{V} \alpha_v = 1, \boldsymbol{W}_v^{(1)^T}\boldsymbol{W}_v^{(1)} = \boldsymbol{I}, \boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{I}, \boldsymbol{W}_v^{(2)^T}\boldsymbol{W}_v^{(2)} = \boldsymbol{I} \end{aligned} \tag{5}$$

where $\boldsymbol{Z}$ is the anchor graph which reflects the similarity score between node representations and anchors. The smaller it is, the less relevant it is between the node and the adaptive virtual anchors [10,11]. $\boldsymbol{W}_v^{(2)} \in \mathbb{R}^{n\times m}$ is a projecting matrix that maps the consistent anchor graph $\boldsymbol{Z} \in \mathbb{R}^{m\times n}$ to the original space of adjacency information in the v-th view. We constrain $\boldsymbol{W}_v^{(1)}, \boldsymbol{W}_v^{(2)}, \boldsymbol{P}$ to be orthogonal to utilize relevant properties for rapid derivation [18]. α_v is the attention coefficient of the v-th view, and we set some constraints for it to better balance the weight ratio among different views. λ is the weight factor to balance between the structural information and attribute information.

To solve Eq. (5), we design the optimization solver named the five-block coordinate descent optimization described in Section 3.2.3 to obtain the adaptive virtual anchors $\boldsymbol{P}$ and the virtual anchor graph $\boldsymbol{Z}$.

3.2.3. The five-block coordinate descent optimization

In this subsection, we describe the optimization solver named the five-block coordinate descent optimization, which is utilized to obtain the adaptive virtual anchors $\boldsymbol{P}$ and the virtual anchor graph $\boldsymbol{Z}$. Five-Block Coordinate Descent Optimization method is a customized solver, derived from Alternating Direction Method of Multipliers (ADMM) [22], but specifically designed to tackle the specific and complex multivariable optimization problem in this paper. When optimizing each subproblem, our solver effectively leverages the relationship between the Frobenius Norm and the matrix trace to streamline the computational process. Further, confronted with the specific constraints in AVAC, we incorporate singular value decomposition, the Schwarz-Cauchy inequality, and other analytical techniques to enhance the optimization, ensuring an efficient and effective resolution of the complex multivariate optimization problem in this paper.

There are five groups of variables in Eq. (5), and when considering all variables simultaneously, Eq. (5) is not jointly convex. Thus, we design the five-block coordinate descent optimization to solve this multivariate optimization problem. We can divide the multivariate optimization problem into five subproblems and minimize Eq. (5) by solving the following five subproblems iteratively:

- Attribute projecting matrix $\boldsymbol{W}_v^{(1)}$ -Subproblem. Fix $\boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{W}_v^{(2)}$, α_v , and update the attribute projecting matrix $\boldsymbol{W}_v^{(1)}$.
- Topology projecting matrix $\boldsymbol{W}_v^{(2)}$ -Subproblem. Fix $\boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{W}_v^{(1)}$, α_v , and update the topology projecting matrix $\boldsymbol{W}_v^{(2)}$.
- Adaptive virtual anchors $\boldsymbol{P}$ -Subproblem. Fix $\boldsymbol{Z}, \boldsymbol{W}_v^{(1)}, \boldsymbol{W}_v^{(2)}$, α_v , and update adaptive virtual anchors $\boldsymbol{P}$.
- Adaptive virtual Anchor graph $\boldsymbol{Z}$ -Subproblem. Fix $\boldsymbol{P}, \boldsymbol{W}_v^{(1)}, \boldsymbol{W}_v^{(2)}$, α_v , and update the anchor graph $\boldsymbol{Z}$.
- Attention for individual views α_v -Subproblem. Fix $\boldsymbol{Z}, \boldsymbol{W}_v^{(1)}, \boldsymbol{W}_v^{(2)}, \boldsymbol{P}$, and update the attention score for each view α_v .

The five subproblems are solved as follows iteratively.

Subproblem-1. Attribute projecting matrix $\boldsymbol{W}_v^{(1)}$ -Subproblem. Fix $\boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{W}_v^{(2)}$, α_v , and the optimization function can be presented as

$$\begin{aligned} &\min_{\boldsymbol{W}_v^{(1)}} \sum_{v=1}^{V} \alpha_v^2 \left(||\hat{\boldsymbol{X}}_v - \boldsymbol{W}_v^{(1)} \boldsymbol{P} \boldsymbol{Z}||_F^2 \right), \\ &s.t. \boldsymbol{W}_v^{(1)^T} \boldsymbol{W}_v^{(1)} = \boldsymbol{I}. \end{aligned} \tag{6}$$

For each view, the value of $\boldsymbol{W}_v^{(1)}$ is independent of other views, therefore, according to relationships between the Frobenius norm and matrix trace, we extend the Frobenius norm, delete some irrelevant items, and transform Eq. (6) as

$$\begin{aligned} &\min_{\boldsymbol{W}_v^{(1)}} \sum_{v=1}^{V} \alpha_v^2 \left(||\hat{\boldsymbol{X}}_v - \boldsymbol{W}_v^{(1)} \boldsymbol{P} \boldsymbol{Z}||_F^2 \right), \\ &s.t. \quad \boldsymbol{W}_v^{(1)^T} \boldsymbol{W}_v^{(1)} = \boldsymbol{I}, \\ \Leftrightarrow &\min_{\boldsymbol{W}_v^{(1)}} \sum_{v=1}^{V} \alpha_v^2 Tr\left(\hat{\boldsymbol{X}}_v^T \hat{\boldsymbol{X}}_v - 2 \hat{\boldsymbol{X}}_v^T \boldsymbol{W}_v^{(1)} \boldsymbol{P} \boldsymbol{Z} + \boldsymbol{Z}^T \boldsymbol{P}^T \boldsymbol{P} \boldsymbol{Z} \right) \\ \Leftrightarrow &\min_{\boldsymbol{W}_v^{(1)}} \sum_{v=1}^{V} \alpha_v^2 Tr\left(-2 \hat{\boldsymbol{X}}_v^T \boldsymbol{W}_v^{(1)} \boldsymbol{P} \boldsymbol{Z} \right) \\ \Leftrightarrow &\max_{\boldsymbol{W}_v^{(1)}} Tr\left(\boldsymbol{W}_v^{(1)^T} \boldsymbol{C} \right), \end{aligned} \tag{7}$$

where $\boldsymbol{C}$ represents $\hat{\boldsymbol{X}}_v \boldsymbol{Z}^T \boldsymbol{P}^T$. Assuming that the singular value decomposition (SVD) of $\boldsymbol{C}$ is $\boldsymbol{U}_c \boldsymbol{\Sigma}_c \boldsymbol{R}_c^T$, $\boldsymbol{W}_v^{(1)}$ should be $\boldsymbol{U}_c \boldsymbol{R}_c^T$ based on Proposition 1.

Proposition 1. *When the singular value decomposition of $\boldsymbol{C}$ is $\boldsymbol{C} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{R}^T$, the following constrained problem has closed form solution $\boldsymbol{M} = \boldsymbol{U} \boldsymbol{R}^T$.*

$$\max_{\boldsymbol{M}} Tr(\boldsymbol{M}^T \boldsymbol{C}), \text{ s.t. } \quad \boldsymbol{M}^T \boldsymbol{M} = \boldsymbol{I}. \tag{8}$$

Proof. Assuming the singular value decomposition of $\boldsymbol{C}$ is $\boldsymbol{C} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{R}^T$, we can obtain

$$Tr(\boldsymbol{M}^T \boldsymbol{C}) = Tr(\boldsymbol{M}^T \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{R}^T) = Tr(\boldsymbol{R}^T \boldsymbol{M}^T \boldsymbol{U} \boldsymbol{\Sigma}) \tag{9}$$

Let $\boldsymbol{Q} = \boldsymbol{R}^T \boldsymbol{M}^T \boldsymbol{U}$, it is evident that $\boldsymbol{R}^T \boldsymbol{M}^T \boldsymbol{U} \boldsymbol{U}^T \boldsymbol{M} \boldsymbol{R} = \boldsymbol{I}$. Then we can get $Tr(\boldsymbol{R}^T \boldsymbol{M}^T \boldsymbol{U} \boldsymbol{\Sigma}) = Tr(\boldsymbol{Q} \boldsymbol{\Sigma}) \leq Tr(\boldsymbol{I} \boldsymbol{\Sigma}) = \sum_{i=1}^{t} \sigma_i$, where σ_i is the *i-th* diagonal element of Σ. Thus, when $\boldsymbol{Q} \boldsymbol{\Sigma} = \boldsymbol{I} \boldsymbol{\Sigma}$, indicating $\boldsymbol{R}^T \boldsymbol{M}^T \boldsymbol{U} = \boldsymbol{I}$, Eq. (8) achieves the maximum, and we get the closed solution $\boldsymbol{M} = \boldsymbol{U} \boldsymbol{R}^T$. □

Subproblem-2. Topology projecting matrix $\boldsymbol{W}_v^{(2)}$ -Subproblem. The optimization of $\boldsymbol{W}_v^{(2)}$ is similar to $\boldsymbol{W}_v^{(1)}$. When $\boldsymbol{Z}, \boldsymbol{P}, \boldsymbol{W}_v^{(1)}, \alpha_v$ are fixed, Eq. (5) can be transformed into the following problem

$$\begin{aligned} &\min_{\boldsymbol{W}_v^{(2)}} \sum_{v=1}^{V} \alpha_v^2 \left(\lambda ||\boldsymbol{A}_v - \boldsymbol{W}_v^{(2)} \boldsymbol{Z}||_F^2 \right), \\ &s.t. \boldsymbol{W}_v^{(2)^T} \boldsymbol{W}_v^{(2)} = \boldsymbol{I}. \end{aligned} \tag{10}$$

According to the relationship between the Frobenius norm and matrix trace, we can change Eq. (10) into

$$\max_{\boldsymbol{W}_v^{(2)}} Tr(\boldsymbol{W}_v^{(2)^T} \boldsymbol{B}), \quad s.t. \quad \boldsymbol{W}_v^{(2)^T} \boldsymbol{W}_v^{(2)} = \boldsymbol{I}, \tag{11}$$

where $\boldsymbol{B}=\boldsymbol{A}_v \boldsymbol{Z}^T$, therefore $\boldsymbol{W}_v^{(2)}$ equals $\boldsymbol{U}_b \boldsymbol{R}_b^T$ when $\boldsymbol{B} = \boldsymbol{U}_b \boldsymbol{\Sigma}_b \boldsymbol{R}_b^T$.

Subproblem-3. Adaptive virtual anchors $\boldsymbol{P}$ -Subproblem. When $\boldsymbol{Z}, \boldsymbol{W}_v^{(1)}, \boldsymbol{W}_v^{(2)}, \alpha_v$ fixed, Eq. (5) can be transformed into the following formula

$$\min_{\boldsymbol{P}} \sum_{v=1}^{V} \alpha_v^2 \left(||\hat{\boldsymbol{X}}_v - \boldsymbol{W}_v^{(1)} \boldsymbol{P} \boldsymbol{Z}||_F^2 \right), \quad s.t. \quad \boldsymbol{P}^T \boldsymbol{P} = \boldsymbol{I}. \tag{12}$$

Moreover, Eq. (12) can be rewritten as

$$\begin{aligned} &\min_{\boldsymbol{P}} \sum_{v=1}^{V} \alpha_v^2 \left(||\hat{\boldsymbol{X}}_v - \boldsymbol{W}_v^{(1)} \boldsymbol{P} \boldsymbol{Z}||_F^2 \right), \\ \Leftrightarrow &\min_{\boldsymbol{P}} \sum_{v=1}^{V} \alpha_v^2 Tr\left(\hat{\boldsymbol{X}}_v^T \hat{\boldsymbol{X}}_v - 2 \boldsymbol{P}^T \boldsymbol{W}_v^{(1)^T} \hat{\boldsymbol{X}}_v \boldsymbol{Z}^T + \boldsymbol{Z}^T \boldsymbol{Z} \right) \\ \Leftrightarrow &\max_{\boldsymbol{P}} \sum_{v=1}^{V} \alpha_v^2 Tr\left(\boldsymbol{P}^T \boldsymbol{W}_v^{(1)^T} \hat{\boldsymbol{X}}_v \boldsymbol{Z}^T \right) \\ \Leftrightarrow &\max_{\boldsymbol{P}} Tr\left(\boldsymbol{P}^T \boldsymbol{F} \right), \ s.t. \quad \boldsymbol{P}^T \boldsymbol{P} = \boldsymbol{I}, \end{aligned} \tag{13}$$

where $\boldsymbol{F} = \sum_{v=1}^{V} \alpha_v^2 \boldsymbol{W}_v^{(1)^T} \hat{\boldsymbol{X}}_v \boldsymbol{Z}^T$. Supposing optimization results of $\boldsymbol{F}$ is $\boldsymbol{U}_f \boldsymbol{\Sigma}_f \boldsymbol{R}_f^T$, $\boldsymbol{P}$ equals to $\boldsymbol{U}_f \boldsymbol{V}_f^T$.

Subproblem-4. Adaptive virtual Anchor graph $\boldsymbol{Z}$ -Subproblem. When we fix $\boldsymbol{P}, \boldsymbol{W}_v^{(1)}, \boldsymbol{W}_v^{(2)}$, α_v , we can transform Eq. (5) based on the relationship between the Frobenius norm and matrix trace. Then we can delete some terms unrelated to $\boldsymbol{Z}$, which have no effect on the first derivative of the equation. Finally, we set the first derivative of the equation to zero to obtain the optimization result. Therefore, we have

$$\begin{aligned} &\frac{\partial \sum_{v=1}^{V} \alpha_v^2 \left(||\hat{\boldsymbol{X}}_v - \boldsymbol{W}_v^{(1)} \boldsymbol{P} \boldsymbol{Z}||_F^2 + \lambda ||\boldsymbol{A}_v - \boldsymbol{W}_v^{(2)} \boldsymbol{Z}||_F^2 \right)}{\partial \boldsymbol{Z}} \\ \Leftrightarrow &\frac{\partial \sum_{v=1}^{V} \alpha_v^2 Tr\left[(\boldsymbol{I} + \lambda \boldsymbol{I}) \boldsymbol{Z}^T \boldsymbol{Z} - 2 \boldsymbol{Z}^T \boldsymbol{P}^T \boldsymbol{W}_v^{(1)^T} \hat{\boldsymbol{X}}_v \right]}{\partial \boldsymbol{Z}} \\ &- \frac{\partial \sum_{v=1}^{V} \alpha_v^2 Tr\left[2 \lambda \boldsymbol{Z}^T \boldsymbol{W}_v^{(2)^T} \boldsymbol{A}_v \right]}{\partial \boldsymbol{Z}}. \end{aligned} \tag{14}$$

When setting Eq. (15) to 0, we can get the optimization as

$$\boldsymbol{Z} = \left[\sum_{v=1}^{V} \alpha_v^2 (\boldsymbol{I} + \lambda \boldsymbol{I}) \right]^{-1} \times \left[\sum_{v=1}^{V} \alpha_v^2 \left(\boldsymbol{P}^T \boldsymbol{W}_v^{(1)^T} \hat{\boldsymbol{X}}_v + \lambda \boldsymbol{W}_v^{(2)^T} \boldsymbol{A}_v \right) \right]. \tag{15}$$

Subproblem-5. Attention for individual views α_v-Subproblem. When updating the attention score α_v, we keep $\boldsymbol{Z}, \boldsymbol{W}_v^{(1)}, \boldsymbol{W}_v^{(2)}, \boldsymbol{P}$ fixed. Marking $\left(||\hat{\boldsymbol{X}}_v - \boldsymbol{W}_v^{(1)} \boldsymbol{P} \boldsymbol{Z}||_F^2 + \lambda ||\boldsymbol{A}_v - \boldsymbol{W}_v^{(2)} \boldsymbol{Z}||_F^2 \right)$ as H, Eq. (5) can be transformed as

$$\sum_{v=1}^{V} \alpha_v^2 H_v^2 \Leftrightarrow \frac{1}{V} \sum_{v=1}^{V} (\alpha_v H_v)^2 \sum_{v=1}^{V} 1^2 \geq \frac{1}{V} \left(\sum_{v=1}^{V} \alpha_v H_v \right)^2. \tag{16}$$

According to the conditions for equality, we can obtain the optimal results when $\alpha_1 H_1 = \alpha_2 H_2 = \ldots = \alpha_V H_V$, and we set $L = \alpha_1 H_1$. According to Schwarz Cauchy inequality, we can acquire $\alpha_v = \frac{L}{H_v}$ where $L = \frac{1}{\frac{1}{H_1} + \frac{1}{H_2} + \ldots + \frac{1}{H_V}}$.

The designed optimization solver (the five-block coordinate descent optimization) divides the multivariate optimization problem into five sub-problems. By solving these subproblems iteratively, we can obtain the updated virtual anchors $\boldsymbol{P}$ and the virtual anchor graph $\boldsymbol{Z}$.

3.3. Algorithm summary and theoretical analysis

In this subsection, we conclude our adaptive virtual anchor clustering algorithm and further analyze its time complexity and convergence. Moreover, we discuss our memory optimization techniques (matrix chunking multiplication) to further reduce memory requirements when dealing with large graphs. The complete procedures of AVAC are outlined in Algorithm 2. As depicted, we initialize $\boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{W}_v^{(1)}, \boldsymbol{W}_v^{(2)}$ with zero matrices. First, we preprocess the multi-view attributed graph via the *h*-order graph filter to obtain smooth representations. Then, we update $\boldsymbol{W}_v^{(1)}, \boldsymbol{W}_v^{(2)}$ to prepare the attributed projecting matrix and topology projecting matrix. After that, we can update the adaptive virtual anchors by solving the $\boldsymbol{P}$ -Subproblem to obtain the optimized virtual anchors. Then, we update $\boldsymbol{Z}, \alpha$ to obtain the adaptive virtual anchor graph. We

Algorithm 2: The AVAC algorithm.

Input: Multi-view attributed graph $\{\boldsymbol{A}_1,\ldots,\boldsymbol{A}_V,\boldsymbol{X}_1,\ldots,\boldsymbol{X}_V\}$, the number of cluster K.
Output: K clusters.
Initialization: Initialize $\boldsymbol{P},\boldsymbol{Z},\boldsymbol{W}_v^{(1)},\boldsymbol{W}_v^{(2)}$ with zero matrix. Initialize α_v with $\frac{1}{v}$.
Perform the h-order graph filter to obtain the smooth representation of multi-view attributed graph $\{\boldsymbol{A}_1,\ldots,\boldsymbol{A}_V,\hat{\boldsymbol{X}}_1,\ldots,\hat{\boldsymbol{X}}_V\}$.
while *not converged* **do**
  Update $\boldsymbol{W}_v^{(1)}$ by solving the attribute projecting matrix $\boldsymbol{W}_v^{(1)}$ -Subproblem.
  Update $\boldsymbol{W}_v^{(2)}$ by solving the topology projecting matrix $\boldsymbol{W}_v^{(2)}$ -Subproblem.
  Update $\boldsymbol{P}$ by solving adaptive virtual anchors $\boldsymbol{P}$ -Subproblem.
  Update $\boldsymbol{Z}$ by solving the anchor graph $\boldsymbol{Z}$ -Subproblem.
  Update α_v by solving the attention score for individual views α_v-Subproblem.
end
Perform SVD on $\boldsymbol{Z}$ to obtain the left singular vector $\boldsymbol{U}$.
Perform clustering algorithm on $\boldsymbol{U}$ to get the K clusters.
return *K clusters.*

can adaptively learn virtual anchors and construct the virtual anchor graph by minimizing Eq. (5). This process effectively captures the underlying distribution and topological information by reconstructing both the attribute matrix and the adjacency matrix. Moreover, we execute Stages B and C alternately to dynamically update the virtual anchors and the virtual anchor graph. After obtaining the anchor graph $\boldsymbol{Z}$, we perform SVD on the anchor graph to acquire the left singular vector $\boldsymbol{U}$. Finally, we execute K-means on the eigenvalue $\boldsymbol{U}$ to get the clustering results.

Time complexity. For AVAC, the cost of optimization for each variable composes the overall computational complexity. When updating $\boldsymbol{W}_v^{(1)}$, the implementation of SVD on $\boldsymbol{C}_v$ takes $\mathcal{O}(d_v d^2)$, and it costs $\mathcal{O}(d_v d k^2)$ for other matrix multiplication. When updating $\boldsymbol{W}_v^{(2)}$, it costs $\mathcal{O}(m^2 n)$ to execute SVD on $\boldsymbol{B}_v$.

When updating $\boldsymbol{Z}$, the inverse operation costs $\mathcal{O}(m^3)$. Since the adjacency matrix is sparse, assuming that the number of non-zero elements in the matrix is n_z, where n_z is much smaller than n^2, the other matrix multiplications cost $\mathcal{O}(md d_v+md_v n+n_z+m^2 n)$. When updating $\boldsymbol{P}$, it takes $\mathcal{O}(md^2)$ for SVD and $\mathcal{O}(dmk^2)$ for matrix multiplications. Furthermore, it costs $\mathcal{O}(1)$ when calculating α_v. In summary, we can find the complexity of AVAC is with respect to $\mathcal{O}(n_z)$, which is far less than n^2. To further improve the efficiency of AVAC, we can consider the adjacency matrix as n vectors, and implement the matrix multiplication in parallel, reducing the time complexity of AVAC to $\mathcal{O}(n)$.

Convergence analysis. We design the five-block coordinate descent optimization for the AVAC method. It monotonically decreases during the iteration process when one subproblem is solved with the others fixed at each iteration. Moreover, we can observe that the lower bound of Eq. (5) is 0. Thus, according to [37], the algorithm can be guaranteed to converge.

Discussion. As the graph size increases, we find that the memory requirement of clustering is increasing. Through experiments, it is clear that matrix multiplications consume the most memory. To better adapt to large graph scenarios, we design the matrix chunking multiplication for AVAC to reduce the memory requirement. As shown in Fig. 4, we divide the original matrix into nine sub-matrices. We decompose large matrix multiplications into several smaller block multiplications. The first block of results ($\boldsymbol{A}\boldsymbol{A}'+\boldsymbol{B}\boldsymbol{D}'+\boldsymbol{C}\boldsymbol{G}'$) is computed by multiplying the first row of matrix $\boldsymbol{L}$ with the first column of matrix $\boldsymbol{M}$. The size of the sub-matrices is determined based on the memory capacity of the running environment.

Table 2
Multi-view attributed graph datasets.

Dataset	Views	Nodes	Attributes	Edges	Clusters
ACM	2	3,025	1,830	29,281 2,210,761	3
DBLP	3	4,057	334	11,113 5,000,495 6,776,335	4
BlogCatalog	2	5,196	8,189 5,196	171,743	6
Flickr	2	7,575	12,047 7,575	479,476	9
AMAP	2	7,650	745 7,650	119,081	8
Wiki	4	2,405	4,973	24,357 12,025	17
Pubmed	2	19,717	500	44,338	3
Computer	2	13,752	767	133,289	10
Ogbn-products	2	2,449,029	100	61,859,140	47

4. Experiment

In this section, we evaluate AVAC with a comparison to the state-of-the-art algorithms over real and synthetic datasets.

4.1. Experimental setup

In this subsection, we introduce metrics, datasets, and comparison algorithms, respectively.

Metrics and Datasets. We use four evaluation metrics to measure clustering results: accuracy (ACC), F1-score (F1), normalized mutual information (NMI), and adjusted rand index (ARI) [11].

Moreover, in the experiments, we use nine datasets covering three types of multi-view attributed graphs. The detailed information is shown in Table 2.

- **Multi-view attributed graph datasets with multiple graph structures.** We implement two real datasets (ACM[3], DBLP[4]) which contain one attributed matrix and multiple adjacency matrices.
- **Multi-view with multiple attribute matrices.** Blogcatalog dataset [23], Pubmed dataset [38], Computer dataset,[5] AMAP dataset[6], Flickr dataset [39] and the Ogbn-products dataset [40] contain one adjacency matrix and multiple attribute matrices. Moreover, the attribute matrix of the Blogcatalog, Flickr, and AMAP datasets in the second view is constructed via a cartesian product [11]. In the Pubmed, Computer and Ogbn-products datasets, the attribute matrix of the second view is constructed using a log-scale of the original ones [41].
- **Multi-view attributed graph datasets with multiple attributes and graph structures.** The Wiki dataset [41] contains multiple attribute matrices and multiple adjacency matrices. In the Wiki dataset, the additional views are created from the initial data which contains a single graph structure and attribute matrix.

Comparison Algorithms. We compare AVAC with several representative works in recent years from the perspectives of accuracy and efficiency on nine datasets, which contain different types of multi-view attributed graphs. The comparison algorithms mainly include the following four categories:

[3] http://dl.acm.org
[4] https://dblp.uni-trier.de/
[5] https://github.com/Karenxt/AGCandIAGC-code/
[6] https://github.com/yueliu1999/Awesome-Deep-Graph-Clustering/

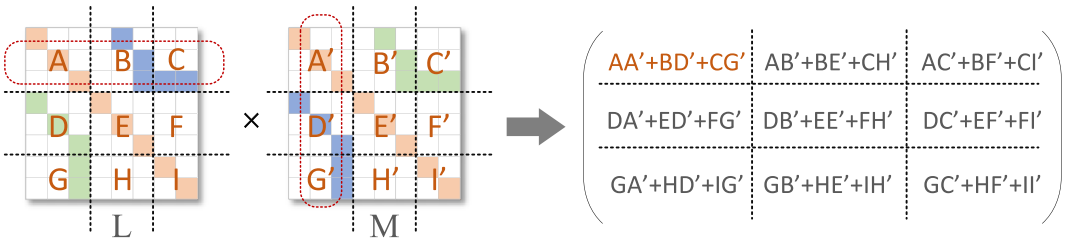


Fig. 4. Matrix chunking multiplication.

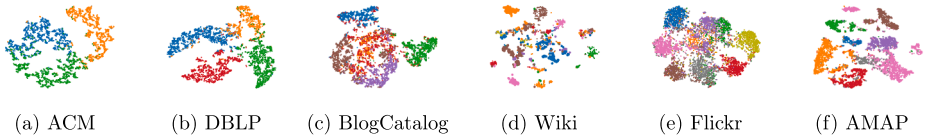


Fig. 5. Two-dimensional projections of AVAC embeddings using t-SNE colored according to real labels.

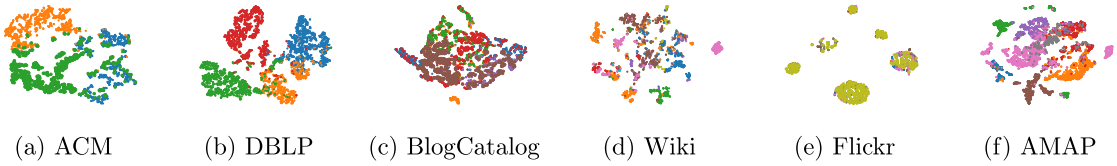


Fig. 6. Two-dimensional projections of DIAGC embeddings using t-SNE colored according to real labels.

- **Subspace clustering methods:** PwMC [42] and MAGC [11].
- **Single-view methods:** SDCN [43], SDCN-avg, AGCN [44], and AGCN-avg. We implement these single-view methods on each view and take the best result as the final result. Furthermore, we calculate average results of all views and remark them as SDCN-avg and AGCN-avg.
- **Anchor-based methods:** SMVSC [18] and MvAGC [10].
- **Deep learning-based method:** DIAGC [30] and GMVC [45].

Experiments are performed on the computer equipped with TITIAN RTX 2080TI GPU of memory 11G and Intel Core i9 9820X CPU of 128G. Based on our memory capacity, for the Ogbn-products dataset, we divide the matrix $2,449,029 \times 2,449,029$ into several $100,000 \times 100,000$ chunks for matrix multiplication. The source code is available at https://github.com/lmyfree/AVAC.

4.2. Clustering results

In this experiment, we evaluate the effectiveness of AVAC and the comparison algorithms based on the clustering results. Tables 3 and 4 list clustering results compared with recent algorithms. Overall, our proposed method AVAC outperforms other advanced methods [10,11,18,30,42–45]. To be precise, we have the following observations.

According to Tables 1 and 3, single-view methods are not suitable for multi-view datasets and fail to deal with multi-view datasets because they cannot take advantage of complementary information between views.

PwMC and MAGC are subspace methods. In comparison, AVAC consistently achieves superior performance. Specifically, on the Flickr dataset, AVAC outperforms MAGC by approximately 40% in terms of accuracy. Furthermore, in the AMAP dataset, AVAC achieves an accuracy that is roughly 50% greater than that of PwMC.

Anchor-based methods for Euclidean data fail to deal with multi-view attributed graphs. We can observe that SMVSC performs worse in multi-view attributed graphs, and is more than 10% lower than AVAC in terms of accuracy in most datasets. It can only focus on attributed matrices without considering multiple graph structures, which leads to lower accuracy. Thus, these anchor methods for Euclidean data are not effective for multi-view attributed graphs. As for MvAGC, the sampled anchor-based method, our method AVAC presents better performances than MvAGC, with about 10% higher in NMI or ARI in most datasets like BlogCatalog, Pubmed, Flickr, and AMAP datasets. This illustrates that our virtual anchors can accurately capture the distribution of the real data compared with the sampled anchors.

Compared with deep-learning methods, AVAC performs better than DIAGC with about 10% higher on four evaluated metrics in the AMAP dataset. AVAC outperforms GMVC across all datasets, achieving over 40% higher accuracy on the BlogCatalog, Flickr, and AMAP datasets. AVAC generates virtual anchors from adaptive learning which makes anchors more representative and contributes to better performances. Additionally, experimental results show that the GMVC model suffers from class imbalance during clustering, leading to lower F1 scores.

According to the results of the BlogCatalog dataset, AVAC is 33% higher than the best advanced methods in accuracy. Also, AVAC is about 6% higher than other methods in the Wiki dataset according to NMI. From the results of the Flickr dataset, AVAC is about 15% higher than other advanced methods on four evaluated metrics. In the Ogbn-products dataset, most methods are unable to process large-scale graphs due to excessive running time or high memory requirements. Compared with MvAGC, AVAC is about 80% higher in the Ogbn-products dataset according to ARI, which indicates AVAC is more effective than other methods. Fig. 5 visualizes the clustering results of our method. Fig. 6 visualizes the clustering results of the latest comparison method, DIAGC. We find that the clustering results of AVAC are clearer and more distinguishable than those of DIAGC, especially in the BlogCatalog and Flickr datasets.

4.3. Time comparison

In this experiment, we assess the time of AVAC and other comparison algorithms. For a fair comparison, we compare the running times of various methods with the best results. From Table 5, we can obviously

Table 3
Clustering accuracy. The symbol "-" indicates that no clustering result was obtained within 24 h, typically due to memory or time constraints.(Single-view methods Anchor-based methods Subspace methods Deep learning-based methods).

Dataset	Blogcatalog				ACM				DBLP			
	ACC	NMI	F1	ARI	ACC	NMI	F1	ARI	ACC	NMI	F1	ARI
SDCN	0.2283	0.0957	0.1337	0.0106	0.9064	0.6939	0.9069	0.7444	0.5891	0.3442	0.5133	0.2867
SDCN-avg	0.2030	0.0488	0.0925	0.0053	0.7829	0.6097	0.7411	0.6022	0.5224	0.2533	0.4344	0.2099
AGCN	0.2575	0.1341	0.1515	0.0307	0.8922	0.6682	0.8913	0.7127	0.7649	0.6137	0.6387	0.6009
AGCN-avg	0.2176	0.0680	0.1014	0.0153	0.7935	0.5610	0.7940	0.5693	0.6290	0.4153	0.5536	0.3816
PwMC	0.1776	0.0004	0.2861	0.0000	0.5190	0.2857	0.5368	0.3566	0.4020	0.1191	0.4253	0.0599
MAGC	0.3369	0.1743	0.2667	0.0791	0.8723	0.5976	0.8705	0.5976	0.9307	0.7787	0.9262	0.8326
SMVAC	0.4076	0.2480	0.2944	0.1095	0.7543	0.4015	0.6187	0.4259	0.5785	0.3080	0.4406	0.1700
MvAGC	0.5454	0.3616	0.5558	0.2827	0.8882	0.6559	0.8893	0.6995	0.9171	0.7470	0.9116	0.8038
GMVC	0.4727	0.3014	0.1054	0.2452	0.9289	0.7479	0.0393	0.8006	0.7550	0.4788	0.2354	0.4943
DIAGC	0.4500	0.2612	0.4159	0.1818	0.9167	0.7127	0.9177	0.7671	0.9007	0.7353	0.8945	0.7700
AVAC (Our)	**0.8749**	**0.7123**	**0.8722**	**0.7278**	**0.9302**	**0.7534**	**0.9306**	**0.8029**	**0.9346**	**0.7886**	**0.9305**	**0.8419**

Dataset	Pubmed				Wiki				Flickr			
	ACC	NMI	F1	ARI	ACC	NMI	F1	ARI	ACC	NMI	F1	ARI
SDCN	0.5470	0.1612	0.5490	0.1502	0.3551	0.3713	0.3137	0.1807	0.1302	0.0197	0.0475	0.0006
SDCN-avg	0.5223	0.1211	0.4615	0.0985	0.2940	0.2452	0.1928	0.1198	0.1240	0.0109	0.0361	0.0003
AGCN	0.5742	0.1431	0.5794	0.1439	0.1672	0.0142	0.0179	0.0005	0.1872	0.0960	0.0922	0.0227
AGCN-avg	0.5256	0.1122	0.5206	0.0989	0.1672	0.0142	0.0179	0.0005	0.1661	0.0572	0.0733	0.0139
PwMC	0.3994	0	0.5260	0	0.1073	0.3344	0.0519	0.0394	0.1172	0.0011	0.2001	0
MAGC	0.6282	0.2324	0.6312	0.2094	**0.5625**	0.4654	0.3851	0.3087	0.2459	0.1480	0.0599	0.1573
SMVAC	0.5389	0.1531	0.4694	0.1343	0.4686	0.4562	0.3148	0.2317	0.4817	0.3632	0.3437	0.2522
MvAGC	0.3994	0.0002	0.1904	0	0.5018	0.4815	0.4220	0.2781	0.3287	0.2707	0.2726	0.1008
GMVC	0.5507	0.2167	0.2006	0.1646	0.4424	0.4633	0.0069	0.2615	0.2055	0.1586	0.0154	0.0904
DIAGC	-	-	-	-	0.5222	0.5027	0.4468	0.3478	0.2429	0.1066	0.2193	0.0813
AVAC (Our)	**0.6495**	**0.2754**	**0.6549**	**0.2551**	0.5417	**0.5611**	**0.4730**	**0.3743**	**0.6417**	**0.5134**	**0.6271**	**0.4692**

Table 4
Clustering accuracy. The symbol "-" indicates that no clustering result was obtained within 24 h, typically due to memory or time constraints.(Single-view methods Anchor-based methods Subspace methods Deep learning-based methods).

Dataset	COM				AMAP				Ogbn-products			
	ACC	NMI	F1	ARI	ACC	NMI	F1	ARI	ACC	NMI	F1	ARI
SDCN	0.3750	0.0011	0.0551	-0.0001	0.2554	0.0037	0.0579	-0.0014	-	-	-	-
SDCN-avg	0.3750	0.0011	0.0551	-0.0001	0.2546	0.0027	0.0546	-0.0007	-	-	-	-
AGCN	0.3911	0.0398	0.0878	0.0610	0.5026	0.4496	0.3895	0.3637	-	-	-	-
AGCN-avg	0.3836	0.0230	0.0763	0.0341	0.3782	0.2257	0.2204	0.1818	-	-	-	-
PwMC	0.3541	0.0522	0.3361	-0.0028	0.2292	0.1160	0.2691	-0.0033	-	-	-	-
MAGC	0.3655	0.0221	0.0853	-0.0019	0.6671	0.6582	0.6008	0.4957	-	-	-	-
SMVAC	0.3520	0.2119	0.3085	0.1062	0.4307	0.2907	0.2891	0.1116	-	-	-	-
MvAGC	0.3747	0.0008	0.0546	-0.0003	0.2539	0.0017	0.0513	0	0.1855	0.0680	0.0379	0.0119
GMVC	0.3424	0.0173	0.0589	0.0566	0.2444	0.0256	0.0426	0.0007	-	-	-	-
DIAGC	-	-	-	-	0.6507	0.5419	0.6004	0.4399	-	-	-	-
AVAC (Our)	**0.4760**	**0.4755**	**0.3997**	**0.3094**	**0.7819**	**0.7029**	**0.7193**	**0.6015**	**0.2618**	**0.1178**	**0.0410**	**0.8582**

observe that AVAC consumes much less time than other advanced methods. Compared with MvAGC, our method AVAC achieves a speedup of 2–5 times while maintaining high accuracy in most datasets. Notably, in the BlogCatalog dataset, AVAC demonstrates a remarkable speedup of 24 times compared to MvAGC. In contrast, as a deep learning approach, DIAGC incurs higher computational costs and requires more resources than alternative methods due to its extensive number of parameters that need to be trained. In the large-scale dataset (Ogbn-products dataset), AVAC is approximately 1.8 times faster than MvAGC. In contrast, SMVAC takes more than 24 h, while PwMC, MAGC, and DIAGC require too much memory.

4.4. Scalability experiments

We conducted scalability experiments at different scales–thousands, tens of thousands, hundreds of thousands, and millions to assess the capability of various methods in handling large-scale graphs, as shown in Table 6. Among these datasets, BlogCatalog, Ogbn-arxiv, and Ogbh-products are publicly available datasets, while BlogCatalog-pro is a synthetic dataset obtained by replicating the nodes of BlogCatalog eight times, resulting in a dataset with 40,000 nodes.

Discussion about the large-scale dataset. To analyze the capacity of different approaches in handling large-scale graphs, we introduce

Table 5
Running time. We mark italic to show the best performances. "OOM" represents out of memory. "Speedup" indicates the multiplier of acceleration compared to the fastest method.

Methods	ACM	DBLP	BlogCatalog	Wiki	Flickr	Com	AMAP	Pubmed	Ogbn-products
SMVAC	15.32s	26.38s	101.50s	26.65s	215.30s	63.37s	42.26s	90.65s	>24h
PwMC	252.59s	305.89s	45.01s	42.33s	136.39s	687.09s	144.04s	1936.72s	OOM
MAGC	83.87s	159.82s	173.40s	103.48s	367.80s	440.82s	54.31s	316.90s	OOM
MvAGC	5.68s	9.70s	31.60s	16.23s	68.81s	115.10s	19.57s	86.39s	14572.29s
DIAGC	267.02s	573.96s	600.85s	371.48s	1131.79s	OOM	1115.16s	OOM	OOM
GMVC	57.97s	89.74s	221.51s	87.51s	635.79s	170.66s	81.13s	686.70s	OOM
AVAC	*1.68s*	*2.00s*	*1.29s*	*4.45s*	*39.49s*	*14.43s*	*7.27s*	*21.90s*	*8063.28s*
Speedup	**3.38x**	**4.85x**	**24.49x**	**3.64x**	**1.74x**	**4.39x**	**2.69x**	**3.94x**	**1.80x**

Table 6
Scalability analysis. "OOM" represents out of memory.(Anchor-based methods Subspace clustering methods Deep learning-based methods).

Dataset / Method	BlogCatalog (Thousand-level)	BlogCatalog-pro (Ten-thousand-level)	Ogbn-arxiv (Hundred-thousand-level)	Ogbh-products (Million-level)
PwMC	45.01s	>6h	OOM	OOM
MAGC	173.40s	OOM	OOM	OOM
SMVAC	101.50s	1244.90s	4879.68s	>24h
MvAGC	31.60s	110.60s	227.63s	14572.29s
GMVC	221.51s	2574.83s	2397.55s	OOM
DIAGC	600.85s	OOM	OOM	OOM
AVAC (Our)	**1.29s**	**68.40s**	**100.39s**	**8063.28s**

the publicly available large graph dataset (Ogbn-products dataset). This dataset contains 2,449,029 nodes and 61,859,140 edges in each view, with each node having 100 dimensions. The memory required to store the two attributed matrices is 934MB, which is equivalent to 1.03 billion nodes with only one dimension. Additionally, the memory of the adjacency matrix in COO sparse format is 1.84GB, roughly the same as the LiveJournal dataset [46] with 112 million edges. Thus, considering the number of attributes and views, we find that the scale of the multi-view attributed graph is significantly larger than merely taking into account the number of nodes.

Scalability Analysis. From scalability experimental results, deep learning methods and several subspace-based methods are not suitable for large-scale graphs due to high complexity. Deep clustering methods (DIAGC) and subspace clustering methods (PwMC, MAGC) have time and space complexities of $\mathcal{O}(N^2)$ or higher. This means that as the data scale increases, the computational cost rises exponentially, severely limiting the scalability of these algorithms for large-scale graph data. As the number of nodes escalated from thousands to tens of thousands, the runtime of PwMC increased dramatically from 45 s to over 6 h, representing a more than 500-fold increase and underscoring its inadequate scalability. Similarly, DIAGC encounters memory exhaustion and is unable to complete computations for 40,000 nodes. In the Ogbn-products dataset, SMVAC (Anchor methods for Euclidean data) consumes over 24 h; while PwMC, DIAGC, and MAGC require more memory than other methods when dealing with the same dataset, so they cannot handle large scenarios at a low cost. Our method can yield better results and speed up by 1.8 times compared with MvAGC. Thus, our method is more suitable for large-scale scenarios compared with other methods.

4.5. Stability analysis

To evaluate the stability of the models, we repeat the experiments on the ACM dataset 50 times with fixed parameters and compare the results with the existing anchor method, MvAGC [10], as shown in Fig. 7. We employ the same settings to ensure the stability of k-means in both our method and MvAGC. This allows a fair comparison of stability between the sampled anchor method and our adaptive virtual anchor method. From Fig. 7, we can observe that the results of AVAC are stable for our anchors are obtained via optimization, while MvAGC, where anchors are selected by a random algorithm, is affected by the random numbers. The results of MvAGC fluctuate ±5%, resulting in poor stability of the model.

4.6. Parameter analysis

AVAC has several parameters to tune, including the number of anchors m, the weighting factor λ, and the filter order h. Fig. 8 shows the results under different parameters. In AVAC, we fix one parameter and vary the other two parameters. For the DBLP dataset, according to the metrics NMI and ARI, we can observe that the weighting factor λ affects the performance to some degree. In the DBLP dataset, when we set it to 100, AVAC can obtain excellent results considering all evaluation metrics. Besides, it is obvious that we should not set the value of anchor number m too large. According to the results, we infer that an excessive number of optimized anchors may lead to the case that multiple representative anchors belong to the same class. This may lead to large distances within a class and overly broad class boundaries, making it challenging to accurately partition the clusters. Furthermore, we find that the filter order h can combine attribute information with structural information efficiently.

Moreover, we visualize the view weights of most datasets in Fig. 9 and find that the weights of each view in ACM, DBLP, and Blogcatalog datasets are similar, indicating that each view in these datasets is equally important and contains a large amount of information. In AMAP, Computer, and Flickr datasets, the weight of the first view is significantly greater than that of the second view, indicating that the first view may contain more useful information than the second view.

4.7. Ablation analysis

To better evaluate the importance of each part in AVAC, we have redesigned three ablation experiments as follows:

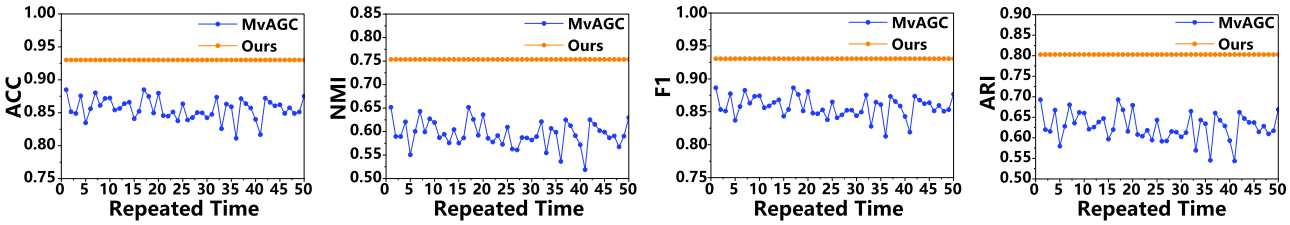
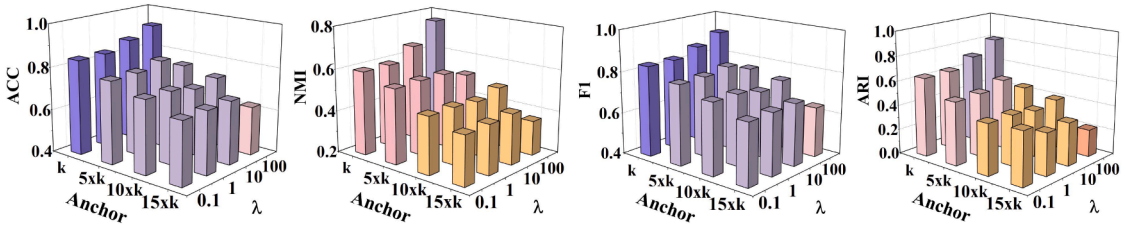
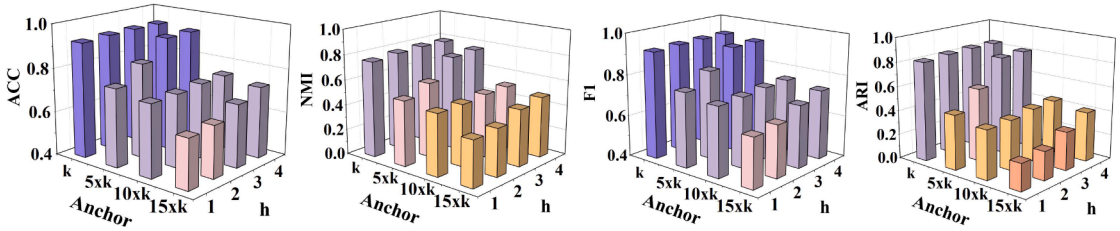


Fig. 7. Stability analysis on the ACM dataset.



(a) Parameter analysis of Weighting factor and anchor number on the DBLP dataset



(b) Parameter analysis of *h*-order filter and anchor number on the DBLP dataset

Fig. 8. Parameter analysis for AVAC.

Table 7
Ablation analysis.

Dataset	ACM				DBLP				Wiki			
	ACC	NMI	F1	ARI	ACC	NMI	F1	ARI	ACC	NMI	F1	ARI
w/o TR	0.8614	0.5685	0.8610	0.6340	0.4333	0.2409	0.4575	0.0758	0.5234	0.5489	0.4737	0.3482
w/o Graph Filter	0.9137	0.7059	0.9150	0.7589	0.9169	0.7427	0.9126	0.7995	0.5480	0.5582	0.4732	0.3668
w/o FR	0.5990	0.3220	0.5722	0.3173	0.9115	0.7343	0.9052	0.7913	0.4099	0.3769	0.3543	0.2225
AVAC(ours)	0.9302	0.7534	0.9306	0.8029	0.9346	0.7886	0.9305	0.8419	0.5417	0.5611	0.4730	0.3743

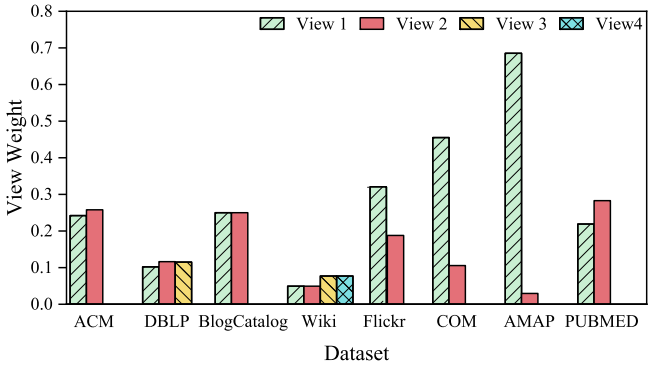


Fig. 9. View weights.

- **W/o (With/Without) Topology Reconstruction (TR).** According to Table 7, we can observe that topology reconstruction plays an important role in clustering. Without topology reconstruction, we may ignore the structure information to some degree and fail to capture the real structure of nodes.
- **W/o Graph Filter.** From Table 7, we can find that the graph filter has a tiny impact on clustering results. The graph filter attempts to combine structure information with attributes and obtain smooth representations. However, without the graph filter, the topology reconstruction can contribute to capturing the real structure of nodes, so that the graph filter influences the results little.
- **W/o Feature Reconstruction (FR).** The results in Table 7 indicate that feature reconstruction has a greater impact on the ACM dataset, which may be due to the higher attribute dimensions of the ACM dataset. When facing high-dimensional feature datasets, removing the feature reconstruction will ignore important information about attributes, thereby affecting clustering results.

4.8. Summary

Through extensive experiments, we can observe that compared to state-of-the-art approaches, AVAC can significantly improve accuracy while further reducing the time cost of clustering. Particularly for the Blogcatalog dataset, AVAC achieves a speedup of 24 times compared to other methods. Furthermore, we find that AVAC can respond more quickly to large-scale graphs by enlarging the graph scale to millions of nodes. In particular, on the public large-scale dataset, the AVAC algorithm has achieved a speedup of 1.8 times over the state-of-the-art

method MvAGC while improving accuracy significantly. Moreover, compared with other sampled anchor methods like MvAGC [10], AVAC can obtain more stable results.

5. Conclusion

In this paper, we propose the adaptive virtual anchor clustering method (AVAC), which efficiently clusters multi-view attributed graphs. AVAC generates adaptive virtual anchors and connects anchor generation and anchor graph construction closely. By executing these two processes cyclically and alternately, through which these two processes affect each other, AVAC can generate effective anchors that capture more accurate real data distribution and graph structure. Extensive experimental results validate the effectiveness of our method in terms of accuracy, running time, and stability. In the future, we will focus on incomplete clustering and further explore virtual anchor clustering methods that are more suitable for incomplete multi-view attributed graphs.

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.

CRediT authorship contribution statement

Mengyao Li: Writing - review & editing; **Zhibang Yang:** Project administration; **Xu Zhou:** Investigation; **Joey Tianyi Zhou:** Investigation; **Quanqing Xu:** Investigation; **Chuanhui Yang:** Investigation; **Kenli Li:** Supervision; **Keqin Li:** Supervision.

Acknowledgments

The research was supported by the National Natural Science Foundation of China (Grant Nos. 62576051, U23A20317, 62225205, 62572182, 62402481), the Creative Research Groups Program of the National Natural Science Foundation of China (Grant No. 62321003), the Natural Science Foundation of Hunan Province (Grant No. 2023JJ10016), Ant Group through CCF-Ant Research Fund (Grant No. CCF-AFSG RF20250518), and International Talented Young Scientist Program (P2SU43004).

References

[1] Y. Wang, S. Ye, X. Xu, Y. Geng, Z. Zhao, X. Ke, T. Wu, Scalable community search with accuracy guarantee on attributed graphs, in: 40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13–16, 2024, IEEE, Utrecht, The Netherlands, 2024, pp. 2737–2750.
[2] C. He, J. Cheng, G. Chen, Q. Guan, X. Fei, Y. Tang, Detecting communities with multiple topics in attributed networks via self-supervised adaptive graph convolutional network, Inf. Fusion 105 (2024) 102254.
[3] J. Wang, X. Qu, J. Bai, Z. Li, J. Zhang, J. Gao, SAGES: scalable attributed graph embedding with sampling for unsupervised learning, IEEE Trans. Knowl. Data Eng. 35 (5) (2023) 5216–5229.
[4] H. Wang, D. Lian, H. Tong, Q. Liu, Z. Huang, E. Chen, Decoupled representation learning for attributed networks, IEEE Trans. Knowl. Data Eng. 35 (3) (2023) 2430–2444.
[5] N. Zhang, Y. Ye, X. Lian, M. Chen, Top-L most influential community detection over social networks, in: 40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13–16, 2024, IEEE, Utrecht, The Netherlands, 2024, pp. 5767–5779.
[6] C. Yang, J. Han, Revisiting citation prediction with cluster-aware text-enhanced heterogeneous graph neural networks, in: 39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3–7, 2023, IEEE, Anaheim, CA, USA, 2023, pp. 682–695.
[7] A.G. Kermani, A. Kamandi, A. Moeini, Integrating graph structure information and node attributes to predict protein-protein interactions, J. Comput. Sci. 64 (2022) 101837.
[8] S. Jin, Z. Chen, S. Yu, M. Altaf, Z. Ma, Self-augmentation graph contrastive learning for multi-view attribute graph clustering, in: Proceedings of the 2023 Workshop on Advanced Multimedia Computing for Smart Manufacturing and Engineering, AMCSME 2023, Ottawa on, Canada, 29 October 2023, ACM, Ottawa ON, Canada, 2023, pp. 51–56.
[9] X. Ma, S. Xue, J. Wu, J. Yang, C. Paris, S. Nepal, Q.Z. Sheng, Deep multi-attributed-view graph representation learning, IEEE Trans. Netw. Sci. Eng. 9 (5) (2022) 3762–3774.
[10] Z. Lin, Z. Kang, Graph filter-based multi-view attributed graph clustering, in: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19–27 August 2021, ijcai.org, Montreal, Canada, 2021, pp. 2723–2729.
[11] Z. Lin, Z. Kang, L. Zhang, L. Tian, Multi-view attributed graph clustering, IEEE Trans. Knowl. Data Eng. 35 (2) (2023) 1872–1880.
[12] W. Yang, J. Xu, R. Zhou, L. Chen, J. Li, P. Zhao, C. Liu, Multi-view attentive variational learning for group recommendation, in: 40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13–16, 2024, IEEE, Utrecht, The Netherlands, 2024, pp. 5022–5034.
[13] E. Pan, Z. Kang, Multi-view contrastive graph clustering, in: Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6–14, 2021, Virtual, 2021, pp. 2148–2159.
[14] J. Lin, M. Chen, X. Zhu, C. Wang, H. Zhang, Dual information enhanced multi-view attributed graph clustering, arXiv:2211.14987 (2022).
[15] W. Xia, Q. Wang, Q. Gao, X. Zhang, X. Gao, Self-supervised graph convolutional network for multi-view clustering, IEEE Trans. Multim. 24 (2022) 3182–3192.
[16] L. Liu, Z. Kang, J. Ruan, X. He, Multilayer graph contrastive clustering network, Inf. Sci. 613 (2022) 256–267.
[17] S. Fan, X. Wang, C. Shi, E. Lu, K. Lin, B. Wang, One2Multi graph autoencoder for multi-view graph clustering, in: WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20–24, 2020, ACM / IW3C2, Taipei, 2020, pp. 3070–3076.
[18] M. Sun, P. Zhang, S. Wang, S. Zhou, W. Tu, X. Liu, E. Zhu, C. Wang, Scalable multi-view subspace clustering with unified anchors, in: MM '21: ACM Multimedia Conference, Virtual Event, China, October 20, - 24, 2021, ACM, Virtual Event,China, 2021, pp. 3528–3536.
[19] J. Li, Q. Wang, M. Yang, Q. Gao, X. Gao, Efficient anchor graph factorization for multi-view clustering, IEEE Trans. Multim. 26 (2024) 5834–5845.
[20] Y. Mi, H. Chen, Z. Yuan, C. Luo, S.-J. Horng, T. Li, Fast multi-view subspace clustering with balance anchors guidance, Pattern Recognit. 145 (2024) 109895.
[21] B. Yang, X. Zhang, Z. Li, F. Nie, F. Wang, Efficient multi-view K-means clustering with multiple anchor graphs, IEEE Trans. Knowl. Data Eng. 35 (7) (2023) 6887–6900.
[22] P. Neal, C. Eric, P. Borja, E. Jonathan, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends® Mach. Learn. 3 (1) (2011) 1–122.
[23] J. Li, X. Hu, J. Tang, H. Liu, Unsupervised streaming feature selection in social media, in: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19, - 23, 2015, ACM, Melbourne, VIC, Australia, 2015, pp. 1041–1050.
[24] M. Chen, P. Lai, D. Liao, C. Wang, J. Lai, Graph prompt clustering, IEEE Trans. Pattern Anal. Mach. Intell. 47 (2025) 5794–5805.
[25] X. Hu, J. Chen, G. Chen, Y. Tang, W. Pedrycz, Y. Jiang, Multi-view fuzzy clustering for multi-layer and multi-attribute graphs, IEEE Trans. Fuzzy Syst. 33 (2025) 1–16.
[26] X. Lin, R. Yang, H. Zheng, X. Ke, Spectral subspace clustering for attributed graphs, in: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.1, ACM, Toronto, Canada, 2025, pp. 789–799.
[27] N. Mrabah, M. Bouguessa, M.F. Touati, R. Ksantini, Rethinking graph auto-encoder models for attributed graph clustering (Extended abstract), in: 39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3–7, 2023, IEEE, Anaheim, CA, USA, 2023, pp. 3891–3892.
[28] P. Zhu, Q. Wang, Y. Wang, J. Li, Q. Hu, Every node is different: dynamically fusing self-supervised tasks for attributed graph clustering, in: Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, AAAI Press, Vancouver, Canada, 2024, pp. 17184–17192.
[29] H. Zhao, B. Yang, Y. Cen, J. Ren, C. Zhang, Y. Dong, E. Kharlamov, S. Zhao, J. Tang, Pre-training and prompting for few-shot node classification on text-attributed graphs, in: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25–29, 2024, ACM, Barcelona, Spain, 2024, pp. 4467–4478.
[30] J.-Q. Lin, M.-S. Chen, X.-R. Zhu, C.-D. Wang, H. Zhang, Dual information enhanced multiview attributed graph clustering, IEEE Trans. Neural Netw. Learn. Syst. 36 (2024) 1–12.
[31] L. Zhou, Y. Guo, Z. Zhang, Multi-view attributed graph clustering based on graph diffusion convolution with adaptive fusion, Expert Syst. Appl. 260 (2025) 125286.
[32] P. Zhang, Y. Pan, S. Wang, S. Yu, H. Xu, E. Zhu, X. Liu, I.W. Tsang, Max-mahalanobis anchors guidance for multi-view clustering, in: AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, AAAI Press, Philadelphia, PA, USA, 2025, pp. 22488–22496.
[33] Y. Qin, N. Pu, H. Wu, N. Sebe, Discriminative anchor learning for efficient multi-view clustering, IEEE Trans. Multim. 27 (2025) 1386–1396.
[34] Z. Lu, F. Nie, L. Ma, R. Wang, X. Li, Simplifying scalable subspace clustering and its multi-view extension by anchor-to-sample kernel, IEEE Trans. Image Process. 34 (2025) 5084–5098.
[35] X. Yu, H. Liu, Y. Lin, Y. Wu, C. Zhang, Auto-weighted sample-level fusion with anchors for incomplete multi-view clustering, Pattern Recognit. 130 (2022) 108772.
[36] H. Jiang, H. Tao, Z. Jiang, C. Hou, Unaligned multi-view clustering via diversified anchor graph fusion, Pattern Recognit. 170 (2026) 111977.
[37] J.C. Bezdek, R.J. Hathaway, Convergence of alternating optimization, Neural Parallel Sci. Comput. 11 (4) (2003) 351–368.

[38] X. Zhang, H. Liu, Q. Li, X.-M. Wu, X. Zhang, Adaptive graph convolution methods for attributed graph clustering, IEEE Trans. Knowl. Data Eng. 35 (12) (2023) 12384–12399.
[39] R. Yang, J. Shi, Y. Yang, K. Huang, S. Zhang, X. Xiao, Effective and scalable clustering on massive attributed graphs, in: WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19–23, 2021, ACM / IW3C2, Ljubljana, Slovenia„ 2021, pp. 3675–3687.
[40] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, J. Leskovec, Open graph benchmark: datasets for machine learning on graphs, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2020, p. 16.
[41] C. Fettal, L. Labiod, M. Nadif, Simultaneous linear multi-view attributed graph representation learning and clustering, in: Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM 2023, Singapore, 27 February 2023 - 3 March 2023, ACM, Singapore, 2023, pp. 303–311.
[42] F. Nie, J. Li, X. Li, Self-weighted multiview clustering with multiple graphs, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19–25, 2017, ijcai.org, Melbourne, 2017, pp. 2564–2570.
[43] D. Bo, X. Wang, C. Shi, M. Zhu, E. Lu, P. Cui, Structural deep clustering network, in: Proceedings of the Web Conference 2020, Association for Computing Machinery, New York, NY, USA, 2020, pp. 1400–1410.
[44] Z. Peng, H. Liu, Y. Jia, J. Hou, Attention-driven graph clustering network, in: Proceedings of the 29th ACM International Conference on Multimedia, Association for Computing Machinery, New York, NY, USA, 2021, pp. 935–943.
[45] H. He, J. Xu, G. Wen, Y. Ren, N. Zhao, X. Zhu, Graph embedded contrastive learning for multi-view clustering, in: Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25, International Joint Conferences on Artificial Intelligence Organization, Montreal, Canada, 2025, pp. 5336–5344.
[46] A. Mislove, M. Marcon, P.K. Gummadi, P. Druschel, B. Bhattacharjee, Measurement and analysis of online social networks, in: Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference, IMC 2007, San Diego, California, USA, October 24–26, 2007, ACM, San Diego, California, USA, 2007, pp. 29–42.