# Multi-layer Graph Clustering

Max Kramkimel*
Meilame Tayebjee
max.kramkimel@gmail.com
meilame.tayebjee@hec.edu

## ABSTRACT

This papers aims at summarizing and finding the limits of the article "Clustering with Multi-Layer Graphs: A Spectral Perspective" [Dong et al. 2012], published by Xiaowen Dong, Pascal Frossard, Pierre Vandergheynst and Nikolai Nefedov. It deals with graph clustering : nowadays, a lot of data we find is multimodal : it represents the same phenomenon with different metrics or in different setups. This can be seen as a multi-layer graph : we have multiple graphs with the same vertices, but the edges are not the same. [Dong et al. 2012] develop two new methods to extract information from the graphs and cluster the vertices. Both methods use the Laplacian graph matrix and its eigenvalues to extract and combine information from the different layers. The authors of [Dong et al. 2012] implement their algorithms on three real-world datasets. They show either better or competitive performance compared to the known algorithms for multi-layer clustering. We have implemented these two new methods on our own, and made experiments on synthetic datasets. Thanks to these experiments, we show the main limitations of the algorithms developed by [Dong et al. 2012].

## KEYWORDS

multi-layer graph, clustering, spectrum of graph, Laplacian matrix, implementation

## 1 INTRODUCTION

### 1.1 Clustering on multi-layer graphs

Clustering with multi-layers graphs is a quite well-known subject. It studies a set of vertices, with pairwise relationships between them (that can be weighted or not). However, instead of having one set of relationships, we have multiple sets of these relationships. The goal is to extract information (such as clustering), knowing the fact that we have more information than with only one set of relationships. This problem has been studied extensively in the literature, such as in [Schaeffer 2007].

Moreover, this problem is particularly relevant with the development of new technologies. Social networks can be seen as graphs, where users are the nodes. Different proxies can be used to define the relationships between them : friendship, commenting, liking... [Dong et al. 2012] takes as example the MIT Reality Mining Project [Eagle and Sandy], which is a set of mobile phone data. A part of this data set is shown figure 1. The vertices are members of the MIT. In the different graphs, an edge between them means respectively that they have been closed during a given Saturday night, that they have used the same tower to make a call, and that they called each other.

"Clustering with Multi-Layer Graphs: A Spectral Perspective" [Dong et al. 2012] presents two new methods to perform spectral clustering. These methods are inspired from an already existing method on a single graph : the normalized spectral regularization [Shi and Malik 2000]. It takes as a starting point the Laplacian, and find its biggest eigenvalues to perform clustering. The two methods developed by [Dong et al. 2012] will also use the Laplacian matrices from the different layers of the graphs, and perform smart transformations on it, in order to gather as much information as possible.

### 1.2 Our contributions

In the section 2 of this document, we remind the methods used in [Dong et al. 2012] that will serve as a basis for our work, mainly the Normalized Spectral Clustering (which was already existing), the SC-GED and the SC-SR algorithms (these two last algorithms are the ones presented in [Dong et al. 2012]). **Additionally**, we provide a clear pseudo-code for the induced SC-SR algorithm with M layers, while the authors only gave it for 2 layers and provided a mere intuition for the M-layer case. We also propose an accelerated version that yields almost the same results - at least on our experiments (see section 4). We also discuss some sibylline issues such as the choice of the first layer for the SC-SR algorithm.

Section 3 will present their experiments and results. We will leverage this section to introduce the metrics used as well as baselines algorithms, which we will also use, and try to provide our own analysis of the results.

In section 4, we will eventually present our own experiments, based on synthetic data. We try to show in which typical settings SC-GED and SC-SR fare better than baselines, and conversely when do they reach their limitations in limit and asymptotic cases.

We re-implemented each algorithm and all of our experiments are fully reproducible and can be found here: https://github.com/meilame-tayebjee/Multilayer-Graph-Clustering.

## 2 NEW METHODS TO PERFORM MULTI-LAYER GRAPH CLUSTERING

### 2.1 Setting

We first need to give the mathematical notations of our problem, along with some definitions. We will deal with a non-directed graph $\mathcal{G}$ with $M$ layers, called $\mathcal{G}^{(i)}$. Each one of them takes the form $\mathcal{G}^{(i)} = \{V, E^{(i)}, w^{(i)}\}$, where $V$ are the vertices (common to all layers), $E^{(i)}$ the set of edges and $w^{(i)}$ their weights.

The adjacency matrices will be called $W^{(i)}$ in this document. They are symmetric matrices made of 0 and 1. $W^{(i)}_{i,j}$ is equal to 1 if

*Both authors contributed equally to this project.

Authors' address: Max Kramkimel, max.kramkimel@gmail.com; Meilame Tayebjee, meilame.tayebjee@hec.edu.

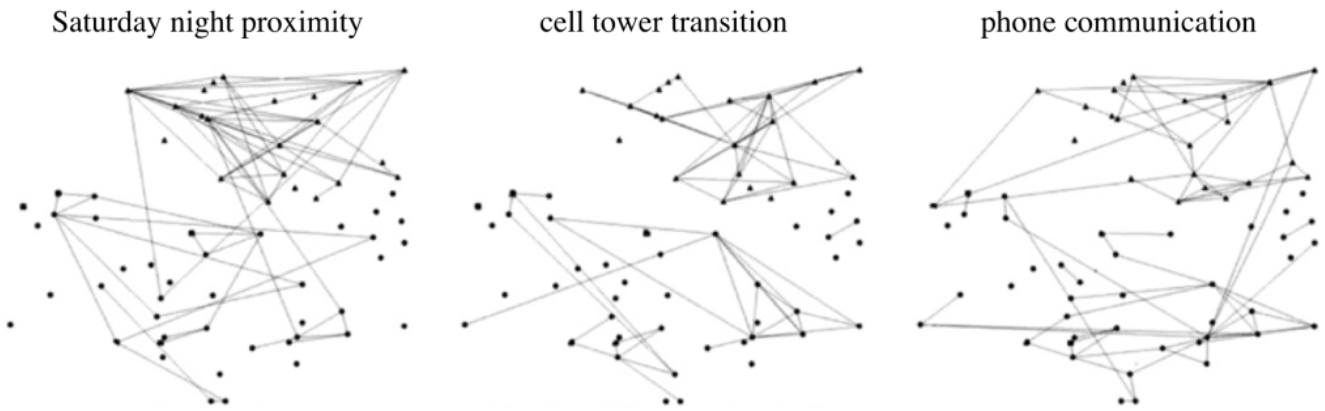


**Figure 1: Extracted from [Dong et al. 2012]. Relationships between MIT members according to their proximity on a Saturday night, the tower they used to make a call this night, and if they called each other.**

the vertices $i$ and $j$ are linked, 0 otherwise. They are quite useful to represent the structure of a graph. The MIT dataset (figure 1) is an example of such a graph. Its adjacency matrices are represented in figure 2.

The degree matrices (we will denote them $D^{(i)}$ in this paper) are diagonal matrices were $D^{(i)}_{j,j}$ is equal to the degree of the vertex $j$, that is to say the number of vertices connected to $j$. Finally the Laplacian matrix is $L = D - W$.

## 2.2 Normalized spectral regularization

To present the algorithms developed by [Dong et al. 2012], we first need to explain the Normalized Spectral Regularization [Shi and Malik 2000], which will be used as a basis for our intuitions. This algorithm is made for single-layer graphs. It takes as a starting point the normalized Laplacian matrix $L_{sym} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}$ or $L_{rw} = D^{-1}(D - W)$ depending on the context. The algorithm then performs the eigendecomposition of the normalized Laplacian matrix, and selects a limited number of eigenvectors. It then uses them to embed the vertices in a low dimensional space and perform a clustering algorithm. In this document we will use the K-means algorithm. The pseudocode of this algorithm is given in algorithm 1, as stated in [Dong et al. 2012].

The key point of this algorithm is to reduce the space dimension, and transform each vertex into a low dimensional vector. In the next two subsections, we will present the new algorithms from [Dong et al. 2012] : "clustering with generalized-eigen decomposition" (SC-GED) and "clustering with spectral regularization" (SC-SR). They both compute in a smart way a joint spectrum for all the layers.

## 2.3 Clustering with generalized eigen-decomposition (SC-GED)

The normalized spectral decomposition computes the eigenvectors of $L_{rw}$, that is to say it finds the matrix of eigenvectors $P$ and the matrix of eigenvalues $\Delta$ such tat $L_{rw} = P\Delta P^{-1}$. When we have $M$ layers, we do not have any guarantee that the $L^{(i)}_{rw}$ will have common eigenvectors (and even less guarantees that all their eigenvectors are going to be shared). Thus, the idea is to find a matrix

**Algorithm 1** Normalized Spectral Clustering ([Shi and Malik 2000])

1: **Input:**
$W$: The $n \times n$ weighted adjacency matrix of graph $\mathcal{G}$ with $n$ vertices
$k$: Target number of clusters
2: Compute the degree matrix $D$.
3: Compute the random walk graph Laplacian $L_{\text{rw}} = D^{-1}(D-W)$.
4: Compute the first $k$ eigenvectors $u_1, \ldots, u_k$ (which correspond to the $k$ smallest eigenvalues) of the eigenvalue problem $L_{\text{rw}}u = \lambda u$.
5: Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing $u_1, \ldots, u_k$ as columns.
6: Let $y_i \in \mathbb{R}^k$ $(i = 1, \ldots, n)$ be the $i$-th row of $U$ to represent the $i$-th vertex in the graph.
7: Cluster $y_i$ in $\mathbb{R}^k$ into $C_1, \ldots, C_k$ using the K-means algorithm.
8: **Output:**
$C_1, \ldots, C_k$: The cluster assignment

$P$ such that $L^{(i)}_{rw} \approx P\Delta^{(i)}P^{-1}$ for all $i$. To simplify the computation, we can even find $P$ and $Q$ such that $L^{(i)}_{rw} \approx P\Delta^{(i)}Q$ and $Q \approx P^{-1}$ for every $i$. To do this, SC-GED optimizes the following program :

$$\begin{aligned} \underset{P,Q \in \mathbb{R}^{n \times n}}{\arg\min}\ S = & \frac{1}{2}\sum_{i=1}^{M} ||L^{(i)}_{\text{rw}} - P\Delta^{(i)}Q||^2_F \\ & + \frac{\alpha}{2}(||P||^2_F + ||Q||^2_F) + \frac{\beta}{2}||PQ - I_n||^2_F \end{aligned} \tag{1}$$

$||\cdot||_F$ is the frobenius norm, and $I_n$ the identity matrix of dimension $n$. The first term represents the error of approximation between $L^{(i)}_{rw}$ and $P\Delta^{(i)}Q$. The second one is implemented to guarantee some numerical stability. The third term characterizes the approximation $Q \approx P^{-1}$.

This problem is not convex in $(P, Q)$, so the optimization is done one variable at a time, and this process is repeated. However, this sections aims at being a summary, so we will not extend on the details. Once the optimization is done, we can perform dimension

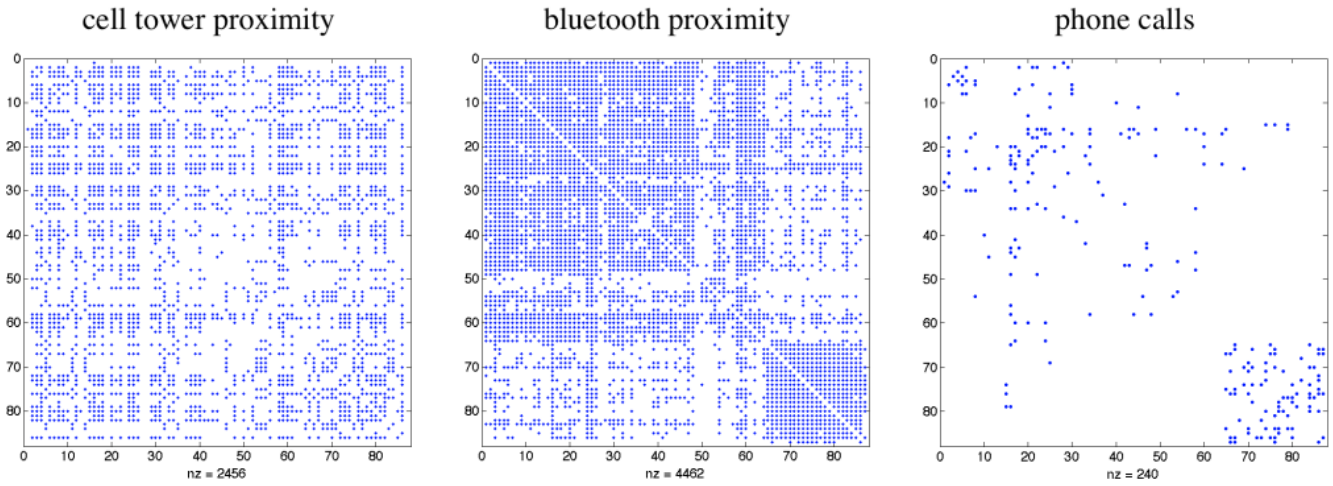


**Figure 2: Extracted from [Dong et al. 2012]. Adjacency matrices corresponding to graphs in figure 1**

reduction and clustering, as it is done in normalized spectral regularization.

The pseudocode is given in algorithm 2, like it is stated in [Dong et al. 2012].

**Algorithm 2** Clustering with generalized eigen-decomposition (**SC-GED**) [Dong et al. 2012]

**Input:**
$W^{(i)}$ $(i = 1, \ldots, M)$: $M$ $n \times n$ weighted adjacency matrices of a $M$-layer graph $\mathcal{G}$ with $n$ vertices
$k$: Target number of clusters
$\alpha, \beta$: hyperparameters
For each $i$, compute the degree matrix $D^{(i)}$.
For each $i$, compute the random walk graph Laplacian $L_{\text{rw}}^{(i)} = D^{(i)\,(-1)}(D^{(i)} - W^{(i)})$.
Solve the optimization problem in Eq. (1) to get the joint eigenvector matrix $P$.
Let $U' \in \mathbb{R}^{n \times k}$ be the matrix containing the first $k$ columns of $P$.
Let $y_i \in \mathbb{R}^k$ $(i = 1, \ldots, n)$ be the $i$-th row of $U'$ to represent the $i$-th vertex in the graph.
Cluster $y_i$ in $\mathbb{R}^k$ into $C_1, \ldots, C_k$ using the K-means algorithm.
**Output:**
$C_1, \ldots, C_k$: The cluster assignment

The authors [Dong et al. 2012] recommend using $\alpha = 10$ and $\beta = 100$, or at least $\alpha << \beta$. A cross-selection in a particular case is available in the code, yet in section 4, we will mainly stick with these recommended values.

## 2.4 Clustering with spectral regularization (SC-SR) [Dong et al. 2012]

*2.4.1 Generalities.* The authors detail this algorithm for $M = 2$ - we will give the generalized version in algorithm 4. The main goal is to compute a joint spectrum over the 2 layers. First, we consider the $k$ first eigenvalues of $\mathcal{G}^{(1)}$, called $u_1, ..., u_k$. These eigenvalues have an interesting property : they can be used as mapping function into a sub-dimensional dimensional space, while having a property of smoothness : the points that were close before the mapping remain close. However, we do not have any guarantee that the eigenvalues from $\mathcal{G}^{(1)}$ will be smooth mapping function on $\mathcal{G}^{(2)}$ (the eigenvalues from $\mathcal{G}^{(2)}$ would). The idea is to find new vectors as mapping functions that would be close from the eigenvalues of $\mathcal{G}^{(1)}$, but would have smoothness properties on $\mathcal{G}^{(2)}$ ; basically, a new mapping that will *minimize the disagreement* between $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$. More formally speaking, we minimize the following function :

$$\arg\min_{f_i \in \mathbb{R}^n} \left\{ \frac{1}{2} ||f_i - u_i||_2^2 + \lambda \cdot f_i^T L_{sym}^{(2)} f_i \right\} \quad \text{for} \quad i = 2, \ldots, k$$

where the $f_i$ are the vectors corresponding to our new mapping function. The trade-off between closeness to the eigenvectors of $\mathcal{G}^{(1)}$ and smoothness on $\mathcal{G}^{(2)}$ clearly appears ; $\lambda$ is used to choose the trade-off between the two terms and has to be fine-tuned (see 2.4.2, 4.2.1). This minimization problem can be derived in a closed form [Zhou and Schölkopf 2004]:

$$f_i^* = \frac{1}{\lambda}(L_{\text{sym}} + \frac{1}{\lambda} I)^{-1} u_i \tag{2}$$

We highlight the fact that in this algorithm $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$ do not play a symmetrical role. [Dong et al. 2012] states that "the most informative layer" should be chosen as $\mathcal{G}^{(1)}$, yet do not give any details on the definition of informativeness of a layer. **Our interpretation** of the informativeness of a layer is the NMI score (compared with the ground truth clustering) (see 3.1) obtained when a Normalized Spectral Clustering (1) is performed on it. This implies that the ground truth is used within the algorithm, which is indeed a strong assumption. To clarify, we designed another slightly modified version of SC-SR - **randomSC-SR** - that follows exactly the same pseudo-code as classic SC-SR (3), but chooses randomly the first layer. We will not detail this here for the sake of clarity, but we ran all of our experiments with randomSC-SR to see the impact

of the choice of the first layer. What we can conclude is that this choice is not affecting drastically the results within the framework of our simple experiences. Everything is available in the code.

The M-layer version pseudo-code, inferred from what is said in the article [Dong et al. 2012], is given in 3. Basically, we integrate one layer after each other, update the $(f_i)_i$ using

$$\forall i = 1...k,\ f_i^{new} = \frac{1}{\lambda_{new}}(L_{\text{sym}}^{new} + \frac{1}{\lambda_{new}}I)^{-1} f_i \tag{3}$$

The *new* layer integrated is the one maximizing the NMI (3.1) between - for all the layers $m$ still not integrated:

- the K-Means clustering obtained from the current vectors $(f_i)_i$
- the Normalized Spectral Clustering using layer $m$

We propose a simplified version, called ***ourSC-SR***, where we try to limit the number of calls to K-Means Clustering. We compute once and for all all the Normalized Spectral Clustering (NSC), and the order of integration is directly given by the sorting of the layers using the NMI between the corresponding NSC and the ground truth clustering. **We show empirically that this version is ≈ 1.5 times faster and yields very similar results - at least on the experiments conducted in 4**. Morally, the classic SC-SR integrates the *closest* layer (in terms of informativeness), given the fact that we started from the most informative one at first and got forward integrating the closest layer in a greedy way ; ourSC-SR integrates the best layer among those not integrated, which is very likely to also be the closest...

*2.4.2 Discussion on the hyperparameter $\lambda$.* According to the authors, this parameter $\lambda$ "should loosely reflect the mutual information shared by the two layers being considered" ([Dong et al. 2012]). **Our interpretation** is that $\lambda$ should be high if the two layers are - in a way - similar, in order to have a real joint smooth spectrum (4.2.1 will show that this is not true every time). However, when they are different, we want to trust more the first one (which is supposed to be the most informative), and keep the $f_i$ close to the initial eigenvectors.

In the M-layer case, we can change this parameter for each layer integrated, hence the input being a *sequence* $(\lambda_i)$ for $i = 2...M$. For ourSC-SR, according to the discussion above, **as the layers are ranked from the most informative to the least one**, the sequence should be *decreasing* or at least *non-increasing*. It is also the strategy used by the authors ([Dong et al. 2012]) for the MIT dataset ($\lambda_1 = 2, \lambda_2 = 1$) for the classic SC-SR.

**Algorithm 3** Multi-layer clustering with spectral regularization (**SC-SR**), inferred from [Dong et al. 2012]

**Input:**
$W^{(i)}$ $(i = 1...M)$: $n \times n$ weighted adjacency matrices of graph layers $\mathcal{G}^{(1)}...\mathcal{G}^{(M)}$
$k$: Target number of clusters
$(\lambda_i)_{i=2...M}$ : Trade-off parameters
**for** $i \leftarrow 1$ to $M$ **do**
  Compute Normalized Spectral Clustering over $\mathcal{G}^{(i)}$ (denoted as $clustering_i$ and associated NMI with ground truth clustering
**end for**
Let $\mathcal{G}^{(m_1)}$ being the layer maximizing the NMI with the ground truth
Initialize $f_i \leftarrow u_i$ for $i = 1...k$, $u_i$ being the eigenvectors of $\mathcal{G}^{(m_1)}$
non-integrated-layers $\leftarrow [1...M]/\{m_1\}$
current-clustering $\leftarrow$ K-Means$((f_i)_i)$
**while** non-integrated-layers is not empty **do**
  Search for layer $m$ to integrate : the one that maximizes NMI(current-clustering, $clustering_m$)
  Compute $L_{\text{sym}}^{(m)}$, normalized Laplacian matrix of $\mathcal{G}^{(m)}$
  $f_i \leftarrow \frac{1}{\lambda_i}(L_{\text{sym}}^{(i)} + \frac{1}{\lambda_i}I)^{-1} f_i$
  current-clustering $\leftarrow$ K-Means$((f_i)_i)$
  non-integrated-layers.remove($m$)
**end while**
Let F be the matrix in $\mathbb{R}^{n,k}$ whose columns are $(f_i)_{i=1...k}$
Let $y_i \in \mathbb{R}^k$ $(i = 1, \ldots, n)$ be the $i$-th row of $F$ to represent the $i$-th vertex in the graph.
Cluster $y_i$ in $\mathbb{R}^k$ into $C_1, \ldots, C_k$ using the K-means algorithm.
**Output:**
$C_1, \ldots, C_k$: Cluster assignment

## 3 RESULTS FROM [Dong et al. 2012]

### 3.1 Setting

The article uses 3 real-life datasets to experiment its algorithms. For the sake of concision, we will only tackle the first one.

The first dataset is the MIT Dataset [Eagle and Sandy]. 87 users have been studied, and the authors took three interesting parameters to create graphs : location on a Saturday evening, use of a given cell tower, call between users. The goal is to create six clusters, taking as a reference the administration to which people told they belong. We recall that the graphs and their adjacency matrices are shown in figures 1 and 2.

The authors compare their algorithms to 5 known algorithms of multi-layer clustering :

- SC-SUM : it corresponds to normalized spectral clustering applied to the sum of the adjacency matrix $\sum_{i=1}^{M} W^{(i)}$ instead of $W$
- Kernel-KMeans : it corresponds to the kernel K-means applied on $\sum_{i=1}^{d} u_k^{(i)} u_k^{(i),T}$, d being an hyperparameter
- SC-AL : it corresponds to the normalized spectral clustering applied on the average Laplacian matrix $\frac{1}{M}\sum_{i=1}^{M} L_{rw}^{(i)}$
- Co-Regularization (CoR) [Kumar et al. 2010], which is out of scope for this summary.
- Community detection via modularity maximization (CD) [Nefedov 2011], also out of scope.

In section 4, in addition to SC-GED and SC-SR, we re-implement SC-SUM and SC-AL ; CoR has been imported from [Perry et al.

**Algorithm 4** Simplified Multi-layer clustering with spectral regularization (**ourSC-SR**)

**Input:**
$W^{(i)}$ $(i = 1...M)$: $n \times n$ weighted adjacency matrices of graph layers $\mathcal{G}^{(1)}...\mathcal{G}^{(M)}$
$k$: Target number of clusters
$(\lambda_i)_{i=2...M}$ : Trade-off parameters
**for** $i \leftarrow 1$ to $M$ **do**
  Compute Normalized Spectral Clustering over $\mathcal{G}^{(i)}$ and associated NMI with ground truth clustering
**end for**
Rank the layers based on their informativeness
Initialize $f_i \leftarrow u_i$ for $i = 1...k$, $u_i$ being the eigenvectors of $\mathcal{G}^{(1)}$
**for** $i \leftarrow 2$ to $M$ **do**
  Compute $L^{(i)}_{\text{sym}}$, normalized Laplacian matrix of $\mathcal{G}^{(i)}$
  $f_i \leftarrow \frac{1}{\lambda_i}(L^{(i)}_{\text{sym}} + \frac{1}{\lambda_i} I)^{-1} f_i$
**end for**
Let F be the matrix in $\mathbb{R}^{n,k}$ whose columns are $(f_i)_{i=1...k}$
Let $y_i \in \mathbb{R}^k$ $(i = 1, \ldots, n)$ be the $i$-th row of $F$ to represent the $i$-th vertex in the graph.
Cluster $y_i$ in $\mathbb{R}^k$ into $C_1, \ldots, C_k$ using the K-means algorithm.
**Output:**
$C_1, \ldots, C_k$: Cluster assignment

2021].

In order to be able to compare their results, the authors use three different metrics. We call $\Omega = \{\omega_1, ...., \omega_k\}$ the result clusters, $C = \{c_1, ..., c_k\}$ the true clusters and $N$ the number of elements. The metrics used are :

- Purity :
$$Purity(\Omega, C) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$
- NMI :
$$NMI(\Omega, C) = \frac{I(\Omega; C)}{[H(\Omega) + H(C)]/2}$$
where $I$ denotes the mutual information and $H$ the entropy.
- RI :
$$RI(\Omega, C) = \frac{TP + TN}{TP + FP + FN + TN}$$
where TP, TN, FP and FN mean respectively : True Positive, True Negative, False Positive, False Negative.

We highlight the fact that all of these metrics need to have access to the *ground truth* for the cluster assignment - for the datasets used by the authors, they have access to ground truth through different manners (self-reported affiliations of the subjects for the MIT dataset for instance). In section 4, our experiments will also abide by this rule.

## 3.2 Results

The results on the MIT dataset of all the algorithms for the different metrics are given in figure 3. For each metric, the best two results are in bold. It reveals that SC-SR achieves competitive performance compared to the best algorithm, CoR. Neither of them is always the best algorithm, and their performance look quite close. Moreover, SC-GED seems to be outperformed by the other algorithms. The authors claim that it is quite logical : SC-GED is doing a kind of average on the layers, while SC-SR is more taking into account the different particularities.

Moreover, the authors underline that SC-SR has a better complexity than CoR.

# 4 OUR EXPERIMENTS

## 4.1 Synthetic data generation & notations

We decide to apply these algorithms to synthetically generated data so as to have a complete control over different parameters. We voluntarily generated simple datasets, in order to study them extensively and make them understandable. We are aware that they are degrees of freedom that we did not use.

Our datasets were generated according to the following model, whose parameters are:

- $n$, the number of nodes, the nodes being denoted as integers between 1 and $n$
- $k$, the ground truth number of clusters. All the clusters will always be of size $\lfloor \frac{n}{k} \rfloor$; then, without loss of generality, the ground truth clusters are $C_1 = \{1...\lfloor \frac{n}{k} \rfloor\}$, ..., $C_k = \{\lceil \frac{(k-1)n}{k} \rceil ...n\}$
- $M$, the number of layers, $\mathcal{G}^{(m)}$ the $m^{\text{th}}$ layer (or by metonymy its adjacency graph)
- For a layer $m$, $p_m$ is the probability that two nodes of the same cluster are connected by an edge in $\mathcal{G}^{(m)}$. We use three thresholds for $p_m$:
  - 0.8 (referred to as HIGH)
  - 0.5 (referred to as MEDIUM)
  - 0.3 (referred to as LOW)

  For some examples, this probability will depend of the index of the cluster. We will denote it as $p_{m,i}$, with $i$ the index of the cluster.
- For a layer $m$, $\delta_m$ is the *discrimination parameter* also referred to as the *noise* ; $p_m - \delta_m$ is the probability that two nodes that *are not* in the same cluster are nevertheless connected in $\mathcal{G}^{(m)}$. Thus, $\delta_m \in [0, p_m]$. If $\delta_m \approx p_m$, the information conveyed by the layer $m$ is clear (only the same-cluster nodes are connected) ; if in addition $p_m$ is high, then the information becomes loud and clear. Conversely, if $\delta_m \approx 0$, the layer is very *noisy*, and we can not really distinguish between same-cluster and different-cluster nodes.

## 4.2 Experiments and results

*4.2.1 "M times the same layer".* We start with a very simple setting. We generate $M$ layers with the same parameters: for all $m$, $p_m = p$, the probability that two nodes in the same cluster are connected, is set to 0.8 (HIGH threshold), and a given $\delta$ $(< p)$. We have 100 nodes and the ground truth is known ($k = 4$, Cluster 1 = {1...25}, ..., Cluster 4 = {76...100}). Our main goal here is to see what happens / which algorithm fares better when:

| | single graph layer | | | combination of multiple graph layers | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cell Tower | Bluetooth | Phone call | SC-GED | SC-SR (CT + BT) | SC-SR (all 3 layers) | SC-SUM | K-Kmeans | SC-AL | CoR | CD |
| Purity | 0.5402 | 0.7011 | 0.4253 | 0.7011 | 0.7126 | **0.7241** | 0.6897 | 0.6256 | 0.7011 | **0.7241** | 0.6322 |
| NMI | 0.2023 | 0.4891 | 0.1151 | 0.5073 | 0.5221 | **0.5519** | 0.5100 | 0.3867 | **0.5345** | 0.5289 | 0.3985 |
| RI | 0.6902 | 0.7637 | 0.3192 | 0.7477 | 0.7797 | **0.7864** | 0.7618 | 0.7283 | 0.7712 | **0.7872** | 0.7439 |

(a) clustering performance on the MIT dataset

**Figure 3: Extracted from [Dong et al. 2012]. Evaluation of algorithms on 3 datasets for different metrics**

- $M$, the number of layers, increases (from 1 to 10). In [Dong et al. 2012], only 3-layer graphs are studied.
- $\delta$ is varying between $p$ (clear information) and 0 (totally noisy/hidden information)

∗ But we first make a little detour here to optimize the parameter $\lambda$ - or at least see the behavior when it is varying - in the SC-SR algorithm. Here, we chose to keep $\lambda_i$ constant for all the layers, as they share exactly the same information (see 2.4.2).

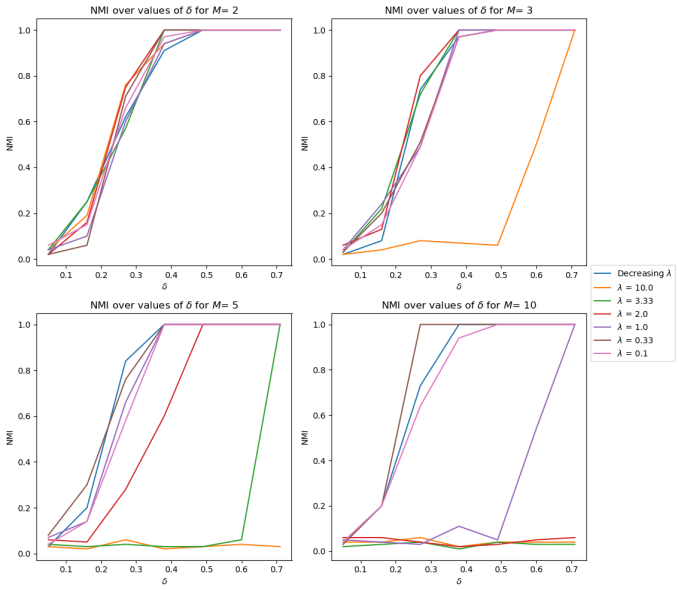


**Figure 4: SC-SR : NMI over $\delta$, for different $M$ and $\lambda$**

Figure 4 shows the results. Unsurprisingly, the NMI grows as the discrimination $\delta$ grows ; moreover, there seems to be a $\delta_{min}$ (slightly before 0.4), where the clustering reaches perfection.

However, if the parameter does not seem to have a huge impact for a small number of layers, for a large $\lambda$, we can see that performances are strongly degrading as the number of layers $M$ grows. Intuitively, when we put too much weight on the smoothing part, we put too much weight on the new layer being integrated, blurring the information gathered until there ; this repeated a larger number of times undermines deeply the model.

∗ Figure 5 displays the results of the experiment. In SC-GED, we set $\alpha = 10, \beta = 100$ (as recommended by the authors), in SC-SR $\lambda = \frac{1}{3}$ (constant) which seems to be the best performer following the discussion above (see Figure 4).

Clearly, SC-SUM and SC-AL outperform the three other algorithms, especially when the number of layers increases. As these algorithms rely on a summation and averaging, they largely benefit from the repetition of "independent and identically distributed" layers. Other algorithms that rely on projections and do not leverage the structure of the problem here, can not catch up. SC-SR, however, is performing a bit better than CoR and SC-GED.

To be noted that there is still a threshold $\delta^{(A)}_{min}$ where a given algorithm A reaches perfection. This threshold seems stable for SC-SR, SC-GED and CoR as M increases ; however, it sharply decreases for SC-SUM and SC-AL: the number of clusters compensate the lack of discrimination. These algorithms are able to (over ?)-fit in this type of problems.

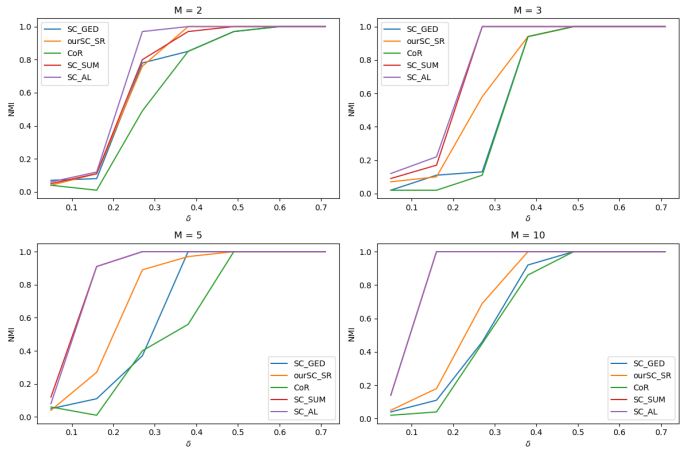


**Figure 5: Benchmark for the "M-times-the-same-layer" test**

*4.2.2 Adding layers of noise.* In this section, we will deal with an example that takes more advantages of the specific features

of SC-SR. We will create graphs of $M$ layers ($k = 5$), with two informative layers (the first and last ones, ($p_1 = HIGH, \delta_1 = 0.4$), ($p_M = LOW, \delta_M = 0.2$)), and $M - 2$ layers of noise between them ($p_i = 0.4, \delta_i = 0$), $i \in \{2, ..., M-1\}$. Figure 6 shows the results of the algorithms for the different metrics when $M$ increases. We will not discuss how we fitted $\lambda$ for the sake of concision.

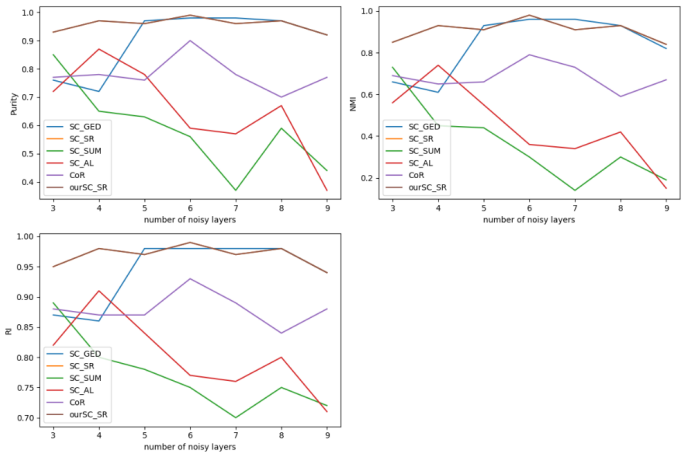


**Figure 6: Purity, NMI and RI when the number of noisy layers increases**

Unsurprisingly, the performance of SC-SUM and SC-AL shrink when the number of noisy layers increases. These algorithms are based on average computation, so they are strongly affected by the noisy layers. Moreover, the probability of connection in the noisy layers is 0.4, which is more than all the probabilities of connection in the last layer. The computation of averages makes impossible for these algorithms to take benefit from the last layer. On the contrary SC-SR keeps a constant performance, which was expected : it performs a projection, and is able to keep the information while going through all the layers. CoR, which was showing similar performance to SC-SR in [Dong et al. 2012] is clearly underperforming on this example.

Finally, what is more surprising is that SC-GED shows very good performance. Indeed, even if if this algorithm is not really based on average computation, it tries to find a kind of "average eigenvector" for all the layers.

*4.2.3 Layers identifying different clusters.* In this section, we will consider $M = k = 5$. Each layer is going to convey information about one and only one cluster. In the first layer, the nodes from cluster 1 are connected with probability $p_{1,1} = HIGH$. Two nodes from another cluster are connected with probability $p_{1,j} = HIGH - \delta$, $j \neq 1$. Two nodes that do not belong to the same cluster are also connected with probability $HIGH - \delta$. We generalizes this to all the layers : the layer $j$ conveys information about the $j^{th}$ cluster (that is to say $p_{j,j} = HIGH$, and all other connection probabilities are equal to $HIGH - \delta$). Figure 7 shows the different performances of the algorithm when $\delta$ increases. We see that until $\delta = 0.6$, SC-SR shows equal performance or very little improvements compared to the other algorithms. SC-GED is not part of the best algorithms. Moreover for $\delta = 0.7$, the algorithms are clearly outperformed by SC-SUM. It was expected in this kind of setting that SC-SUM behaves well : as it is averaging, and as the probabilities of connection of cluster $i$ in layer $i$ are bigger than the noise of connection in the other layers. However SC-SR should have worked quite well, it should be able to catch the specificities of each layer. On the contrary, we could have expected that SC-GED did not work quite well. Indeed, the Laplacian matrices are very different, finding common eigenvectors might not be the best approximation possible.

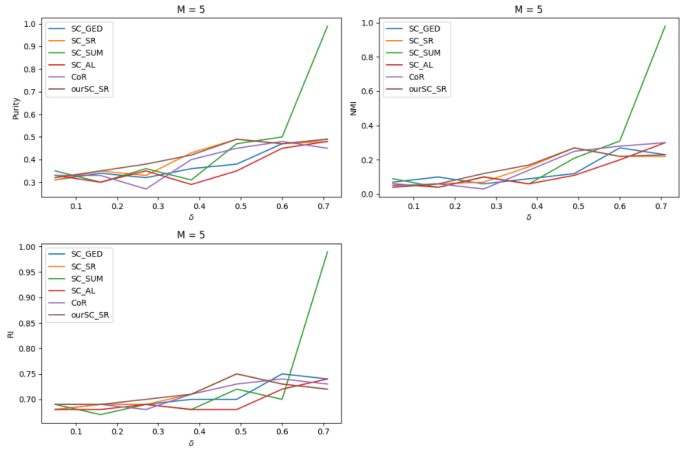


**Figure 7: Purity, NMI and RI when $\delta$ increases**

*4.2.4 ourSC-SR compared to SC-SR.* When we look closely at Figures 6 and 7, we see that SC-SR and ourSC-SR often have very similar performances (it is even impossible to make the difference between them in Figure 6), and ourSC-SR sometimes shows better performance (but sometimes worse). Moreover, if we compare the runtime of the algorithms (see Figure 8, that corresponds to the runtime of the last test of section 4.2.2), ourSC-SR is around 1.5 faster. This is globally the time difference we noticed between the two algorithms during our experiments.

| Algorithm | Purity | NMI | RI | Run Time |
|---|---|---|---|---|
| SC_GED | 0.92 | 0.82 | 0.94 | 40.80 |
| SC_SR | 0.92 | 0.84 | 0.94 | 1.23 |
| SC_SUM | 0.44 | 0.19 | 0.72 | 0.00 |
| SC_AL | 0.37 | 0.15 | 0.71 | 0.20 |
| CoR | 0.77 | 0.67 | 0.88 | 26.43 |
| ourSC_SR | 0.92 | 0.84 | 0.94 | 0.85 |

**Figure 8: Run time of the different algorithms for the last test of Figure 6**

## 4.3 Final insights and potential next steps

We ran a lot of supplementary tests that we did not include in the report, but that are available in the code going along with it.

After having done these experiments, it seems that SC-GED and SC-SR are able to show very good performance, but on very specific settings. In most cases, the baseline algorithms show equal or better performance, with a running time which is much better (see Figure 8). SC-GED and SC-SR are useful in settings where the data is very noisy. Moreover, they do not benefit from the repetition of patterns.

A lot of interesting work could still be done about these algorithms. Indeed, we conducted a lot of different tests to understand the particularities of each algorithm, but we made a lot of simplifications because the number of possibilities is huge.

Interesting next steps could have been, if we had more time:

- Have deeper insights for hyperparameter selection in SC-GED.
- Have a deeper insight on the first layer selection in SC-SR and find a middle ground between getting rid of needing ground truth and choosing completely at random as in randomSC-SR.
- Use a clustering algorithm other than K-Means once the data has been embedded in the low-dimensional space.
- Apply to real-world data.

# REFERENCES


Xiaowen Dong, Pascal Frossard, P. Vandergheynst, and N. Nefedov. 2012. Clustering With Multi-Layer Graphs: A Spectral Perspective. *IEEE Transactions on Signal Processing* 60, 11 (nov 2012), 5820–5831. https://doi.org/10.1109/tsp.2012.2212886

Nathan Eagle and Alex (Sandy) Pentland. 2006. Reality Mining: Sensing Complex Social Systems. *Personal Ubiquitous Comput.* 10, 4 (mar 2006), 255–268. https://doi.org/10.1007/s00779-005-0046-3

Abhishek Kumar, Prateek Rai, and Hal Daumé III. 2010. Co-regularized Spectral Clustering with Multiple Kernels. In *NIPS 2010 Workshop: New Directions in Multiple Kernel Learning*.

Nikolay Nefedov. 2011. Multi-Membership Communities Detection in Mobile Networks. In *Workshop of the International Conference on Web Intelligence, Mining and Semantics*.

Ronan Perry, Gavin Mischler, Richard Guo, Theodore Lee, Alexander Chang, Arman Koul, Cameron Franz, Hugo Richard, Iain Carmichael, Pierre Ablin, Alexandre Gramfort, and Joshua T. Vogelstein. 2021. mvlearn: Multiview Machine Learning in Python. *Journal of Machine Learning Research* 22, 109 (2021), 1–7.

Satu Elisa Schaeffer. 2007. Survey: Graph Clustering. *Comput. Sci. Rev.* 1, 1 (aug 2007), 27–64. https://doi.org/10.1016/j.cosrev.2007.05.001

Jianbo Shi and Jitendra Malik. 2000. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. and Mach. Intell.* 22, 8 (Aug. 2000), 888–905.

Dengyong Zhou and Bernhard Schölkopf. 2004. A Regularization Framework for Learning from Graph Data. In *ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*. 132–137.