# Self Weighted Multiplex Modularity Maximization for Multiview Clustering

**Noureddine Henka** NOUREDDINE.HENKA@RTE-FRANCE.COM
*L2S and R&D RTE*
**Mohamad Assaad** MOHAMAD.ASSAAD@CENTRALESUPELEC.FR
*L2S, Gif-sur-Yvette, France*
**Sami Tazi** SAMI.TAZI@RTE-FRANCE.COM
*R&D RTE, La defense, France*

**Editors:** Berrin Yanıkoğlu and Wray Buntine

## Abstract

In response to the challenge of representing data from multiple sources, researchers have proposed the use of multiplex graphs as a solution. Multiplex graphs are particularly useful for representing multi-view data, where each layer represents a specific type of interaction. Pillar community detection of multiplex graphs is a clustering application that computes groups of vertices across all layers. Modularity maximization is a popular technique for graph clustering, which has been generalized to multiplex graphs. However, this generalization did not consider the importance of each layer in pillar clustering. This paper presents a new technique called Self Weighted Multiplex Modularity (SWMM), which optimizes the weights associated with each layer and the partition that maximizes the multiplex modularity. The paper proposes two optimization methods, iterative and direct, and demonstrates the effectiveness and robustness of the technique in accurately retrieving clusters even when data is highly missing.

**Keywords:** Multiview data, Multiplex Network, Self Weighted Multiplex Modularity, Impartial Representation, Incomplete Representation

## 1. Introduction

Network modeling is commonly used to represent complex systems across various fields, including the social, biological, power grid, and telecommunication systems. This approach simplifies the understanding of complex behaviors and has been explored in several studies Torres et al. (2020); Zweig (2016). In recent years, clustering has become an increasingly popular research topic. Community detection algorithms have been widely developed to address the challenge of identifying similar vertices within a network Fortunato (2010). As the amount of data continues to increase, traditional techniques focused on a single network representation are becoming insufficient. This has led to a growing demand for algorithms that handle multi-relational datasets, as noted in N.Musmeci (2017). Various prior studies have investigated multiplex networks to address the challenge of multiple relation representations, including recent works such as Kong et al. (2021). Multiplex networks are modeling techniques that account for various interactions among individuals of the same nature, with each interaction type represented by a distinct layer. This approach

enhances the representation of multi-relational systems and provides greater flexibility for handling the heterogeneous relationships of networks.

Community detection algorithms have been utilized in diverse applications to cluster multiplex graphs, producing various types of clustering outcomes. As explained in Magnani et al. (2019), the set of shared vertices between the layers is called an actor, while each layer's vertex is called a node. Therefore, when the resulting communities comprise a set of actors where vertices belonging to the same actor are grouped together, the method is known as pillar community detection , which is the central theme of our study.

Multi-view data, which consists of individuals represented by different similarity measures, with each similarity referred to as a "view" Chao et al. (2017), were widely modeled by multiplex networks Saha (2013). In this representation, each view is represented by a distinct layer. Therefore, clustering multi-view data can be regarded as pillar multiplex community detection on the corresponding multiplex network Chen et al. (2022). This method is particularly advantageous for representing complex system structures with networks that offer a faithful manifold representation of the data.

A good community on a network is commonly characterized by dense intra-community links and weak inter-community links Choudhury and Paul (2013). One popular metric for evaluating communities' quality is the modularity, which compares the density of links within communities to that of a random graph Newman (2006). While the maximization of modularity has been widely used for community detection in single-layer networks, recent works have focused on generalizing the modularity to multi-layer networks, particularly for computing multiplex network community detection Mucha et al. (2010). However, in many community detection applications involving multiplex networks or multi-view data, the layers or views may have different levels of importance for the clustering Nie et al. (2017). To address this issue, we propose a new method called the Self Weighted Multiplex Modularity (SWMM) for multiplex and multi-view clustering, which can automatically compute the weights for each layer and detect pillar communities in multiplex networks. Our approach, which includes two techniques (iterative and direct), is also robust to noised layers, incomplete data within the layers, and outperforms the existing methods, notably multi-view network-based clustering algorithms.

The paper is structured as follows. In Section 2, we review the existing literature on multiplex community detection and multi-view clustering. We then provide in Section 3, a brief overview of the single-layer modularity maximization, and then we explain the SWMM model. We also discuss the optimization process, where both iterative and direct optimization strategies are presented. In Section 4, the experiment follows, where we show the effectiveness of SWMM at retrieving clusters, demonstrate the incorporation of weights to improve clustering performance, and illustrate its performance with high missing data. Finally, in Section 5, we conclude the work by a discussion.

## 2. Related Work

### 2.1. Multi-view graph clustering

Multi-view data represents multiple relationships between individuals, where each view captures a specific type of similarity. Previous works on multi-view clustering have mainly focused on network representation Zhan et al. (2018), in which a network models the simi-

larity in each view. A unified representation is then computed to fuse all layers associated with views. Some of these works apply traditional algorithms on the unified representation to compute the multi-view clustering, while others penalize the unified representation for obtaining $K$ connected components, in which each component represents a cluster, and $K$ is the number of clusters. Spectral clustering has also been adapted for multi-view data clustering Xia et al. (2014). It is mainly based on low spectral dimension embedding, followed by applying a K-means algorithm on the embedded representation. However, these techniques may be susceptible to non-informative views within the data, and they may need to consider the complete representation of individuals within the views, while additionally facing the potential loss of the local representation of each layer after their merger.

### 2.2. Multiplex graph community detection

Multiplex graphs have become a popular research area due to their ability to represent multiple interactions among individuals. However, traditional community detection algorithms designed for mono-layer networks need to be improved for computing communities in multiplex networks. As mentioned previously, our focus is on pillar community detection, which involves assigning vertices of the same actor to the same cluster. The algorithms of pillar multiplex community detection are classified into three categories: *flattening methods*, *layer by layer*, and *direct multi-layer method*. The multiplex flattening consists of merging layers of multiplex networks into a mono-layer network. Clusters are computed by traditional community detection algorithms on the flattened network Berlingerio et al. (2011); Kim et al. (2016). However, once the multiplex graph is flattened, the local structure of edges may be biased towards the most appearing ones, influencing the resulting clusters. The layer-by-layer technique consists of processing each layer of the multiplex graph independently using a traditional community detection algorithm. Then, the multiplex clustering is performed by merging the clusters from each layer, such as using pattern mining or consensus approach Tagarelli et al. (2017). However, the result may not be a pillar, and once the clusters are computed, the similarity between the layer may be loosed. The direct technique computes community detection on the multiplex network. It includes matrix factorization Papalexakis et al. (2013), and extension of mono-layer algorithm to multiplex such as random walk De Domenico et al. (2015), density based Afsarmanesh and Magnani (2016) and label propagation Boutemine and Bouguessa (2017).

The modularity maximization technique has been extended to community detection of multiplex graphs. In Mucha et al. (2010), the authors propose a generalized Louvain to maximize the modularity for the multiplex graphs. However, these models fail to consider the importance of each layer to maximize the multiplex modularity, and they do not address the issue of incomplete representation. In the context of multi-view clustering, it has been observed that different views possess varying degrees of importance in achieving the optimal clustering, as highlighted in Wang et al. (2020). To address these limitations, we propose a self-weighted multiplex modularity for pillar community detection that computes weights to account for the importance of each layer. Furthermore, we demonstrate the relevance of this approach in addressing the incomplete representation issue when individuals are presented in partial layers.

## 3. Self Weighted Modularity Maximization

### 3.1. Single-layer community detection by modularity maximization

Consider a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{W})$, where $\mathbf{V}$ is the set of vertices, $\mathbf{E}$ is the set of edges, and $\mathbf{W}$ is the set of weights. Let $\mathbf{A}$ stand for the adjacency matrix of $\mathbf{G}$, such that $A_{ij}$ denotes the weight of the edge between vertices $i$ and $j$ respectively, while being zero otherwise. Given a partition $\mathbf{C}$ of the vertex set $\mathbf{V}$, the modularity is expressed as:

$$Q = \frac{1}{2m} \sum_{c \in \mathbf{C}} \sum_{i,j \in c} \left[ \mathrm{A}_{ij} - P_{ij} \right] \tag{1}$$

where $m$ denotes the network density, $c$ is a set of vertices within partition $\mathbf{C}$, and $P_{ij}$ represents the expected edges between vertices $i$ and $j$ under the null model Fortunato (2010), respectively. The null model assumes a random distribution of edges while preserving the degree sequence of the graph. The quality of a given partition can be assessed by comparing the modularity of the observed graph with that of the random graph, with a higher difference indicating a better community. A popular choice of null model for networks with positive edge weights is the Newman–Girvan null network, where $Pij = k_i k_j / 2m$, where $k_i, k_j$ is the density of vertices $i$ and $j$ respectively Newman (2006). Therefore, the modularity of a single layer is equivalent to the following:

$$Q = \frac{1}{2m} \sum_{c \in \mathbf{C}} \sum_{i,j \in c} \left[ \mathrm{A}_{ij} - \frac{k_i k_j}{2m} \right] \tag{2}$$

it is also written as :

$$Q = \frac{1}{2m} \sum_{i,j \in V} \left( \mathrm{A}_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{3}$$

where $c_i, c_j$ are the clusters of vertex $i$ and $j$ respectively, and $\delta(c_i, c_j) = 1$ if the vertices $i, j$ belong to the same cluster, and zeros otherwise. The modularity model implies that vertices within the same community should be more densely connected than other vertices in the network. Modularity maximization has been widely employed in computing community detection Dugué and Perez (2015), stated as follows:

$$\max_{c \in \mathbf{C}} \frac{1}{2m} \sum_{i,j \in V} \left( \mathrm{A}_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{4}$$

The problem of maximizing modularity is considered to be NP-hard Brandes et al. (2008), which means that finding the exact solution in polynomial time is unfeasible. Therefore, various heuristic techniques have been developed to approximate the optimal partition, such as greedy methods Blondel et al. (2008), spectral relaxations Newman (2013), integer linear programming Brandes et al. (2008), simulated annealing Guimerà et al. (2004). In the following, we aims to expand the modularity maximization approach to address the issue of multiplex pillar community detection with self-weighted optimization.

### 3.2. Model of Self Weighted Multiplex Modularity

Let consider a multiplex graph denoted as $\mathcal{G} = \{G^1, ..., G^L\}$ having $L$ layer, with $G^l = \{V, E^l\}$ represents a single layer, where $V$ indicates the set of vertices with $|V| = N$, $E^l$ the set of edges within layer, and $l$. $l \in [1, L]$ indicates the layer index. Let $\mathcal{A} = \{A^1, ..., A^L\}$ be the corresponding adjacency matrix of multiplex graph $\mathcal{G}$, where $A^l$ stands for the adjacency matrix of the graph $G^l$. Consider $\mathbf{\Omega} = \{\omega_1, ..., \omega_L\}$ be the set of weights associated to each layer. The SWMM can be formulated as follows:

$$Q_{weighted} = \sum_{l=1}^{L} \omega^l \frac{1}{2m^l} \sum_{i,j \in N} \big(A_{ij}^l - \frac{k_i^l k_j^l}{2m^l}\big)\delta(c_i, c_j) \tag{5}$$

where $A_{ij}^l$ is the weight of the edge between vertices $i$ and $j$ in layer $l$, $k_i^l, k_j^l$ are the density of vertices $i$ and $j$ in layer $l$ respectively, and $m^l$ is the density of $G^l$. The community detection of multiplex graph formulated from SWMM can be compute from the following:

$$\begin{aligned} \max_{\mathbf{\Delta}, \mathbf{\Omega}} \sum_{l=1}^{L} \omega_l \frac{1}{2m^l} \sum_{i,j \in N} \big(A_{ij}^l - \frac{k_i^l k_j^l}{2m^l}\big)\delta(c_i, c_j) \\ st. \sum_{l=1}^{L} \omega_l = 1, \omega_l \geq 0 \end{aligned} \tag{6}$$

$\mathbf{\Delta} = [\delta(c_0, c_0), \delta(c_0, c_1), ..., \delta(c_N, c_{N-1}), \delta(c_N, c_N)]$ is the set of partition variables *i.e.* $i, j \in N, \delta(c_i, c_j) = 1$ if vertices $i, j$ are in the same clusters, and 0 otherwise.

It is important to note that the constraints on $\delta$ that ensure reflexivity, symmetry, and transitivity are crucial for obtaining a valid clustering solution. Although they may be omitted in 6, they should be enforced during the optimization process to obtain a valid solution. These constraints are satisfied through various techniques, such as using a greedy algorithm that gradually builds the clustering solution by satisfying the constraints.

To optimize the previous model, a two-stage iteration can be employed. In the first stage, heuristics are used to maximize the multiplex modularity while fixing the weights. In the second stage, the weights are optimized for a fixed partition obtained in the preceding stage. The optimization of the weights can be expressed as follows:

$$\begin{aligned} \max_{\mathbf{\Omega}} \sum_{l=1}^{L} \omega_l Q_l \\ st. \sum_{l=1}^{L} \omega_l = 1, \omega_l \geq 0 \end{aligned} \tag{7}$$

where $Q_l$ is the modularity of layer $l$ from partition that was computed previously. The optimization of the weights may suffer from a sparse solution. In this context, we introduce

the entropy, which enables control the weight distribution as follows:

$$\begin{aligned} \max_{\mathbf{\Omega}} \sum_{l=1}^{L} \omega_l * Q_l + \eta \mathcal{H}(\mathbf{\Omega}) \\ s.t \sum_{l=1}^{L} \omega_l = 1, \omega_l \geq 0 \end{aligned} \tag{8}$$

where $\mathcal{H}(\mathbf{\Omega})$ is the entropy of weights $\mathbf{\Omega}$, $\eta$ is a hyperparameter allowing to scale between the entropy and the weighted value of the modularity, the set of $\eta$ will be addressed in a subsequent section. Notice that for fixed partition, the $\sum_{l=1}^{L} \omega_l * Q_l$ is a positive linear function subject to linear constraints, and $\mathcal{H}(\mathbf{\Omega})$ is positive concave function. As a result, the overall model to maximize becomes concave.

### 3.3. Optimization of self-weighted multiplex modularity

The community detection on multiplex graph using modularity, corresponds to the partition that maximized the multiplex modularity. Recall that optimizing community detection on a single graph using modularity is known to be an NP-hard problem, which extends to the case of a multiplex graph. In addition to optimizing the partition, the weights also need to be optimized. To address this, we present two optimization techniques. The first technique is an iterative model, where the weights and modularity are optimized alternately. The second technique is a direct approach, where the weights and modularity are optimized simultaneously.

#### 3.3.1. Iterative model

We propose an alternating method. At first, for a giving weights, the Louvain algorithm is used in order to maximize the multiplex modularity. The Louvain algorithm starts by assigning each vertex to its own community. Next, the algorithm computes the gain in multiplex modularity obtained by transferring a vertex to each of its neighbors and its own communities. The gain in multiplex modularity is computed using the following formula:

$$\Delta Q_{multiplex} = \sum_{l=1}^{L} \omega_l \Delta Q_l \tag{9}$$

Where $\Delta Q_l$ is the gain of modularity of layer $l$. Therefore, for any given vertex $i$ with current community $c_i$, the gain of modularity by moving $i$ from community $c_i$ to community $c$ in layer $l$ is computed as :

$$\Delta Q_l(i, c) = \frac{[k^l_{(i,c)} - k^l_{(i,c_i)} + 2\Delta out^l_{(i,c)}]}{2m^l} \tag{10}$$

where $k^l_{(i,c)}$ is the density of vertex $i$ in the community $c$ in layer $l$, $k^l_{(i,c_i)}$ is the density of vertex $i$ in the community $c_i$ for layer $l$, and $\Delta out^l_{(i,c)}$ is the change in the fraction of the total edges weight that is outside the community $c$ when vertex $i$ is moved to the community $c$. The vertex $i$ is moved to the community that maximizes the gain of multiplex

modularity. This process is computed for all vertices and repeated until improvements in multiplex modularity can no longer be achieved. Based on the resulting groups, super-vertices are constructed by merging each community into a single entity, in which self-loop edges represent the intra-community density. This entire process is repeated until no further enhancements in modularity can be made.

For a giving partition, the modularity of a layer $l$ is noted as $Q_l$. The weights are then optimized as follows.

**Proposition 1** *The weights can be calculated using the following formula with the aid of Lagrange multipliers:*

$$\omega_l = \frac{exp^{\frac{Q_l}{\eta}}}{\sum_{l=1}^{L} exp^{\frac{Q_l}{\eta}}} \tag{11}$$

**Proof** The solution of the 8 using Lagrance technique is defined as follows:

$$\mathcal{L}(\mathbf{\Omega}) = \sum_{l=1}^{L} \omega_l * Q_l + \eta \mathcal{H}(\mathbf{\Omega}) + \lambda\Big(\sum_{l} \omega - 1\Big) \tag{12}$$

where $\lambda$ is the Lagrange multiplier. Moreover, $w_l$ satisfy the **KKT** conditions.

$$\frac{\mathcal{L}(\mathbf{\Omega})}{\omega_l} = 0 \tag{13}$$

then:

$$Q_l - \eta - \eta log(\omega_l) + \lambda = 0$$

The $w_l$ is then equal to:

$$\omega_l = exp^{\frac{Q_l - \eta + \lambda}{\eta}} \tag{14}$$

From the constraint of normalization, we get

$$\sum_{l} \omega_l - 1 = 0$$

$$\lambda = -\eta log(\Big(\sum_{l} exp^{\frac{Q_l - \eta}{\eta}}\Big)$$

Therefore, the analytical solution of $\omega_l$ is written as:

$$\omega_l = \frac{exp^{\frac{Q_l}{\eta}}}{\sum_{l} exp^{\frac{Q_l}{\eta}}} \tag{15}$$

■

The analytical solution of the weights is known as the Gibbs distribution. When $\eta \to \infty$, the

weights tend to be uniformly distributed among all the layers, while $\eta \to 0$, the sparse linear programming solution of weights is recovered. An iterative optimization process occurs between the weights and the partitions until no further changes occur or the maximum number of iterations is reached, as mentioned in algorithm 1. This algorithm results in a locally optimal solution. To enhance the solution, we propose direct method that optimize both partition and weights at the same time.

**Algorithm 1** Self Weighed Modularity Maximization - Iterative Model

**Input** $\mathcal{G} = \{G_1, ...G_L\}$ multiplex graph with $L$ layers, $\eta$

Initialize the weights uniformly
**while** *no change in modularity or max iteration* **do**
    Optimize the partition of multiplex graph using Louvain algorithm with fixed weights
    Update the weights using equation 11
**end**

**Output**: Clusters from double iteration Self Weighted Multiplex Modularity Maximization

#### 3.3.2. Direct model

For every partition, the weights are computed by 11, which shows that the weights depend on the partition. Therefore, the direct method replaces the weights with their partition-dependent values, as follows:

$$\max_{\mathbf{\Delta}} \sum_{l=1}^{L} \frac{exp^{\frac{Q_l}{\eta}}}{\sum_{l'=1}^{L} exp^{\frac{Q_{l'}}{\eta}}} Q_l \tag{16}$$

In this case, only the set of partition $\mathbf{\Delta}$ needs to be optimized. The same Louvain heuristic is used to maximize modularity on the direct model. However, it is not straightforward to express the gain of modularity as a function of a single-layer graph. Therefore, we consider that $\frac{exp^{\frac{Q_l}{\eta}}}{\sum_{l'=1}^{L} exp^{\frac{Q_{l'}}{\eta}}}$ remains unchanged when a vertex $i$ is moved from its partition $c_i$ to $c$. It means that $i$ is moved from $c_i$ to cluster $c$ that maximizes the multiplex modularity. Then, $\frac{exp^{\frac{Q_l}{\eta}}}{\sum_{l'=1}^{L} exp^{\frac{Q_{l'}}{\eta}}}$ is updated by the gain from the new partition. This process is repeated until no improvement of the multiplex modularity is achieved. The partitions are then joined into super-vertices by merging each community into one entity. Self-loop edges represent the intra-community density. The whole process is repeated until there is no further improvement in modularity, as shown in algorithm 2.

### 3.4. Insensitivity to incomplete data

The missing data is considered as individuals who are missing a certain feature value. We model these individuals as vertices without edges within the layers where their features are absent. The property of modularity, as outlined in Brandes et al. (2008), preserves

**Algorithm 2** Self Weighed Modularity Maximization - Direct Model

**Input** $\mathcal{G} = \{G_1, ...G_L\}$ multiplex graph with $L$ layers, $\eta$ **while** *no change in modularity or max iteration* **do**

  **while** *no change in modularity or max iteration* **do**

    Fix the fraction that represent the weights 11

    Compute the gain of modularity of moving a vertex $i$ to its neighbors 10

    Select the cluster that maximizes the multiplex modularity

    Update weights according to the gain obtained previously 11

  **end**

  The clusters are joined in super vertex

**end**

**Output**: clusters from one iteration Self Weighted Multiplex Modularity Maximization

the insensitivity towards isolated vertices. Consequently, these individuals will be clustered according to the layers in which they are available without adding bias to the modularity. This intrinsic property allows to perform the clustering of missing data without any changing in the data. We present its performance in Section 4.

### 3.5. Set of $\eta$

The parameter $\eta$ is used to balance the weights distribution and the multiplex modularity, aiming to find the optimal value that maximizes the multiplex modularity. According to Brandes et al. (2008), the modularity of a graph is limited to the range $[-\frac{1}{2}, 1]$. Furthermore, the maximum value of $H(\omega)$ for $L$ layers is attained when the distribution is uniform and equals $log(L)$. Since the weights distribution exhibits variations between the Dirac and uniform distributions, an exploration of the parameter $\eta$ is conducted within the range of $]0, \frac{k}{log(L)}[$ where $k \geq 0$. In this study, we set $k = 10$. Notably, if $\eta$ exceeds $\frac{k}{log(L)}$, the resulting clustering remains unchanged, yielding the same clustering as that obtained from a uniform distribution.

## 4. Experiments

In this paper, we evaluate the performance of the SWMM on several synthetic and real data. Our primary focus is to assess the robustness of SWMM in the presence of noisy layers and missing data. To do so, we compare the Direct Method (SWMM-DM) and Iterative Method (SWMM-IM) with Graph based Multi-view Clustering (GMC) Wang et al. (2020), Graph Fusion Spectral Clustering (GFSC) Kang et al. (2019) and Multi-view Clustering with Graph Learning (MCGL) Zhan et al. (2017). All the algorithms under comparison are based on multiplex graph representation, where the clustering process is performed.

SWMM inherits from the modularity its capability of automatic identification of cluster's number and its scalability for huge datasets. Throughout the experiments, SWMM finds the correct number of clusters automatically. Furthermore, SWMM derives from faithfully representing non-convex manifold structure via graphs. It also presents a robust solution for handling data from high-dimensional spaces, where individual features or feature types are presented within distinct layers. In our experiments, we employ Normalized Mutual

Information (NMI) McDaid et al. (2011) as a comparison metric to assess the performance of various algorithms.

### 4.1. Toy experiment

We construct toy data to evaluate the robustness of SWMM with respect to noisy communities, missing vertices, and to show the benefit of weights to the multiplex modularity.

#### 4.1.1. Robustness to noise

Our initial experiment involved creating multiple sets of 10 views, each with 100 vertices. Each view was a four-block diagonal matrix, with each block corresponding to individuals in the same cluster. The edges outside the blocks represent the noise $e$ between the clusters, which is defined as the fraction between the Mean Weights Between the different Clusters (MWBC) and Mean Weights Inside the Cluster (MWIC) *.i.e* $e = \frac{MWBC}{MWIC}$. Figure 1 shows three views with different noise level. Each set of 10 views was generated with a specific noise level. All the edges are generated randomly. Figure 2 shows the performance of all algorithms with respect to noise $e$. We notice that up to a high noise level $e = 0.9$, SWMM-DM and SWMM-IM performances is maintained. In contrast, others cannot recover the correct clustering.

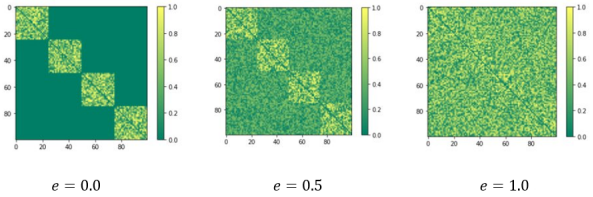



Figure 1: At the left, a layer without noise between the blocks: $e = 0.0$. In the middle, a layer with $e = 0.5$. At the right, a layer with $e = 1.0$ that can be seen as a random graph.

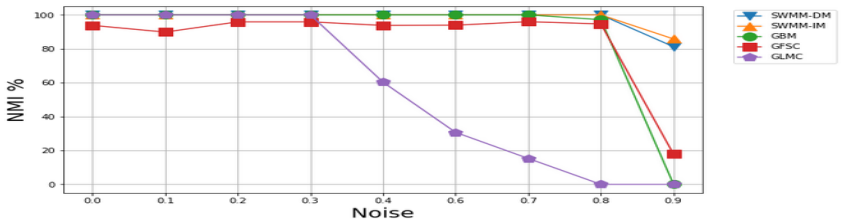



Figure 2: Performance on NMI of SWMM-IM, SWMM-DM, GMC, GFSC, and MCGL on sets on 10 views with different noise levels $e$.

#### 4.1.2. Incomplete representation

To evaluate the robustness with respect to incomplete representation, which can be interpreted as isolated vertices, we conduct the previous experimentation where individuals are

randomly omitted. Let $\gamma = \frac{m}{N*L}$ define the percentage of missing vertices, where $m$ is the number of missing vertices, $N, L$ is the number of vertices of each layer, and the number of views, respectively. The missing data is randomly distributed over each view to simulate various percentages of missing vertices, $\gamma \in [0, 0.9]$. Figure 3 presents views with different values of $\gamma$. Figure 4 illustrates the performance of all methods across different levels of $\gamma$ and $e$. It is clearly that SWMM-DM and SWMM-IM perform significantly better than all other algorithms with even high percentage levels of noise and incomplete data, whereas the other methods fail to compute the clusters after a certain level of incomplete data. Furthermore, for a more comprehensive assessment of the performance of SWMM-DM and SWMM-IM, Figure 5 illustrates their performance with respect to missing vertices with $e = 0.8$. The results indicate that up to a high percentage of missing vertices ($\gamma \leq 0.6$), SWMM-DM and SWMM-IM achieve satisfactory performance with an $NMI \geq 85\%$. In contrast, the other algorithms do not even converge under the same condition.

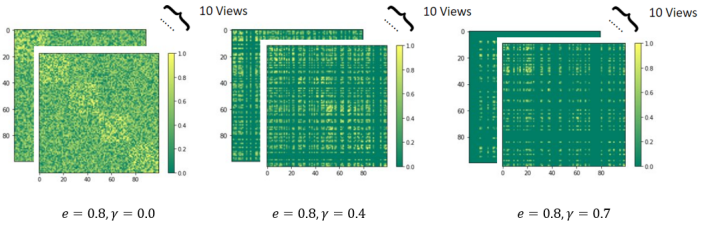
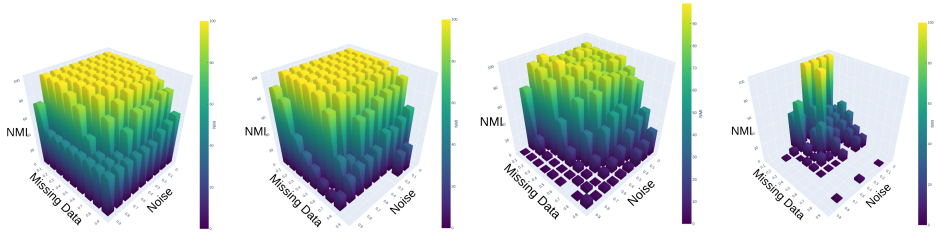



Figure 3: Ten layers with four block diagonals for each one. $e$ represents the noise and $\gamma$ represents the rate of missing data

### 4.1.3. Weights incorporation

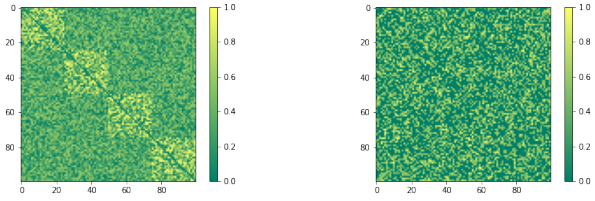
The objective of the third toy experiment is to demonstrate the advantages of incorporating weights in the multiplex modularity. This objective is accomplished by constructing two layers. The first layer is characterized by a noise distribution of $e_1 = 0.7$, while the second layer is a random graph without communities, as illustrated in figure 6. The partition that maximizes the multiplex modularity, resulting in $NMI = 100\%$, successfully recovers the




(a) SWMM-DM (b) SWMM-IM (c) GFSC (d) MCGL

Figure 4: NMI performance regarding noise value and missing data. NMI, $e$, and $\gamma$ is represented on the axis z,y, and x, respectively. Under the same condition, SWMM-ID, SWMM-IM, GFSC, GBM, and MCGL are tested on the same synthetic data.


Figure 5: The NMI score of SWMM-DM and SWMM-IM regarding to $\gamma$, when $e = 0.8$



$(a)$ View 1, $e_1 = 0.7$ $(b)$ View 2, random graph

Figure 6: The first layer has a community structure. However, the second layer is a random graph with the same distribution. It serves to compare SWMM and Uniformally-WMM

clusters from the first layer. It assigns $\omega_1 \simeq 1$ and $\omega_2 \simeq 1e^{-73}$ to the first and second views, respectively. In contrast, the multiplex modularity model that neglects the importance of layers and assigns uniform weights to all layers fails to identify the clusters. The uniformly weighted multiplex modularity yields an NMI less than 80%.

### 4.2. Real datasets

To apply the SWMM algorithm to multi-view datasets, it is essential to represent the views as layers within a multiplex graph. We employed two distinct methods for constructing the multiplex graph layers based on the data's nature. In the case when views are represented as a graph or bipartite graph, such as site-site or site-word views, like in the BBC dataset Greene and Cunningham (2006). We constructed a graph by quantifying the correspondences between vertices sharing common links. However, for views represented by raw multi-dimensional feature data, we constructed the corresponding weighted graph using a graph Gaussian kernel, defined as follows:

$$k(X_i^l, X_j^l) = exp(-||X_i^l - X_j^l||^2/2\sigma^2) \tag{17}$$

where $X_i^l, X_j^l$ are the feature vectors for individuals $i, j$ respectively in view $l$, $\sigma$ is a hyperparameter that defines the kernel width and is typically chosen as the mean distance between individuals. In our experiments, we set $\sigma$ as the mean distance of the given layer, and subsequently selected the ten nearest neighbors to form the graph. The algorithms are assessed on *Multi-feature handwritten dataset* (Mfeat)[1], *BBC dataset*[2], *BBCSport dataset*[3],

1. http://archive.ics.uci.edu/ml/datasets/Multiple+Features
2. http://mlg.ucd.ie/datasets/segment.html
3. http://mlg.ucd.ie/datasets/segment.html

| Data set | individuals | views | clusters |
|---|---|---|---|
| Mfeat | 2,000 | 6 | 10 |
| BBC | 2,225 | 9 | 5 |
| BBCSport | 737 | 9 | 5 |
| 3Sources | 416 | 3 | 6 |
| One-hundred | 1,600 | 3 | 100 |
| ALIO | 11,250 | 4 | 100 |

Table 1: The summary of real datasets

| Method | Mfeat | BBC | BBCSport | 3Sources | One-hundred | ALIO | Average |
|---|---|---|---|---|---|---|---|
| GFSC | 82.48 | 57.49 | 54.88 | 38.50 | 90.14 | 47.20 | 61.78 |
| MCGL | 90.50 | 7.41 | 39.65 | 10.34 | 91.30 | 46.25 | 47.57 |
| GMC | 90.50 | 56.28 | 65.32 | 62.16 | 92.92 | 57.05 | 70.70 |
| SWMM IM | **92.30** | **80.19** | **74.25** | **75.98** | **94.57** | **86.00** | **83.88** |
| SWMM DM | **92.41** | **84.15** | **79.08** | **80.71** | **95.31** | **88.04** | **86.61** |

Table 2: Multiview clustering performance in terms of NMI on real datasets. We set at the bold the first two performed algorithms

*3Sources dataset*[4], *One-hundred plant species leaves dataset* (One-hundred)[5], and *Amsterdam Library of object Images* (ALOI)[6]. The table 1 summarizes the datasets.

The performance evaluation results are presented in table 2, highlighting the notable performance of SWMM-IM and SWMM-DM in multi-view clustering tasks. On average, both models of SWMM outperform graphs-based models for multi-view clustering. In addition, it is worth mentioning that SWMM performs remarkably well on various datasets, including smaller datasets such as BBC and 3Sources, as well as larger datasets like ALIO.

#### 4.2.1. Missing data in real datasets

The performance of all algorithms was assessed in terms of NMI for retrieving clusters under different levels of missing individuals $\gamma$. Due to space limitations, we present the results specifically for the BBCSport dataset in figure 7. We can notice the performance of SWMM-DM and SWMM-IM compared to all other algorithms, demonstrating their ability to persistently recover clusters even in the presence of a high level of missing data $\gamma = 0.5$. This robustness to missing data is crucial for real-world applications, where incomplete information is a common challenge.

## 5. Conclusion

In conclusion, the Self Weighted Multiplex Modularity algorithm has been specifically designed to detect pillar communities in multiplex graphs. We have presented two techniques,

4. http://mlg.ucd.ie/datasets/3sources.html
5. https://archive.ics.uci.edu/dataset/241/one+hundred+plant+species+leaves+data+set
6. https://elki-project.github.io/datasets/multi_view


Figure 7: The performance of SWMM-DM, SWMM-IM, GFSC, MCGL and GMC on BBC-Sport dataset regarding to level of missing data.

the iterative and direct methods, for optimizing the partition process. Through the conducted toy and real data experiments, we have successfully demonstrated the effectiveness of SWMM in handling noisy clusters and missing data. The experimental results highlight the suitability of SWMM for analyzing multi-view data, showcasing its superiority over several state-of-the-art algorithms. By leveraging graph-based representations that capture both local and global structures, SWMM emerges as a powerful unsupervised method. The incorporation of weights to indicate the importance of different views enables the identification of layers that significantly contribute to maximizing distinct divisions, thereby enhancing the algorithm's robustness for feature selection.

## References

Nazanin Afsarmanesh and Matteo Magnani. Finding overlapping communities in multiplex networks. *arXiv preprint arXiv:1602.03746*, 2016.

Michele Berlingerio, Michele Coscia, and Fosca Giannotti. Finding and characterizing communities in multidimensional networks. In *2011 International Conference on Advances in Social Networks Analysis and Mining*, pages 490–494, 2011. doi: 10.1109/ASONAM.2011.104.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, oct 2008. doi: 10.1088/1742-5468/2008/10/p10008. URL https://doi.org/10.1088%2F1742-5468%2F2008%2F10%2Fp10008.

Oualid Boutemine and Mohamed Bouguessa. Mining community structures in multidimensional networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(4): 1–36, 2017.

Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008. doi: 10.1109/TKDE.2007.190689.

Guoqing Chao, Shiliang Sun, and Jinbo Bi. A survey on multi-view clustering, 2017. URL https://arxiv.org/abs/1712.06246.

Man-Sheng Chen, Jia-Qi Lin, Xiang-Long Li, Bao-Yu Liu, Chang-Dong Wang, Dong Huang, and Jian-Huang Lai. Representation learning in multi-view clustering: A literature review. *Data Science and Engineering*, 7:1–17, 08 2022. doi: 10.1007/s41019-022-00190-8.

Deepjyoti Choudhury and Arnab Paul. Community detection in social networks: an overview. *International Journal of Research in Engineering and Technology*, 2(2):6–13, 2013.

Manlio De Domenico, Andrea Lancichinetti, Alex Arenas, and Martin Rosvall. Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems. *Physical Review X*, 5(1):011027, 2015.

Nicolas Dugué and Anthony Perez. *Directed Louvain: maximizing modularity in directed networks*. PhD thesis, Université d'Orléans, 2015.

Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010. ISSN 0370-1573. doi: https://doi.org/10.1016/j.physrep.2009.11.002. URL https://www.sciencedirect.com/science/article/pii/S0370157309002841.

Derek Greene and Pádraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press, 2006.

Roger Guimerà, Marta Sales-Pardo, and Luís Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 70:025101, 09 2004. doi: 10.1103/PhysRevE.70.025101.

Zhao Kang, Guoxin Shi, Shudong Huang, Wenyu Chen, Xiaorong Pu, Joey Tianyi Zhou, and Zenglin Xu. Multi-graph fusion for multi-view spectral clustering, 2019.

Jungeun Kim, Jae-Gil Lee, and Sungsu Lim. Differential flattening: A novel framework for community detection in multi-layer graphs. *ACM Trans. Intell. Syst. Technol.*, 8(2), oct 2016. ISSN 2157-6904. doi: 10.1145/2898362. URL https://doi.org/10.1145/2898362.

Zhaoming Kong, Lichao Sun, Hao Peng, Liang Zhan, Yong Chen, and Lifang He. Multiplex graph networks for multimodal brain network analysis, 2021. URL https://arxiv.org/abs/2108.00158.

Matteo Magnani, Obaida Hanteer, Roberto Interdonato, Luca Rossi, and Andrea Tagarelli. Community detection in multiplex networks, 2019. URL https://arxiv.org/abs/1910.07646.

Aaron F McDaid, Derek Greene, and Neil Hurley. Normalized mutual information to evaluate overlapping community finding algorithms. *arXiv preprint arXiv:1110.2515*, 2011.

Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and Jukka-Pekka Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010. doi: 10.1126/science.1184819. URL https://www.science.org/doi/abs/10.1126/science.1184819.

M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006. doi: 10.1073/pnas.0601602103. URL https://www.pnas.org/doi/abs/10.1073/pnas.0601602103.

Mark EJ Newman. Spectral methods for community detection and graph partitioning. *Physical Review E*, 88(4):042822, 2013.

Feiping Nie, Jing Li, and Xuelong Li. Self-weighted multiview clustering with multiple graphs. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2564–2570, 2017. doi: 10.24963/ijcai.2017/357. URL https://doi.org/10.24963/ijcai.2017/357.

T.Aste N.Musmeci, V.Nicosia. 02 2017.

Evangelos E Papalexakis, Leman Akoglu, and Dino Ience. Do more views of a graph help? community detection and clustering in multi-graphs. In *Proceedings of the 16th International Conference on Information Fusion*, pages 899–905. IEEE, 2013.

Moumita Saha. A graph based approach to multiview clustering. In *Pattern Recognition and Machine Intelligence: 5th International Conference, PReMI 2013, Kolkata, India, December 10-14, 2013. Proceedings 5*, pages 128–133. Springer, 2013.

Andrea Tagarelli, Alessia Amelio, and Francesco Gullo. Ensemble-based community detection in multilayer networks. *Data Mining and Knowledge Discovery*, 31:1506–1543, 2017.

Leo Torres, Ann S. Blevins, Danielle S. Bassett, and Tina Eliassi-Rad. The why, how, and when of representations for complex systems, 2020. URL https://arxiv.org/abs/2006.02870.

Hao Wang, Yan Yang, and Bing Liu. Gmc: Graph-based multi-view clustering. *IEEE Transactions on Knowledge and Data Engineering*, 32(6):1116–1129, 2020. doi: 10.1109/TKDE.2019.2903810.

Rongkai Xia, Yan Pan, Lei Du, and Jian Yin. Robust multi-view spectral clustering via low-rank and sparse decomposition. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, page 2149–2155. AAAI Press, 2014.

Kun Zhan, Changqing Zhang, Junpeng Guan, and Junsheng Wang. Graph learning for multiview clustering. *IEEE transactions on cybernetics*, 48(10):2887–2895, 2017.

Kun Zhan, Chaoxi Niu, Changlu Chen, Feiping Nie, Changqing Zhang, and Yi Yang. Graph structure fusion for multiview clustering. *IEEE Transactions on Knowledge and Data Engineering*, 31(10):1984–1993, 2018.

Katharina A. Zweig. *Network Representations of Complex Systems*, pages 109–148. Springer Vienna, Vienna, 2016. ISBN 978-3-7091-0741-6. doi: 10.1007/978-3-7091-0741-6_5. URL https://doi.org/10.1007/978-3-7091-0741-6_5.