Multi-View Graph Learning by Joint Modeling of Consistency and Inconsistency

Youwei Liang\textsuperscript{\textcopyright}, Dong Huang\textsuperscript{\textcopyright}, Member, IEEE, Chang-Dong Wang\textsuperscript{\textcopyright}, Member, IEEE, and Philip S. Yu\textsuperscript{\textcopyright}, Life Fellow, IEEE

Abstract—Graph learning has emerged as a promising technique for multi-view clustering due to its ability to learn a unified and robust graph from multiple views. However, existing graph learning methods mostly focus on the multi-view consistency issue, yet often neglect the inconsistency between views, which makes them vulnerable to possibly low-quality or noisy datasets. To overcome this limitation, we propose a new multi-view graph learning framework, which for the first time simultaneously and explicitly models multi-view consistency and inconsistency in a unified objective function, through which the consistent and inconsistent parts of each single-view graph as well as the unified graph that fuses the consistent parts can be iteratively learned. Though optimizing the objective function is NP-hard, we design a highly efficient optimization algorithm that can obtain an approximate solution with linear time complexity in the number of edges in the unified graph. Furthermore, our multi-view graph learning approach can be applied to both similarity graphs and dissimilarity graphs, which lead to two graph fusion-based variants in our framework. Experiments on 12 multi-view datasets have demonstrated the robustness and efficiency of the proposed approach. The code is available at https://github.com/youwei-liang/Multi-view_Graph_Learning.

Index Terms—Consistency, data clustering, efficient optimization, inconsistency, multi-view clustering, multi-view graph learning.

I. INTRODUCTION

MULTI-VIEW data consist of features collected from multiple heterogeneous sources (or views). Multiple views of features can provide rich and complementary information for discovering the underlying cluster structure of data. It has been a popular research topic in recent years as to how to exploit the features effectively and jointly from multiple views and thus achieve robust clustering results for multi-view data [1]–[10].

In the literature, numerous (single-view) clustering methods have been developed [11], among which the graph-based methods are one of the most widely studied categories [12]–[14]. The graph-based methods typically construct a graph structure and then partition the graph to obtain the clustering result. In these methods, the construction of the graph is independent of clustering, and the clustering performance heavily relies on the predefined graph. To alleviate this limitation, some graph learning methods have been presented [15]–[18], where the graph structure can be adaptively learned in the clustering process. Recently, inspired by single-view graph learning [15], [16], [19], multi-view graph learning has rapidly emerged as a powerful technique for enhancing multi-view clustering performance [2]–[5], [20]. Notably, Zhan et al. [2]–[4] developed several multi-view graph learning approaches that are able to fuse multiple graphs into a unified graph with the desired number of connected components. Nie et al. [5] proposed a self-weighted scheme for fusing multiple graphs with the importance of each view considered. Despite this significant progress, a common limitation to these multi-view graph learning methods lies in that they mostly focus on the consistency of multiple views, but lack the ability to explicitly consider both multi-view consistency and inconsistency (which may be brought in by noise, corruption, or view-specific characteristics) in their frameworks, which may degrade their performances when faced with complex or possibly noisy data.

To deal with the potential noise or corruptions in a graph, Bojchevski et al. [21] proposed a new graph-based clustering method based on the latent decomposition of the similarity graph into two graphs, namely, the good graph and the corrupted graph. Though it is able to learn a good graph by eliminating the influence of potential noise, this graph learning method [21] is only applicable to a single graph (for a single view) and cannot be utilized in the multi-view graph learning task where multiple graphs from multiple views are involved. Thereby, how to jointly model the multi-view consistency (which can be viewed as the multi-view good graphs) and the multi-view inconsistency (which can be viewed as the multi-view corrupted graphs) in a unified graph learning model to improve multi-view clustering performance is still an open problem.
To tackle this problem, this article proposes a novel multi-view graph learning approach, which is further applied to multi-view clustering. In this article, we argue that the simultaneous modeling of multi-view consistency and inconsistency can significantly benefit the multi-view graph learning process. In particular, with the graph structures of multiple views given, their consistency and inconsistency are simultaneously leveraged to learn a unified graph. It is intuitive to assume that the graph of each view can be decomposed into two parts, i.e., the consistent part and the inconsistent part, and the goal is to learn and remove the inconsistent (or noisy) parts while preserving the consistent parts. Specifically, we formulate the multi-view consistency and multi-view inconsistency as well as the graph fusion term into a new objective function. By iteratively optimizing the objective function, the multi-view graph decomposition and the multi-view graph fusion are simultaneously achieved. With the fused graph obtained, some conventional graph-based methods such as spectral clustering (SC) can be performed to obtain the final multi-view clustering result. For clarity, we provide an illustration for our multi-view graph learning model in Fig. 1. As shown in the first row of Fig. 1, the four similarity (affinity) matrices from four views appear to be corrupted to different extents, and our similarity graph fusion (SGF) method can effectively remove most of these corruptions (or inconsistency) while yielding a unified and better graph with their consistent parts fused and strengthened. As shown in the second row in Fig. 1, by graph fusion with both consistency and inconsistency considered, the final clustering (in the fifth column) on the fused graph is significantly better than the clustering on the single-view graphs. It also shows that the proposed approach is able to achieve superior performance at the presence of some highly corrupted graph (e.g., the fourth single-view graph in Fig. 1).

For clarity, the main contributions of this work are summarized as follows.

1) We propose a novel multi-view graph learning approach, which for the first time, to the best of our knowledge, simultaneously and explicitly models multi-view consistency and multi-view inconsistency in a unified objective function, where multi-view consistency can be iteratively learned and fused into a unified graph as the multi-view inconsistency is automatically identified and removed.

2) Though optimizing the objective function is NP-hard, we design a highly efficient algorithm to obtain an approximate solution by exploiting the structures of the quadratic programs (QPs) based on eigenvalue analysis and constraint simplification. Our algorithm has roughly linear time complexity in the number of effective edges in an affinity graph or the number of nodes in a kNN graph.

3) A novel multi-view clustering framework based on multi-view graph learning is presented, which is further extended into two graph fusion variants, corresponding to distance (dissimilarity) graph fusion (DGF) and SGF. The proposed DGF and SGF have shown superior performance over the state-of-the-art multi-view clustering algorithms in extensive experiments.

4) Even without dataset-specific hyperparameter tuning, the proposed algorithms can still achieve highly competitive clustering results on various multi-view datasets when compared to the best results obtained by some other state-of-the-art multi-view clustering approaches.

A preliminary version of this article was reported in [22]. In this article, we have made significant revisions and added a substantial analysis of the proposed framework. First, we have gained deeper understandings of the optimization problem (e.g., Lemmas 1 and 2 and Sections IV-B–IV-D and IV-F) and have almost completely revised the optimization algorithm, making it more efficient and stable than the projection method in [22] (as demonstrated by extensive comparison experiments). Our theoretical analysis reveals that one of the optimization subproblems can be reformulated as the sum of many quadratic functions that share the same Hessian matrix, which allows us to modify the d.c. algorithm (DCA) [23] to optimize these quadratic functions all at once. Second, because of improved stability, the proposed framework with the new optimization method can learn a good graph even without dataset-specific parameter tuning, which will be reported in the Supplementary Material. Third, we theoretically show that the time complexity of the new optimization approach is basically linear in the number of edges in the affinity graph or the
number of nodes in the kNN graph while exactly solving the problem that is NP-hard. We also provide more comprehensive experimental results to consolidate the theoretical arguments. Fourth, we introduce multi-view dense representation (MVDR) to replace the approach in [22] that consumes much more memory. Also, a graph normalization method is added to the two graph fusion algorithms (SGF and DGF), which improves the performance of clustering. Fifth, we introduce view-specific weights in our framework so that users can ensure that the unified graph is closer to the important views specific to their applications. Last but not least, Section VI is substantially extended, where more benchmark datasets are used and more experimental comparison and analysis are provided, which further demonstrate the efficiency, effectiveness, and robustness of the proposed framework.

The rest of this article is organized as follows. In Section II, we review the related work in multi-view clustering, especially in multi-view graph learning. In Section III, we propose the novel multi-view graph learning framework. In Section IV, we theoretically analyze the optimization problem in our framework and present a highly efficient algorithm to solve it. In Section V, two specific graph fusion versions for multi-view SC are presented based on the proposed framework. Finally, we report the experimental results in Section VI and conclude this article in Section VII. More experimental results are reported in the Supplementary Material.

II. RELATED WORK

In recent years, multi-view clustering has been a popular topic and many multi-view clustering algorithms have been developed from different technical perspectives. Bickel and Scheffer [24] extended the semisupervised co-training approaches [25] to multi-view clustering. The basic idea of co-training is to iterate over all views and optimize an objective function in the next view using the result obtained from the last view. However, co-training-based multi-view clustering algorithms may not converge [24], and thus, it is difficult to decide when to stop.

Kumar et al. [26] proposed co-regularized SC based on maximizing clustering agreement among all views. They presented an alternative regularization scheme that regularizes each view-specific set of eigenvectors toward a common centroid and used the common centroid to obtain the clustering result. The basic idea of their algorithm is that all views should yield a consensus clustering result. The idea of maximizing clustering agreement of all views is exploited by many other multi-view clustering approaches [27]–[29]. For example, Zong et al. [28] introduced weighted multi-view SC, where the clustering agreement between two views can be measured by the largest canonical angle between the subspaces spanned by the eigenvectors of the normalized Laplacian matrices of the views. Therefore, minimizing the canonical angles leads to maximizing the clustering agreement of all views. Nie et al. [30] presented another weighting scheme for different views, where procrustes analysis is used to obtain a consensus cluster indicator matrix from the spectral embedding of multi-view kernels. Notably, the capacity of each view is considered according to its adaptability to a global indicator matrix and smaller weights are given to the views with lower capacity. In contrast, we introduce a novel weighting scheme from a different perspective. Specifically, the weighting strategy is directly applied to the multi-view graphs, instead of Laplacian matrix [28] or indicator matrix [30]. Also, the weight for each view is iteratively learned under a multiojective loss function by considering the consistency and inconsistency between views and a smaller weight is given to the view that is either less adaptable to the unified graph or inconsistent to other views. The explicit modeling of multi-view consistency and inconsistency improves our results when faced with complex multi-view datasets (see Table 5 in the Supplementary Material).

Another popular multi-view clustering method flourishes by using graph learning approaches. Huang et al. [31] proposed a method for aggregating affinity matrices for SC, which attempts to reduce the influence of unreliable and irrelevant features in data. Nie et al. [5] proposed a parameter-free self-weighted scheme to fuse multiple graphs with the importance of each view considered. Lin et al. [17] proposed to learn a consensus graph with attributed data. Li et al. [18] presented to construct an essential similarity graph in a spectral embedding space. To eliminate the potential noise in data, Xia et al. [32] proposed robust multi-view SC, which aims to learn an intrinsic transition matrix from multiple views by restricting the transition matrix to be low rank. Zhan et al. [2]–[4] proposed to learn an intrinsic similarity graph from multiple similarity graphs. Their approaches learn the consensus graph by tuning the fused graph structure until it contains exactly the desired number of connected components $n_c$, where the time-consuming spectral decomposition is required for each single iteration. In contrast, we do not perform the expensive decomposition during learning. Instead, we propose a lightweight optimization algorithm to fuse the multi-view graphs to a unified graph with the inconsistency considered and SC is performed on the unified graph, which gives higher efficacy and efficiency.

III. LEARNING A CONSISTENT GRAPH WITH INCONSISTENCY CONSIDERED

In this section, we propose a new multi-view graph learning method that is capable of joint modeling of multi-view consistency and multi-view inconsistency in a unified optimization framework. The idea is to decompose the adjacency matrix of each graph (for each view) into two parts: the consistent part and the inconsistent part. By the definition of inconsistency, we design a novel objective function that can automatically identify the inconsistent parts and fuse the consistent parts into a unified adjacency matrix. By iteratively optimizing the objective function, the inconsistent and consistent parts of each view as well as the unified adjacency matrix are iteratively learned.

Let $W(i) \in \mathbb{R}^{n \times n}$ denote the similarity matrix for the $i$th view, with $n$ being the number of instances (data points). We assume that the similarity matrix is scale invariant in the applications, i.e., a similarity matrix $W(i)$ is equivalent to $kW(i)$ for all $k > 0$. This is a reasonable assumption because
in practice, we typically only care about the relative similarity of two nodes instead of their absolute similarity. For instance, we typically care whether the similarity between nodes $x$ and $y$ is higher than that of nodes $y$ and $z$.

Specifically, this assumption holds well for SC since scaling does not affect the eigenvectors of a matrix. To elaborate, suppose that we have two similarity matrices $W^{(i)}$ and $W^{(j)}$ with $W^{(i)} = \lambda W^{(j)}$. Then, their symmetrically normalized Laplacian matrices $L^{(i)} = L^{(j)}$, which give exactly the same clustering results in normalized cut [13]. Under the scale-invariant assumption, we need to scale the similarity matrices before fusing them into a unified similarity matrix, i.e., multiplying $W^{(i)}$ with a learnable scaling coefficient $\alpha_i$. To make the scaling result unique, we restrict the sum of the coefficients to 1, i.e., $\mathbf{1} = 1$. All the scaled similarity matrices should be close to the unified similarity matrix $S$. Hence, we have the following constrained optimization problem:

$$\min_{\alpha, S} \sum_{i=1}^{v} \|\alpha_i W^{(i)} - S\|_F^2$$

$$\text{s.t. } \mathbf{1} = 1, \quad \alpha \geq 0, \quad S \geq 0 \quad (1)$$

where $v$ is the number of views.

To jointly model multi-view consistency and multi-view inconsistency, we decompose the similarity matrix $W^{(i)}$ for the $i$th view into two parts: the consistent part $A^{(i)}$ and the inconsistent part $E^{(i)}$

$$W^{(i)} = A^{(i)} + E^{(i)} \quad (2)$$

with $A^{(i)}, E^{(i)} \in \mathbb{R}^{n \times n}$. The core question is how to find matrices $A^{(i)}$ and $E^{(i)}$ for $i = 1, \ldots, v$. In the single-view setting, Bojchevski et al. [21] characterized $E$ to be sparse. Directly extending this idea to the multi-view decomposition will not make use of any multi-view information, which is critical to multi-view clustering. As shown in (3) and (4), we explicitly consider the interplay between different views by introducing the cross-view terms $E^{(i)} \circ E^{(j)}$ into our objective function. These terms ensure that the inconsistent parts $E^{(i)}$ are really inconsistent with each other.

Different from previous decomposition works [21], [32] that mainly focus on modeling the noise in data, in this article, the inconsistency in multi-view data is a much broader concept, which involves not only noise but also the difference in view-specific characteristics. Technically, we can define the multi-view inconsistency as the optimal objective in (1), which measures the least possible conflict between the common adjacency matrix $S$ and the adjacency matrix for different views. Thus, multi-view inconsistency could be any misalignment of the relationship between two nodes in different graphs. However, as there is no prior knowledge about the source of the misalignment, it could be simply due to the situation that the node pair has a different relationship in different graphs. For example, if the nodes represent websites, where one view is text data and another view is image data, then the images of two websites can be exactly the same, while the text on them could describe very different things. Thus, the two websites are close to each other in the image view and far away in the text view. This kind of inconsistency is clearly not due to noise but simply an intrinsic characteristic of the nodes. Also, when we try to model the overall relationship of the two websites, we cannot declare one view as noise simply because it is very different from the other. On the other hand, the noise in multi-view data is simply the “noise” in common sense, which is brought in by some kind of error in the data such as corruption of signal in a communication channel or typos in text. While noise is typically considered sparse on a similarity graph [21], [32], the inconsistency may not. Since the relationship of nodes on a graph may be intrinsically different across views, inconsistency can appear everywhere on the similarity graphs.

Thus, the sparsity of noise within a single similarity matrix is not suitable for identifying the inconsistency on multi-view similarity graphs. Instead, we assume that the inconsistency is sparse across views. For example, suppose that we have five views and the similarities between nodes $x$ and $y$ on each view are 3.16, 3.19, 3.22, 3.17, and 3.95 (assuming that we have properly scaled the similarity matrices). We tend to believe that a good similarity measure between $x$ and $y$ is 3.20 (i.e., the consistent part). The similarity on the fifth view has a deviation of 0.75 (i.e., the inconsistent part) from the consistent part. We say that the inconsistency is sparse across views because only the fifth view has a relatively large inconsistent part. To ensure that the inconsistency is sparse across views, it is natural to decrease the sum of the products of the inconsistent parts, i.e.,

$$\sum_{i,j=1}^{v} \sum_{i \neq j} \text{sum}((\alpha_i E^{(i)}) \circ (\alpha_j E^{(j)})) \quad (3)$$

where $\circ$ denotes the Hadamard product (elementwise multiplication) of two matrices and $\text{sum}(\cdot)$ is the operator of summing all elements in a matrix. We scale the inconsistent part of each view to make them have fair contributions to the sum. Furthermore, we generally do not want the inconsistent parts to be too large, which leads to preventing the following value from becoming too large during learning:

$$\sum_{i=1}^{v} \text{sum}((\alpha_i E^{(i)}) \circ (\alpha_j E^{(j)})) \quad (4)$$

To jointly model multi-view consistency and inconsistency in a unified optimization framework, we combine the three terms (1), (3), and (4) into a unified objective function

$$\min_{\alpha, A^{(1)}, \ldots, A^{(v)}, E^{(1)}, \ldots, E^{(v)}, S} \sum_{i=1}^{v} \|\alpha_i A^{(i)} - S\|_F^2$$

$$+ \beta \sum_{i=1}^{v} \text{sum}((\alpha_i E^{(i)}) \circ (\alpha_j E^{(j)}))$$

$$+ \gamma \sum_{i,j=1}^{v} \text{sum}((\alpha_i E^{(i)}) \circ (\alpha_j E^{(j)})) \quad (6)$$

$$\text{s.t. } \mathbf{1} = 1, \quad \alpha \geq 0, \quad S \geq 0$$

$$W^{(i)} = A^{(i)} + E^{(i)}$$

$$A^{(i)} \geq 0, \quad E^{(i)} \geq 0, \quad i = 1, \ldots, v \quad (8)$$
where $\beta$ and $\gamma$ are parameters, $\beta$ controls the magnitude of inconsistent parts, and $\gamma$ prevents the consistent parts being incorrectly moved to inconsistent parts (i.e., ensuring sparsity of multi-view inconsistency) as we explain next.

To see how this objective can remove multi-view inconsistency, let us initialize $A^{(i)}$ as $W^{(i)}$ and $E^{(i)}$ as $0$. During an iterative learning process, if the inconsistent parts are correctly moved to $E^{(i)}$ from $A^{(i)}$, $a_iA^{(i)}$ will get closer to the consistent component of all views (i.e., $S$), and thus, the first term in our objective will decrease. Although the second and third terms will increase in this case, their increases will be small because of the sparsity of cross-view inconsistency and the small value of $\beta$ and thus canceled out by the considerable amount of reduction of the first term in the overall objective, and the net result is the reduction of the overall objective. Therefore, the optimization process is actually moving the inconsistent parts from the original similarity matrix $W^{(i)}$ to the matrix $E^{(i)}$ by minimizing the overall objective, which is the core principle of how we simultaneously model multi-view consistency and multi-view inconsistency in a unified optimization framework.

We shall rewrite the objective function in order to better apply optimization techniques to solve it. Let $B$ be a $v$-by-$v$ matrix with its diagonal elements being $\beta$ and off-diagonal elements being $\gamma$. Then, our objective function can be written in a more compact form

$$
\begin{align*}
\min_{\alpha} & \sum_{i=1}^{v} \lambda_i \|a_iA^{(i)} - S\|_F^2 + \sum_{i,j=1}^{v} B_{ij}\lambda_i\lambda_j a_i a_j \\
& \times \sum ((W^{(i)} - A^{(i)}) \circ (W^{(j)} - A^{(j)})) \\
\text{s.t.} & \alpha^T 1 = 1, \quad \alpha \geq 0, \quad S \geq 0 \\
& W^{(i)} \geq A^{(i)} \geq 0, \quad i = 1, \ldots, v
\end{align*}
$$

(9)

where $\lambda_i$ is a parameter to incorporate the importance of the $i$th view, and a higher value indicates greater importance. In particular, the parameter $\lambda_i$ is introduced to provide the users with an opportunity to incorporate prior knowledge through it. Yet, in this article, we mainly focus on the unsupervised multi-view clustering task, and the parameter $\lambda_i = 1$ can be simply used for all views when no prior knowledge is involved.

When prior knowledge is involved, the requirement for $\lambda_i$ is $\lambda_i > 0$ because negative value does not make sense, and if one wants to set $\lambda_i = 0$, then the $i$th view can be simply removed from the objective. In Section IV, the optimization problem will be theoretically analyzed, and a highly efficient algorithm will be developed to approximately solve it.

IV. OPTIMIZATION

Though the proposed framework, if optimized properly, can identify multi-view inconsistency and fuse consistent parts into a unified graph, the optimization of objective (9) is difficult compared to the single-view graph learning setting. First, the objective function (9) is not jointly convex on all variables, which rules out the possibility of directly applying convex optimization techniques. Second, there are a lot of couplings between the variables, i.e., different variables are multiplied together, which causes the exponent of variables as large as 4. Third and unfortunately, the optimization of objective (9) turns out to be NP-hard, which will be explained in Section IV-G1.

In the preliminary version of this article [22], a projection-based method is applied to optimize the objective. Although the projection-based method is a good heuristic, we observe that the objective increases in some optimization iterations, and it may yield unsatisfactory clustering performance on some datasets, as we will show in the Supplementary Material. Based on better understandings of the problem, we develop the following approaches to tackle these problems. First, we simplify that the constraints by proving one constraint can be automatically satisfied in Lemma 1. Second, objective (9) can be rewritten as two forms of quadratic functions, corresponding to Subproblem (1) and Subproblem (3) so that we can optimize them alternately. Finally, Subproblem (3) consists of at least $n$ nonconvex QPs with box constraints, which are difficult to solve since they are NP-hard [33]. By proving that these QPs share the same Hessian that has the desired property (Lemma 2), we are able to modify the DCA [23] to solve them all at once, which is much more efficient than solving them sequentially.

To facilitate these approaches, the MVDR is further proposed, where the nonzero elements of adjacency matrices of all views are arranged into a dense matrix. A significant advantage of the proposed approaches is that they mainly involve matrix–vector and matrix–matrix multiplications, which contribute to their high efficiency for large-scale problems.

A. Constraint Simplification

We first show that the constraint $S \geq 0$ in Problem (9) can be removed, while the global minimizer(s) remains the same. Define the following sets:

$$
\begin{align*}
G_0 &= \{\alpha \geq 0 \mid \alpha^T 1 = 1\}, \\
G_i &= \{A^{(i)} \mid W^{(i)} \geq A^{(i)} \geq 0\}, \quad i = 1, \ldots, v \\
G &= G_0 \times G_1 \times \cdots \times G_v \times \mathbb{R}^{n \times n} \\
G_+ &= G_0 \times G_1 \times \cdots \times G_v \times \mathbb{R}_{\geq 0}^{n \times n} \\
G_- &= G_0 \times G_1 \times \cdots \times G_v \times (\mathbb{R}^{n \times n} \setminus \mathbb{R}_{\geq 0}^{n \times n})
\end{align*}
$$

(10) (11) (12) (13) (14)

where $\times$ denotes the Cartesian product. Clearly, $G_+, G_\in \subset G$ and $G_+ \cup G_- = G$. Then, the objective function (9) is denoted by $f: G \rightarrow \mathbb{R}$. The following lemma shows that with the constraint $S \geq 0$ removed, the minimizer of the resulting problem still satisfies $S \geq 0$.

**Lemma 1:** For every minimizer $x^*$ of the problem

$$
\min_{x} f(x), \quad \text{s.t.} \quad x \in G
$$

(15)

we have $x^* \in G_+$.

**Proof:** Suppose that $x^* \notin G_+$. Then, $x^* \notin G_-$. Suppose that $x^* = (\alpha, A^{(1)}, \ldots, A^{(v)}, S)$, where $S_{pq} < 0$ for some $p, q \in \{1, \ldots, n\}$. Let $\mathcal{V} = \{(p, q) \mid S_{pq} < 0\}$. Let $\tilde{x} = (\alpha, A^{(1)}, \ldots, A^{(v)}, \tilde{S}) \in G_+$, where $\tilde{S} \in \mathbb{R}_{\geq 0}^{n \times n}$ such that $\tilde{S}_{pq} = 0$ for all $(p, q) \in \mathcal{V}$ and $S_{pq} = S_{pq}$ for all $(p, q) \not\in \mathcal{V}$. Let $c = \sum_{k=1}^{v} \lambda_k \sum_{(p, q) \in \mathcal{V}} (a_k A^{(k)}_{pq} - S_{pq})^2 + \sum_{i,j=1}^{v} B_{ij}\lambda_i\lambda_j a_i a_j \sum ((W^{(i)} - A^{(i)}) \circ (W^{(j)} - A^{(j)}))$. Then,

$$
f(x^*) = f(\alpha, A^{(1)}, \ldots, A^{(v)}, S) = \sum_{k=1}^{v} \lambda_k \sum_{(p, q) \in \mathcal{V}} (a_k A^{(k)}_{pq} - S_{pq})^2 + c
$$
as shown in (17) and (18), which show that (9) is actually the region $G$.

C. Multi-View Dense Representation

Since there are typically a lot of zero elements in the multi-view (sparse) adjacency matrices $W^{(i)}$, many terms within the summation in (17) and (18) vanish. We can exploit this to make our algorithm more efficient. In our preliminary version [22], the optimization is performed on sparse matrices, but we propose a better approach by constructing a dense matrix with only the nonzero elements from the sparse matrices of all views.

Let $\mathcal{F}$ be the common index set of the nonzero elements in the adjacency matrices of all views. Let $w^{(i)}$ be a row vector by taking the nonzero elements from $W^{(i)}$, corresponding to the indices in $\mathcal{F}$. Then, we stack the row vectors $w^{(1)}, \ldots, w^{(n)}$ to form a $v$-by-$n_c$ matrix $W$ (note that the notation differs from $W^{(i)}$), where $n_c$ is the number of elements in the nonzero index set $\mathcal{F}$. Similarly, we form a $v$-by-$n_c$ matrix $A$ from $A^{(1)}, \ldots, A^{(n)}$, according to $\mathcal{F}$. Then, the inconsistent part for all views can be represented by the matrix $E = W - A$ and the fused graph is represented by a row vector $s$ of length $n_c$. MVDR is more efficient than sparse matrix representation since the dense matrix requires only 1/3 memory of the sparse matrix and it is faster to access elements in a dense matrix than in a sparse one. For these reasons, we use only MVDR in our optimization algorithm.

We also normalize the multi-view adjacency matrices before performing optimization because normalization aids the optimization process and can reduce the total number of iterations. A typical normalization method is to divide the (nonnegative) adjacency matrices by its sum. For the MVDR $W$, we normalize each row separately.

D. Subproblem (1)

With $A$ and $s$ fixed, we optimize $q(\alpha)$. Note that $q(\alpha)$ is a quadratic function of $\alpha$. Formally, Subproblem (1) is formulated as a standard quadratic program (StQP) [34]

$$\min_{\alpha} q(\alpha) = 1/2 \alpha^T H \alpha - \alpha^T c$$

subject to $\alpha^T 1 = 1$, $\alpha \geq 0$

where $H$ and $c$ are computed by (17) as we now explain. Let $h$ be a vector where $h_i = \sum_{j=1}^{n_c} A^{(i)}_{ij}$, and let $T$ be a $v$-by-$v$ diagonal matrix where $T_{ii} = \lambda_i h_i$. Let $Z$ and $P$ be $v$-by-$v$ matrices defined as $Z_{ij} = B_{ij} \lambda_i \lambda_j$ and $P = Z \circ (E^T E)^{-1}$ where $E = W - A \in \mathbb{R}^{v \times n_c}$ is the inconsistent parts for all views. According to (17), the Hessian of $q(\alpha)$ is $H = 2(T + P)$. By (17), the linear coefficient of $q(\alpha)$ can be defined as $c_i = 2 \lambda_i A_i s_i^T$, where $A_i$ is the $i$th row of $A$.

We exploit the away-step Frank–Wolfe (AFW) algorithm [35], [36] (Algorithm 1) to solve Subproblem (1), which is able to identify the active set (i.e., the set $\{i : \alpha_i^* = 0\}$ for the minimizer $\alpha^*$) in a natural way [36], [37]. The feasible set of the variable $\alpha$ in StQP is $\{\alpha | \alpha^T 1 = 1, \alpha \geq 0\}$, which is a simplex (i.e., a hyperplane with boundary). The initial point of iterations in AFW is chosen to be on the simplex. Both the Frank–Wolfe direction and the away direction are chosen to be parallel to the plane so that the update of $\alpha$ using the formula $\alpha \leftarrow \alpha + \eta d$ (as in line 16 in Algorithm 1) will guarantee that the next point will also lie on the hyperplane. Also, the maximum step size $\eta_{\text{max}}$ is chosen to ensure that the next point is within the boundary of the simplex. Therefore, the iterations of $\alpha$ always lie on the simplex (hence the feasible set). The
AFW algorithm has low computation at each iteration and thus adds little overheads to the whole graph learning algorithm. Maybe the biggest advantage of using the AFW algorithm is that it is an iterative algorithm, i.e., it accepts an initial point and iteratively updates the point by stepping toward a descent direction. As we will see next, the whole graph learning algorithm (Algorithm 3) is itself iterative. Thus, every time we start AFW in Algorithm 3, we can simply initialize $\alpha$ in AFW as the latest $\alpha$ instead of random initialization. The benefit of this initialization is that, if Algorithm 3 is close to convergence, the latest $\alpha$ is also close to the minimizer of $\alpha$.

We now elaborate the exact line search method at line 15 of Algorithm 1. The purpose of exact line search is to determine a descent direction. As we will see next, the whole graph learning algorithm (Algorithm 3) is itself iterative. Thus, every time we start AFW in Algorithm 3, we can simply initialize $\alpha$ in AFW as the latest $\alpha$ instead of random initialization. The benefit of this initialization is that, if Algorithm 3 is close to convergence, the latest $\alpha$ is also close to the minimizer of $\alpha$, which reduces the number of iterations within AFW.

We now elaborate the exact line search method at line 15 of Algorithm 1. The purpose of exact line search is to determine the best step size $\eta$ such that the objective $f(\eta)$ in (21) decreases the most. We substitute the new point $\alpha + \eta d$ into the objective $1/2 \alpha^T H \alpha - \alpha^T e$ and get a quadratic function of the step size $\eta$.

$$f(\eta) = 1/2 (\alpha + \eta d)^T H (\alpha + \eta d) - (\alpha + \eta d)^T e$$

$$= 1/2 \eta^2 d^T H d + \eta (\alpha^T H d - d^T e) + \text{constant.}$$

The minimum of this quadratic function in the interval $(0, \eta_{\text{max}}]$ can be obtained by a simple analysis of its axis of symmetry.

### E. Subproblem (2)

In this section, we update $s$ with $\alpha$ and $A$ fixed. Taking the derivative of objective function (9) with respect to $S$ gives

$$2 \left( \sum_{i=1}^{n} \lambda_i \right) S - 2 \sum_{i=1}^{n} \lambda_i a_i A^{(i)} = 0.$$
Algorithm 2 Parallelized DCA for QPs With Box Constraints (Modified Algorithm 2a in [23])

**Input:** $D$ (Hessian), $L$ (combined linear coefficient), $W$ (upper bound), $A$ (initial point), $N$ (number of iterations)

**Output:** $A$

1. Compute $\rho$, the largest eigenvalue of $D$.
2. $H \leftarrow \rho I - D$ $I \in \mathbb{R}^{n \times n}$ is an identity matrix.
3. for $i = 1, \ldots, N$ do
4. $Y \leftarrow HA$
5. $A \leftarrow (Y + L)/\rho$
6. $A \leftarrow \text{mid}(0, A, W)$
7. end for

Note that for all $i \in \{1, \ldots, v\}$, $a_i \geq 0$ and $\sum_{i=1}^{v} a_i = 1$ by constraints (20). Thus, there exists $j \in \{1, \ldots, v\}$ such that $a_j > 0$. Then, $Q_{jj} > 0$ and thus $D_{jj} > 0$. Suppose that $D$ is negative semidefinite. Then, for any vector $v \in \mathbb{R}^n$, $v^T Dv \leq 0$. If we let $v = e_j$, where $e_j \in \mathbb{R}^n$ is the $j$th unit vector (i.e., its $j$th element is 1 and all other elements are 0), then $e_j^T D e_j = D_{jj} > 0$. This leads to a contradiction, and thus, $D$ is not negative semidefinite. Hence, $D$ has at least one positive eigenvalue.

Another reason to use DCA is that $D$ can be indefinite, which can be seen by adjusting $\beta, \gamma,$ and $\lambda$. Consequently, the QPs in Subproblem (3) are nonconvex, and thus, solving them is NP-hard [33], while DCA is designed for such nonconvex QPs [23].

In Algorithm 2, $\rho$ in line 1 can be obtained via the implicit restarted Lanczos method [38], and $\text{mid}(\cdot, \cdot, \cdot)$ denotes the elementwise median operator, i.e., the $(i, j)$th element of $\text{mid}(0, A, W) = \min(\max(0, A_{ij}), W_{ij})$. We found that Algorithm 2 converges very fast, usually in three iterations, as pointed out in [23]. Thus, $N = 3$ is our default.

G. Consistent Graph Learning Algorithm

By alternatively solving the three subproblems, objective (9) is optimized and the inconsistent part of each view is removed; the unified adjacency matrix of all views is iteratively learned. The complete consistent graph learning algorithm is presented in Algorithm 3. We next analyze the convergence and complexity of the proposed algorithm.

1) Convergence Analysis: While Subproblem (3) is NP-hard and, thus, finding a global minimizer of objective (9) is also NP-hard, we adopt an alternating optimization approach to iteratively update the local variables $\alpha$, $S$, and $A$. In Subproblems (1), since both the Frank–Wolfe direction and the away direction are descent directions, the objective value always decreases via the AFW algorithm [35], [36]. In Subproblem (2), the objective function (9) is a quadratic form of $S$, and its Hessian is $\left(\sum_{i=1}^{n_e} I, I\right)$ and thus positive definite. Hence, the stationary point obtained by (23) is a global minimizer of Subproblem (2). In Subproblem (3), the iteration converges to a Karush–Kuhn–Tucker point and it is a local minimum in most cases [23]. Since objective (9) is bounded below by 0, the alternating solving of the three subproblems can converge to a local minimizer of the total optimization problem (9) in most cases.

Algorithm 3 Consistent Graph Learning

**Input:** Adjacency matrices $\{W^{(1)}, \ldots, W^{(v)}\}$, $\beta$, $\gamma$, $\lambda$, $M$ (max iteration)

**Output:** Adjacency matrix of the unified graph $S$

1. Construct multi-view dense representation $W$ according to the index set $F$
2. Normalization: $W_{ij} = W_{ij}/(\sum_{k=1}^{n} W_{ik})$
3. Initialization: $A = W$, $\alpha = 1/v$; set $s$ by Eq. (24)
repeat
4. Use AFW (Algorithm 1) to update $\alpha$
5. Update $s$ by Eq. (24)
6. Use DCA (Algorithm 2) to update $A$
7. until convergence or max iteration is reached
8. Construct $S$ with $s$ according to the index set $F$

2) Complexity Analysis: In Subproblem (1), computing the Hessian $H$ and linear coefficient $c$ of the StQP requires $O(v^2 n_e)$ time and $O(v^2 (v + n_e))$ space. Solving the StQP via the AFW algorithm requires $O(v^2)$ space and $O(v^2)$ time at each iteration and $O(m_1 v^2)$ time in total, where $m_1$ is the number of iterations in AFW. Then, solving Subproblem (1) requires $O(v^2 (m_1 + n_e))$ space and $O(v^2)$ space in AFW. Then, solving Subproblem (2) requires $O(n_e)$ time and $O(n_e)$ space. In Subproblem (3), computing $D$ and $L$ requires $O(v^2 n_e)$ time and $O(v^2)$ space. Solving the StQP in Algorithm 2, computing the largest eigenvalue $\rho$ of $D$ requires $O(v^2 \kappa(D))$ time and $O(v^2)$ space. Thus, the total time complexity of Algorithm 3 is $O(m_2 v^2 (m_1 + \kappa(D) + n_e))$, where $m_2$ is the number of iterations of the loop in Algorithm 3, and the total space complexity is $O(v^2 n_e)$.

We have found that the iterations $m_2$ are typically less than 20 in our experiments (see Fig. 1 in the Supplementary Material), and thus, $m_2$ can be considered as a constant. Since $n_e$ is typically very large in practice, it is reasonable to assume that $n_e \gg m_1$ and $n_e \gg \kappa(D)$. Then, the time complexity of Algorithm 3 is $O(v^2 n_e)$, which means that the running time is linear in the number of edges $n_e$ and quadratic in the number of views $v$. The linear complexity in $n_e$ is due to the fact that we solve the optimization problem separately for each of the edges, for which the parallelized DCA for QPs with box constraints is designed. For a kNN graph, the number of edges is $kn$, where $k$ is the number of nearest neighbors and $n$ is the number of nodes (data points). Note that the fused graph may not have a similar number of edges as the kNN graphs. In an extreme case when the kNN of a node for each view has no overlap and that is the case for all nodes, the fused graph has roughly $vkn$ edges, which is the upper bound of the number of edges in the fused graph. Technically, if the set of kNN for node $i$ in view $j$ is denoted by $N_{i,j}^{(v)}$ and $N_{i,j}^{(v)} \cap N_{i,k}^{(v)} = \emptyset$ for all $j \neq k \in \{1, \ldots, v\}$, the number of edges in the fused graph reaches the upper bound of $vkn$. In that case, $n_e \approx vkn$ and the complexity of Algorithm 3 becomes $O(kn v^2)$. However, if the kNN for different views has large overlap, then $n_e \approx kn$ and the complexity becomes $O(kn v^2)$. In the Supplementary
Material, we will report experimental results that show the empirical complexity aligns very well with the theoretical complexity \( O(kn\nu^2) \). Specifically, we will show that the running time is typically quadratic to the number of views as shown in Fig. 4 in the Supplementary Material, which suggests that the upper bound of edge number is hardly reached in practice.

V. TWO GRAPH FUSION VERSIONS

The proposed multi-view graph learning method is applicable to both similarity graphs and distance (dissimilarity) graphs. Thus, we extend our multi-view clustering framework into two graph fusion versions, SGF and DGF. The reason to fuse distance matrix is that distance may better reflect the relationship between data points than similarity as we soon explain in Section V-B.

A. Similarity Graph Fusion

SGF fuses multiple similarity graphs into one. If full similarity graphs are available, we first construct \( kNN \) similarity graphs, which means that only the edges connecting a node and its \( k \)-nearest neighbors are kept on the similarity graph [14]. It is worth noting that the \( kNN \) graphs used in our algorithms are slightly different from the usual ones. The difference is that, when constructing the \( kNN \) graph for a view, we keep the edge connecting nodes \( x_i \) and \( x_j \) if \( x_i \) is among the \( k \)-nearest neighbors of \( x_j \) in any view. Then, the positions of nonzero elements in the similarity matrices for all views will be the same (i.e., the index set \( F \)), and we use them to construct the MVDR. Note that this will cause the number of nonzero elements in each row of the learned unified matrix greater than \( k \). Hence, we select the \( kNNs \) for each node after learning the unified graph.

As \( kNNs \) capture the local structure by preserving the edges of the nearest neighbors, we further normalize these edges in \( kNN \) distance graphs and strengthen a small portion of strong edges (associated with low distance) from a global perspective before learning the unified graph. Let \( D \) be the edges in a \( kNN \) distance graph, and let \( \mu = \text{mean}(D) \) and \( \sigma = \text{std}(D) \) be the mean and standard deviation of these edges, respectively. Without loss of generality, the edges that are \( 1 \sigma \) lower than the mean distance are considered the strong edges and will be set to zero. Formally, the normalization for \( kNN \) graph is

\[
D = \max((D - \mu + \sigma)/\sigma, 0) = \max((D - \mu)/\sigma + 1, 0)
\]

(27)

where all operations are elementwise. Note that the distance of 0 will be transformed to the maximum similarity of 1 by means of the Gaussian kernel mapping, and hence, the strong edges that are set to 0 are “strengthened.” More importantly, the normalization of \( kNN \) graphs provides a baseline for the graph learning algorithm to compare edges from multiple views with different mean and different standard deviations, which aids the process of fusing them into one unified graph.

Finally, we perform SC on the final unified graph to obtain the clustering results. The SGF algorithm for SC is summarized in Algorithm 4.

| Algorithm 4 SGF for SC |
|------------------------|
| **Input:** Dataset with \( v \) views \( \mathcal{X} = \{X^{(1)}, \ldots, X^{(v)}\} \), number of clusters \( n_c \), \( \beta, \gamma \) (for Algorithm 3), \( k \) (number of \( kNN \)) |
| **Output:** Cluster indicator vector \( c \) |
| 1: Construct \( kNN \) distance graphs \( \{W^{(1)}, \ldots, W^{(v)}\} \) that share neighbors across views |
| 2: Apply Eq. (27) to \( W^{(i)} \) for \( i = 1, \ldots, v \) |
| 3: Apply Gaussian kernel function to transform \( kNN \) distance graphs to \( kNN \) similarity graphs |
| 4: Use Algorithm 3 to obtain the unified similarity matrix \( S \) from the \( kNN \) similarity graphs |
| 5: Keep the \( k \) largest elements at each row of \( S \) and set other elements to 0 |
| 6: \( S \leftarrow (S + S^T)/2 \) |
| 7: Perform spectral clustering on \( S \) to obtain cluster indicator vector \( c \) |

B. Distance (Dissimilarity) Graph Fusion

DGF learns the unified graph directly from multiple distance (dissimilarity) graphs since fusion of distance may better preserve the relationship between nodes than the fusion of similarity. We know that distance is transformed to similarity with a kernel (similarity) function, typically with the Gaussian kernel \( k(x_i, x_j) = \exp(-(d(x_i, x_j))^2/(2\rho^2)) \), where \( d(x_i, x_j) \) is the distance between \( x_i \) and \( x_j \) under some metric. The kernel function may bias the intrinsic relationship between nodes in the original graphs and exert a negative influence on the consistent graph learning process. Therefore, we suggest directly learning a unified graph from the distance (dissimilarity) graphs of all views. Then, we apply a kernel (similarity) function to the learned unified graph to transform distance to similarity, that is, lines 3 and 4 in Algorithm 4 become as follows.

3: Use Algorithm 3 to obtain the unified distance matrix from the \( kNN \) distance graphs |

4: Apply Gaussian kernel function to transform the unified distance graph to unified similarity graph \( S \)

A natural question in practice is how to measure the distance (dissimilarity) between data points (nodes). A widely used metric is Euclidean distance, while the choice largely depends on the applications. If the features are words frequency, the cosine distance is more suitable than the Euclidean distance and it is calculated as \( d(x_i, x_j) = 1 - x_i^T x_j/\|x_i\|\|x_j\| \).

VI. EXPERIMENTS

We perform extensive experiments to compare the two proposed graph learning-based multi-view SC algorithms, namely, SGF and DGF, against seven state-of-the-art multi-view SC algorithms, namely, co-regularized SC (CoReg) [26], robust multi-view spectral clustering (RMSC) [32], affinity aggregation for spectral clustering (AASC) [31], weighted multi-view spectral clustering based on spectral perturbation (WMSC) [28], multi-view clustering via adaptively weighted
procrustes (AWP) [30], graph learning for multi-view clustering (MVGL) [2], and multi-view consensus graph clustering (MCGC) [4]. In addition, the conventional SC [13] is also performed on each view of the datasets, and the best single-view SC performance is reported.

A. Datasets and Evaluation Metrics

Except for the datasets used in the preliminary version of this article, we conduct experiments on more datasets in this article, with a total of 12 datasets as we now introduce. The ORL dataset contains 400 face images of 40 distinct subjects. The Yale dataset contains 165 grayscale images of 15 individuals. The Reuters dataset contains 1200 documents, where each document is in five languages (views). The BBCSport dataset consists of 544 documents from the BBC Sport website. The NUS-WIDE dataset contains multi-view features extracted from images of the NUS-WIDE-OBJECT dataset reported in [40]. The Reuters-21578 dataset is a collection of documents that appeared on Reuters news in 1987. The MSRC-v1 dataset [41] contains 240 pixelwise labeled images. The CiteSeer dataset contains 3312 documents. The ALOI dataset is a collection of 110,250 images of 1000 small objects [42]. Since it is too large for some algorithms such as CoReg and RMSC, we follow Houle et al. [43] to use a subset. The Flower17 dataset consists of images of 17 categories of flower, and the multi-view features are extracted by Nilsback and Zisserman [44], [45]. The Caltech101 dataset consists of 544 documents from the BBC Sport dataset.

The Yale dataset contains 165 grayscale images of 5 individuals. The Reuters dataset contains 1200 documents, where each document is in five languages (views). The BBCSport dataset consists of 544 documents from the BBC Sport website. The NUS-WIDE dataset contains multi-view features extracted from images of the NUS-WIDE-OBJECT dataset reported in [40]. The Reuters-21578 dataset is a collection of documents that appeared on Reuters news in 1987.

The MSRC-v1 dataset contains 240 pixelwise labeled images. The CiteSeer dataset contains 3312 documents. The ALOI dataset is a collection of 110,250 images of 1000 small objects [42]. Since it is too large for some algorithms such as CoReg and RMSC, we follow Houle et al. [43] to use a subset. The Flower17 dataset consists of images of 17 categories of flower, and the multi-view features are extracted by Nilsback and Zisserman [44], [45]. The Caltech101 dataset consists of 544 documents from the BBC Sport dataset.

The statistics of the datasets are summarized in Table I and more details of these datasets can be found in the Supplementary Material.

1 The Yale dataset contains 165 grayscale images of 15 individuals. The Reuters dataset contains 1200 documents, where each document is in five languages (views). The BBCSport dataset consists of 544 documents from the BBC Sport website.

The ALOI dataset is a collection of 110,250 images of 1000 small objects [42]. Since it is too large for some algorithms such as CoReg and RMSC, we follow Houle et al. [43] to use a subset. The Flower17 dataset consists of images of 17 categories of flower, and the multi-view features are extracted by Nilsback and Zisserman [44], [45]. The Caltech101 dataset consists of 544 documents from the BBC Sport dataset.

The Yale dataset contains 165 grayscale images of 15 individuals. The Reuters dataset contains 1200 documents, where each document is in five languages (views). The BBCSport dataset consists of 544 documents from the BBC Sport website. The NUS-WIDE dataset contains multi-view features extracted from images of the NUS-WIDE-OBJECT dataset reported in [40]. The Reuters-21578 dataset is a collection of documents that appeared on Reuters news in 1987.

The MSRC-v1 dataset contains 240 pixelwise labeled images. The CiteSeer dataset contains 3312 documents. The ALOI dataset is a collection of 110,250 images of 1000 small objects [42]. Since it is too large for some algorithms such as CoReg and RMSC, we follow Houle et al. [43] to use a subset. The Flower17 dataset consists of images of 17 categories of flower, and the multi-view features are extracted by Nilsback and Zisserman [44], [45]. The Caltech101 dataset consists of 544 documents from the BBC Sport dataset.

The statistics of the datasets are summarized in Table I and more details of these datasets can be found in the Supplementary Material.

1 http://cam-orl.co.U.K/facedatabase.html
2 http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html
3 http://lig-membres.imag.fr/grimal/data.html
4 http://mlg.ucd.ie/datasets/segment.html
5 https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+
collection
6 http://lig-membres.imag.fr/grimal/data.html
7 https://elki-project.github.io/datasets/multi_view
8 http://www.robots.ox.ac.U.K/~vgg/data/flowers/17/
9 http://www.vision.caltech.edu/Image_Datasets/Caltech101/
10https://archive.ics.uci.edu/ml/datasets/Multi+Features

TABLE I
STATISTICS OF THE REAL-WORLD DATASETS

| Dataset     | # views | # classes | # instances |
|-------------|---------|-----------|-------------|
| ORL         | 3       | 40        | 400         |
| Yale        | 3       | 15        | 165         |
| Reuters     | 5       | 6         | 1200        |
| BBCSport    | 2       | 5         | 544         |
| NUS-WIDE    | 5       | 31        | 2000        |
| Reuters-21578 | 5 | 6          | 1500        |
| MSRC-v1     | 5       | 7         | 210         |
| CiteSeer    | 2       | 6         | 3312        |
| ALOI        | 4       | 100       | 10800       |
| Flower17    | 7       | 17        | 1360        |
| Caltech101  | 6       | 102       | 9144        |
| UCI Digits  | 6       | 10        | 2000        |

purity [48] are used to measure the clustering performance. ACC is defined by

\[
\text{ACC} = \frac{1}{n} \sum_{i=1}^{n} \delta(r_i, \text{map}(r_i))
\]  

(28)

where \(n\) data points belong to \(k\) clusters, \(r_i\) and \(r_i\) denote the ground-truth label and the predicted label of the \(i\)th sample, respectively, denoting the predicted clustering label, and the Dirac delta function \(\delta(x, y) = 1\) if \(x = y\), else 0. Also, \(\text{map}(r_i)\) is the optimal mapping function that permutes the predicted labels to match the ground-truth labels, which is found by the Hungarian algorithm [49]. After the best permutation by the Hungarian algorithm, let \(C_i\) and \(T_i\) denote the set of samples in the \(i\)th cluster by the clustering and the ground truth, respectively. Let \(a_i = |C_i|\) and \(b_i = |T_i|\) be the number of entries in the sets, and \(n_{ij} = |C_i \cap T_j|\). NMI is defined by

\[
\text{NMI} = \frac{\sum_{i=1}^{v} \sum_{j=1}^{v} n_{ij} \log \frac{n_{ij}}{n_i n_j}}{\sqrt{\left( \sum_{i=1}^{v} a_i \log \frac{a_i}{n} \right) \left( \sum_{i=1}^{v} b_i \log \frac{b_i}{n} \right)}},
\]  

(29)

Purity quantifies the extent to which a cluster \(C_i\) contains entities from only one partition \(T_j\) and is defined by

\[
\text{Purity} = \frac{1}{n} \sum_{i=1}^{v} \max_{j=1}^{n} n_{ij}.
\]  

(30)

ARI is defined by

\[
\text{ARI} = \frac{\sum_{ij} (n_{ij})^2 - \left( \sum_{i} (n_i)^2 \sum_{j} (n_j)^2 \right) / (n^2)}{\frac{1}{2} \left( \sum_{i} (n_i)^2 + \sum_{j} (n_j)^2 \right) - \left( \sum_{i} (n_i)^2 \sum_{j} (n_j)^2 \right) / (n^2)},
\]  

(31)

where all summations are from 1 to \(v\).

B. Experimental Setup

We downloaded the source code of AWP, AASC, MVGL, and MCGC from the authors’ websites and implement other algorithms following the instruction in the original papers. We conduct all experiments with MATLAB R2019b on a machine with an Intel Core i9-9960X 16-core CPU and 128-GB RAM. For the algorithms which use the Laplacian matrices of graphs, we adopt the symmetrically normalized Laplacian \(L_{sym} = I - D^{-1/2}SD^{-1/2}\), where \(S\) and \(D\) are the adjacency matrix and degree matrix of the graph, respectively [13], [14]. We fix the number of the nearest neighbors in \(k\)NN to 6 in all experiments. For the algorithms with parameter(s), which include CoReg (one parameter), RMSC (one parameter), MCGC (one parameter), WMSC (two parameters), and DGF and SFG (two parameters), we use grid search to test the parameter(s) of these algorithms on the grids \(10^{-5}, 10^{-4}, \ldots, 10^{-3}\)^m, where \(m\) is the number of parameters of the algorithm, and we report the scores with the best parameter(s) (i.e., the parameter(s) achieving the highest NMI) found on each dataset. Note that the grids contain values that are very close to the parameters suggested by the authors. Thus, all algorithms should exhibit their best performance in
the experiments. We set the weights $\lambda_i$ for each view to 1 in DGF and SGF, without considering the importance of different views.

We run all algorithms ten times and report the average scores and standard deviation. If $K$-means clustering is used in any algorithm, we run $K$-means ten times and set its maximum
TABLE III
AVERAGE RUNNING TIME AND STANDARD DEVIATION (IN SECONDS) OVER TEN RUNS BY DIFFERENT METHODS, WHERE THE TIME IS RECORDED FROM CONSTRUCTING DISTANCE GRAPHS TO OBTAINING CLUSTERING RESULT (I.E., THE ENTIRE PROCESS); DGF-0 AND SGF-0 STAND FOR THE RUNNING TIME OF THE DGF AND SGF ALGORITHMS, RESPECTIVELY, IN THE PRELIMINARY VERSION OF THE ARTICLE [22]; THE RUNNING TIME OF THE FASTEST TWO MULTI-VIEW CLUSTERING METHODS IN EACH ROW (EXCLUDING SINGLE-VIEW SC) IS HIGHLIGHTED IN BOLD

| Dataset  | AASC | AWP | CoReg | MCGC | MVGL | RMSC | WMSC | SC (best) | DGF-0 | SGF-0 | DGF | SGF |
|----------|------|-----|-------|------|------|------|------|----------|-------|-------|-----|-----|
| ORL      | 0.36 ± 0.00 | 0.26 ± 0.00 | 7.11 ± 0.19 | 1.05 ± 0.02 | 3.97 ± 0.02 | 3.00 ± 0.02 | 0.32 ± 0.00 | 0.26 ± 0.00 | 0.26 ± 0.00 | 0.39 ± 0.00 | 0.45 ± 0.00 | 0.39 ± 0.01 | 0.40 ± 0.00 |
| Yale     | 0.14 ± 0.00 | 0.07 ± 0.00 | 0.41 ± 0.03 | 0.87 ± 0.01 | 1.00 ± 0.02 | 0.11 ± 0.00 | 0.09 ± 0.00 | 0.07 ± 0.00 | 0.12 ± 0.00 | 0.12 ± 0.00 | 0.09 ± 0.00 | 0.12 ± 0.00 |
| Reuters  | 0.66 ± 0.01 | 0.62 ± 0.01 | 1.97 ± 0.04 | 7.77 ± 0.02 | 36.8 ± 0.02 | 12.9 ± 0.02 | 0.66 ± 0.00 | 0.50 ± 0.00 | 1.01 ± 0.01 | 0.87 ± 0.01 | 0.62 ± 0.00 | 0.63 ± 0.01 |
| BBCSport | 0.13 ± 0.01 | 0.10 ± 0.00 | 3.75 ± 0.05 | 0.95 ± 0.02 | 7.62 ± 0.02 | 0.50 ± 0.01 | 0.12 ± 0.00 | 0.10 ± 0.00 | 0.29 ± 0.00 | 0.16 ± 0.00 | 0.11 ± 0.00 | 0.11 ± 0.00 |
| NUS-WIDE | 2.47 ± 0.03 | 0.90 ± 0.02 | 14.8 ± 0.06 | 55.8 ± 0.11 | 54.3 ± 0.05 | 3.68 ± 0.01 | 1.24 ± 0.02 | 0.60 ± 0.02 | 2.94 ± 0.02 | 6.37 ± 0.02 | 1.12 ± 0.02 | 1.12 ± 0.02 |
| Reuters-21578 | 1.70 ± 0.01 | 1.64 ± 0.02 | 5.75 ± 0.05 | 10.00 ± 0.02 | 61.8 ± 0.04 | 5.31 ± 0.02 | 1.69 ± 0.01 | 1.46 ± 0.00 | 2.14 ± 0.01 | 2.33 ± 0.01 | 1.56 ± 0.01 | 1.54 ± 0.01 |
| MSRC-v1  | 0.22 ± 0.01 | 0.04 ± 0.00 | 2.17 ± 0.02 | 0.40 ± 0.01 | 1.36 ± 0.02 | 0.37 ± 0.01 | 0.06 ± 0.00 | 0.04 ± 0.00 | 0.07 ± 0.00 | 0.09 ± 0.00 | 0.08 ± 0.00 | 0.07 ± 0.00 |
| CiteSeer | 1.44 ± 0.01 | 1.18 ± 0.01 | 118 ± 0.09 | 36.6 ± 0.02 | 203 ± 0.07 | 419 ± 1.6 | 1.32 ± 0.01 | 1.01 ± 0.01 | 2.87 ± 0.03 | 2.41 ± 0.03 | 1.20 ± 0.02 | 1.16 ± 0.02 |
| ALOI     | 39.4 ± 41 | 36.1 ± 18 | 2545 ± 17.0 | 713 ± 21 | 5330 ± 6.1 | 388 ± 1.4 | 71.7 ± 0.08 | 18.7 ± 0.6 | 27.2 ± 0.07 | 25.9 ± 0.02 | 9.53 ± 0.03 | 9.67 ± 0.03 |
| Flower17 | 0.70 ± 0.01 | 0.38 ± 0.01 | 50.0 ± 0.19 | 24.3 ± 0.04 | 69.3 ± 0.04 | 9.21 ± 0.21 | 0.49 ± 0.00 | 0.10 ± 0.01 | 1.33 ± 0.01 | 1.16 ± 0.00 | 0.62 ± 0.01 | 0.60 ± 0.01 |
| Caltech101 | 42.5 ± 14 | 38.5 ± 0.03 | 294 ± 0.60 | 1129 ± 4.9 | 7274 ± 8.5 | 2062 ± 0.6 | 41.2 ± 0.07 | 30.2 ± 0.07 | 54.8 ± 0.25 | 78.1 ± 1.2 | 34.1 ± 0.32 | 34.5 ± 0.29 |
| UCI Digits | 1.02 ± 0.01 | 1.33 ± 0.03 | 352 ± 0.88 | 15.7 ± 0.03 | 79.5 ± 0.05 | 22.5 ± 1.6 | 3.08 ± 0.01 | 0.60 ± 0.01 | 1.79 ± 0.00 | 6.10 ± 0.00 | 0.81 ± 0.01 | 0.79 ± 0.02 |

D. Comparisons With the Preliminary Version

We also compare the clustering results of the revised DGF and SGF algorithms against the DGF and SGF in the preliminary versions (denoted by DGF-0 and SGF-0) [22]. As shown in Section A.4 and Table 4 in the Supplementary Material, the revised algorithms obtain higher clustering scores than the preliminary ones on most tested datasets, sometimes surpassing the preliminary ones by a large margin. Moreover, the revised algorithms are more stable than the preliminary ones, performing reasonably well across a wide range of hyperparameters (further details are provided in Fig. 2), while the preliminary algorithms perform less robustly on some datasets or for certain hyperparameters.

E. Empirical Convergence Analysis

We propose an alternating minimization scheme to solve the optimization problem in the proposed graph learning framework by dividing it into three subproblems. As we have analyzed in Section IV-G1, alternatively solving the three subproblems can converge to a local minimizer of the overall optimization problem (9) in most cases. Empirically, we find that the proposed optimization approach always converges in all experiments with different hyperparameters on various datasets, mostly within a few iterations, which demonstrates its reliability and efficiency. The convergence curves are shown in Fig. 1 in the Supplementary Material.

F. Empirical Complexity Analysis

In Section IV-G2, we give a theoretical result that the time complexity of the graph learning algorithm can be reduced to \(O(n_1 v^2)\) or \(O(kn v^2)\), where \(n_1, n, v, \) and \(k\) are, respectively, the number of edges, nodes, views, and nearest neighbors in \(k\)NN. For each of the three factors \(k, n, v\) in the complexity, we fix the other two factors and vary only the one being investigated in the experiments. We plot the results in Figs. 2–4 in the Supplementary Material. As shown in the figures, there is a clear linear trend of the running time of the proposed algorithm with respect to \(k\) and \(n\), respectively.
and a quadratic trend with respect to $v$, in both DGF and SGF. Since the number of edges $n_e = kn$ and the empirical running time is linear in both $k$ and $n$, it can be concluded that the running time is linear in $n_e$, which aligns well with the theoretical result under our assumptions in Section IV-G2. Due to these linear trends, our algorithm is among the most efficient multi-view clustering algorithms, as shown in Table III.

G. Parameters Sensitivity

In this section, we conduct experiments to test the influence of the two hyperparameters $\beta$ and $\gamma$ in the proposed multi-view graph learning algorithm. Since $\lambda$ is meant to provide the users with an opportunity to incorporate prior knowledge through it, we use $\lambda_i = 1$ in this article for all views as no prior knowledge is involved. We test $(\beta, \gamma)$ on the grid $\{10^{-5}, 10^{-4}, \ldots, 10^5\} \times \{10^{-5}, 10^{-4}, \ldots, 10^5\}$. The results with respect to varying parameters $\beta$ and $\gamma$ are shown in Fig. 2, which demonstrates that the performance of the proposed framework is stable across a wide range of parameters. We emphasize that even without parameter tuning, the framework still achieves generally superior performance against the state of the art. Please see the Supplementary Material for the experiments.

H. Algorithmic Efficiency Comparison

We record the running time of each algorithm in our experiments and report the results in Table III. We can see that DGF and SGF are among the two fastest multi-view algorithms on some datasets and run comparably fast on the other datasets against other algorithms. Besides, we also make an efficiency comparison with the DGF and SGF algorithms in the preliminary version of the article [22]. As shown in Table III, the revised algorithms run clearly faster than the preliminary DGF and SGF algorithms [22] (denoted by DGF-0 and SGF-0) on the 12 datasets.

DGF and SGF can be divided into two parts, which correspond to consistent graph learning (Algorithm 3) and SC on the learned uniform graph, respectively. Experiments show that the proposed graph learning algorithm is quite efficient compared to the running time of the entire process of multi-view clustering (see Table 1 in the Supplementary Material). More interestingly, DGF and SGF run faster than single-view SC on the ALOI dataset (in Table III). This may be because the learned similarity graphs in DGF and SGF have larger eigengap [50], which makes the eigen-decomposition (ED) easier and faster (intuitively, this may result from that the nodes on the learned graph are well clustered).

Moreover, all comparing multi-view algorithms in this article require ED or singular value decomposition (SVD) of an $n$-by-$n$ matrix at least two times (some require at least $v$ times), while the proposed DGF and SGF perform ED only once (which is in the SC on the unified graph). Since ED and SVD have at least $\Omega(n^2)$ time complexity [51], which is the complexity bottleneck in many multi-view clustering algorithms, our multi-view clustering algorithms could run several times faster than the comparing multi-view algorithms on very large datasets.
I. Consistency and Inconsistency of Multi-view Learning

In Fig. 1, we see that the learned unified graph by SGF is “cleaner” than any single view and clearly contains the consistent parts of all views. By the theoretical motivation of the proposed framework in Section III, it could learn a better graph if the consistent part of multi-view graphs is the dominant part in most views, i.e., the inconsistent parts are sparse across views. As shown in Fig. 1, although the fourth single view is very noisy and very inconsistent with other views, the first three views exhibit consistent cluster patterns, and thus, the inconsistency is still sparse across views. In the experiments, we do see, on the Caltech101 dataset, that the single-view SC on the best view is slightly better than our algorithms (and is clearly better than other multi-view clustering methods). However, this does not undermine the advantages of the proposed framework where multi-view consistency and inconsistency are simultaneously exploited to learn a better unified graph, which can be verified by the significant performance improvements of our approach on many datasets compared to the best single-view SC.

VII. Conclusion

This article presents a novel multi-view graph learning approach, which for the first time simultaneously and explicitly models multi-view consistency as well as multi-view inconsistency in a unified optimization model, where multi-view consistency can be iteratively learned and fused into a unified graph as the multi-view inconsistency is automatically identified and removed. To optimize the objective, we design an efficient algorithm by exploiting the structures of the QPs in the problem. The proposed algorithm has linear time complexity in the number of edges on the learned graph, even though exactly solving the problem is NP-hard. We further extend the proposed framework to two graph fusion versions, which correspond to DGF and SGF. Experimental results demonstrate the superiority, efficiency, and robustness of the proposed algorithms against several state-of-the-art multi-view SC algorithms on a variety of real-world datasets. Remarkably, it maintains its good performance even without dataset-specific hyperparameter tuning. In the future, we can explicitly learn the importance weight of each view [i.e., \( \lambda \)'s in Problem (9)] in the model formulation by considering the structures in some clustering tasks. Besides, it would be interesting to investigate how to incorporate the proposed graph learning framework into the SC process, where the learning of the good graphs and the clustering can be done simultaneously.

References

[1] C. Zhang, Q. Hu, H. Fu, P. Zhu, and X. Cao, “Latent multi-view subspace clustering,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 4333–4341.
[2] K. Zhan, C. Zhang, J. Guan, and J. Wang, “Graph learning for multi-view clustering,” IEEE Trans. Cybern., vol. 48, no. 10, pp. 2887–2895, Oct. 2018.
[3] K. Zhan, C. Niu, C. Chen, F. Nie, C. Zhang, and Y. Yang, “Graph structure fusion for multi-view clustering,” IEEE Trans. Knowl. Data Eng., vol. 31, no. 10, pp. 1984–1993, Oct. 2019.
[4] K. Zhan, F. Nie, J. Wang, and Y. Yang, “Multiview consensus graph clustering,” IEEE Trans. Image Process., vol. 28, no. 3, pp. 1261–1270, Mar. 2019.
[5] F. Nie, J. Li, and X. Li, “Self-weighted multiview clustering with multiple graphs,” in Proc. Int. Joint Conf. Artif. Intell. (IJCAI), 2017, pp. 2564–2570.
[6] X. Liu et al., “Late fusion incomplete multi-view clustering,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 10, pp. 2410–2423, Oct. 2019.
[7] S. Huang, Z. Kang, I. W. Tsang, and Z. Xu, “Auto-weighted multi-view clustering via kernelized graph learning,” Pattern Recognit., vol. 88, pp. 174–184, Apr. 2019.
[8] S. Sun, X. Xie, and C. Dong, “Multiview learning with generalized eigenvalue proximal support vector machines,” IEEE Trans. Cybern., vol. 49, no. 2, pp. 688–697, Feb. 2018.
[9] S. Sun, Y. Liu, and L. Mao, “Multi-view learning for visual scene recognition with maximum entropy discrimination and deep features,” Inf. Fusion, vol. 50, pp. 43–53, Oct. 2019.
[10] X. Liu et al., “Efficient and effective regularized incomplete multi-view clustering,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 8, pp. 2634–2646, Aug. 2021.
[11] A. K. Jain, “Data clustering: 50 years beyond K-means,” Pattern Recognit. Lett., vol. 31, no. 8, pp. 651–666, 2010.
[12] J. Shi and J. Malik, “Normalized cuts and image segmentation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 8, pp. 888–905, Aug. 2000.
[13] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in Proc. Adv. Neural Inf. Process. Syst., 2002, pp. 849–856.
[14] U. von Luxburg, “A tutorial on spectral clustering,” Statist. Comput., vol. 17, no. 4, pp. 395–416, 2007.
[15] F. Nie, X. Wang, and H. Huang, “Clustering and projected clustering with adaptive neighbors,” in Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2014, pp. 977–986.
[16] F. Nie, X. Wang, M. I. Jordan, and H. Huang, “The constrained Laplacian rank algorithm for graph-based clustering,” in Proc. 36th AAAI Conf. Artif. Intell., 2016, pp. 1–8.
[17] Z. Lin, Z. Kang, L. Zhang, and L. Tian, “Multi-view attributed graph clustering,” IEEE Trans. Knowl. Data Eng., early access, Aug. 6, 2021, doi: 10.1109/TKDE.2021.3101227.
[18] Z. Li, C. Tang, X. Liu, X. Zheng, W. Zhang, and E. Zhu, “Consensus graph learning for multi-view clustering,” IEEE Trans. Multimedia, vol. 24, pp. 2461–2472, 2021.
[19] Z. Kang, H. Pan, S. C. H. Hoi, and Z. Xu, “Robust graph learning from noisy data,” IEEE Trans. Cybern., vol. 50, no. 5, pp. 1833–1843, May 2019.
[20] Z. Kang, Z. Lin, X. Zhu, and W. Xu, “Structured graph learning for scalable subspace clustering: From single view to multiview,” IEEE Trans. Cybern., early access, Mar. 17, 2021, doi: 10.1109/TCYB.2021.3061660.
[21] A. Bojchevski, Y. Matkovic, and S. Grosse-Rieger, “Robust spectral clustering for noisy data: Modeling sparse corruptions improves latent embeddings,” in Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2017, pp. 737–746.
[22] Y. Liang, D. Huang, and C.-D. Wang, “Consistency meets inconsistency: A unified graph learning framework for multi-view clustering,” in Proc. IEEE Int. Conf. Data Mining (ICDM), Nov. 2019, pp. 1204–1209.
[23] L. T. H. An and P. D. Tao, “A branch and bound method via DC optimization algorithms and ellipsoidal technique for box constrained nonconvex quadratic problems,” J. Global Optim., vol. 13, no. 2, pp. 171–206, 1998.
[24] S. Bickel and T. Scheffer, “Multi-view clustering,” in Proc. ICDM, vol. 4, Nov. 2004, pp. 19–26.
[25] A. Blum and T. Mitchell, “Combining labeled and unlabeled data with co-training,” in Proc. 11th Annu. Conf. Comput. Learn. Theory (COLT), 1998, pp. 92–100.
[26] A. Kumar, P. Rai, and H. Daume, “Co-regularized multi-view spectral clustering,” in Proc. Adv. Neural Inf. Process. Syst., 2011, pp. 1413–1421.
[27] Y. Wang, W. Zhang, L. Wu, X. Lin, M. Fang, and S. Pan, “Iterative views agreement: An iterative low-rank based structured optimization method to multi-view spectral clustering,” 2016, arXiv:1608.05560.
