Uncovering Hidden Block Structure for Clustering

Luce le Gorrec[1]✉, Sandrine Mouysset[1], Iain S. Duff[2], Philip A. Knight[1], and Daniel Ruiz[1]

[1] Université de Toulouse - IRIT, 2 rue Camichel, Toulouse, France
{forename}.{surname}@enseeiht.fr
[2] R71, STFC Rutherford Appleton Laboratory, Didcot, Oxon, OX11 0QX, UK and CERFACS, Toulouse, France iain.duff@stfc.ac.uk
[3] University of Strathclyde, Richmond Street, Glasgow, UK
p.a.knight@strath.ac.uk

Abstract. We present a multistage procedure to cluster directed and undirected weighted graphs by finding the block structure of their adjacency matrices. A central part of the process is to scale the adjacency matrix into a doubly-stochastic form, which permits detection of the whole matrix block structure with minimal spectral information (theoretically a single pair of singular vectors suffices).
We present the different stages of our method, namely the impact of the doubly-stochastic scaling on singular vectors, detection of the block structure by means of these vectors, and details such as cluster refinement and a stopping criterion. Then we test the algorithm's effectiveness by using it on two unsupervised classification tasks: community detection in networks and shape detection in clouds of points in two dimensions. By comparing results of our approach with those of widely used algorithms designed for specific purposes, we observe that our method is competitive (for community detection) if not superior (for shape detection) in comparison with existing methods.

Keywords: clustering and unsupervised learning · spectral clustering· doubly-stochastic scaling· community detection· shape detection.

1 Introduction

Grouping together similar elements among a set of data is a challenging example of unsupervised learning for which many so-called clustering techniques have been proposed. A classical way to cluster elements consists in finding blocks in matrices used to represent data—such as adjacency matrices, affinity matrices, contingency tables or a graph Laplacian.

The link between spectral and structural properties of matrices is a motivating factor behind many existing clustering algorithms [12, 28, 16]. To extract multiple clusters, classical spectral algorithms recursively split the nodes in two sets by means of the signs of a given singular vector or eigenvector. This can be costly since each partition into two clusters requires a new vector to be computed. Moreover, the returned partition at each iteration may be a poor fit to the natural block structure in the matrix (an example is shown in the rightmost plot of Figure 2, where the red separation line does not match the blocking). Other existing methods [20, 24, 30] rely on spectral embedding with various graph Laplacian matrices. Such methods require a priori knowledge of the number of clusters.

In this paper, we present a novel approach to spectral clustering. Our aim is to be able to find clusters in a strongly connected, directed weighted graph. We work with a matrix $\mathbf{A}$ that represents the weighted adjacency graph.

The main novelty of our method is to use a doubly-stochastic scaling of the matrix $\mathbf{A}$ before extracting spectral components. That is we find two positive diagonal matrices $\mathbf{D}$ and $\mathbf{F}$ so that all the row sums and column sums of the scaled matrix $\mathbf{P} = \mathbf{DAF}$ are equal to one. We show in Section 2 that the scaling improves the fidelity of the structural information contained in the leading singular vectors of $\mathbf{P}$. In particular, we only need to compute a few singular vectors to obtain significant information on the underlying structure without any prior information on the number of blocks. Additionally, working with singular vectors of $\mathbf{P}$ makes it feasible to analyse directed graphs, and can provide an accurate clustering when the graph structure is distinctly asymmetric.

The paper is structured as follows. In Section 2 we outline the connection between the singular vectors of a doubly-stochastic matrix scaling and its block structure. In Section 3 we use tools from signal processing to detect clustering information in the vectors. Section 4 is dedicated to the rearrangement of the blocks: we have to collate information provided by left and right singular vectors and further refinement may be necessary when we iterate our process by analysing several singular vectors in turn. We also present a bespoke measure we have designed to evaluate the quality of the clustering, and to derive a stopping criterion. Empirical results using benchmark data sets are provided in Section 5 to indicate the effectiveness of our algorithm and finally we present some concluding remarks in Section 6.

2 Doubly-stochastic Scaling

Scaling a matrix into doubly-stochastic form is always possible if the matrix is *bi-irreducible*, that is there exist no row and column permutations that can rearrange $\mathbf{A}$ into a block triangular form [29]. If $\mathbf{A}$ is the adjacency matrix of a strongly connected directed graph then it is bi-irreducible so long as its diagonal is zero free. If necessary, we can add nonzero terms to the diagonal without affecting the underlying block structure.

The following theorem is a straightforward consequence of the Perron–Frobenius theorem [17, 26].

Theorem 1. *Suppose that $\mathbf{S} \in \mathbb{R}^{n\times n}$ is symmetric, irreducible, and doubly-stochastic. The largest eigenvalue of $\mathbf{S}$ is 1, with multiplicity 1, and the associated* *eigenvector is equal to a multiple of* $\mathbf{e}$*, the vector of ones. If* $\mathbf{S}$ *has total support–that is* $\mathbf{S}$ *can be permuted symmetrically into a block diagonal structure with* k *irreducible blocks, then the eigenvalue* 1 *has multiplicity* k*. A basis for the corresponding eigenvectors is*

$$\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\},$$

where v_{pq} *for* $p \in \{1, \ldots, k\}$ *is given by*

$$v_{pq} = \begin{cases} 1, & \text{for all } q \text{ in block } p \\ 0, & \text{otherwise.} \end{cases}$$

If we compute an eigenvector of such a matrix $\mathbf{S}$ associated with the eigenvalue 1 then it will be of the form

$$\mathbf{x} = a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k.$$

By forcing $\mathbf{x}$ to lie in the orthogonal complement of the constant vector $\mathbf{e}$, it is reasonable to assume that the a_i are distinct. Since these vectors are formed by a disjoint partition of $\{1, \ldots, n\}$ we can identify the precise contribution from each block, and then characterise the partition exactly using the set

$$\{a_1, \ldots, a_k\}.$$

Indeed, reordering $\mathbf{x}$ according to the block structure puts it in piecewise constant form. Conversely, symmetrically permuting rows and columns of $\mathbf{S}$ according to the ascending order of entries from $\mathbf{x}$ reveals the block structure of $\mathbf{S}$.

As a corollary of Theorem 1, consider an non-symmetric but still doubly-stochastic matrix $\mathbf{P}$. Both $\mathbf{P}\mathbf{P}^T$ and $\mathbf{P}^T\mathbf{P}$ are also doubly-stochastic, and the results of Theorem 1 apply. Suppose then that we compute the principal left and right singular vectors of $\mathbf{P}$. If either $\mathbf{P}\mathbf{P}^T$ or $\mathbf{P}^T\mathbf{P}$ exhibits some block structure, then the dominant singular vectors of $\mathbf{P}$ have contributions from each of the corresponding basis vectors. Thus we can directly identify the row and/or column block structure of $\mathbf{P}$ using the same trick as for the matrix $\mathbf{S}$.

Our algorithm is designed for matrices that are perturbations of matrices with total support. If the matrix has a structure that is close to block diagonal, the leading singular vectors can be assumed to have a structure similar to the piecewise constant form. The impact of a small perturbation on the scaling of a block diagonal matrix can be found in [11, §2.1]. If we then look for steps in the computed vectors we should be able to reveal some underlying approximate block structure, or at least some row blocks (respectively column blocks) that are weakly correlated with each other. To find a doubly-stochastic scaling we use the Newton method described in [18]. Typically, this is very cheap to do, requiring only matrix–vector products.

To emphasize the power of the doubly-stochastic scaling compared to other spectral clustering techniques, we use a small example in Figure 1 to illustrate how the spectral information may vary when extracting eigenvectors from either the

Laplacian or from the doubly-stochastic matrix. In this example, the graph has three distinct clusters that are loosely linked together. To ensure bi-irreducibility the diagonal of the adjacency matrix of the graph is set to 10^{-8} times the identity matrix, before scaling to doubly-stochastic form. In the lower half of Figure 1 we see the numerical structure of the eigenvectors used to identify the clusters. The lower graph is a sorted version of the upper. For the Laplacian we have illustrated the eigenvectors of the two smallest positive eigenvalues (the Fiedler vector in blue) and for the stochastic scaling we have used the two subdominant vectors. In the case of the Laplacian matrix, we can see that the eigenvectors cannot easily resolve the clusters (the same is also true for the normalised Laplacian, too). On the other hand, for the doubly-stochastic scaling we can use either of the vectors to resolve the clusters perfectly and unambiguously.

As we said in Section 1, it is possible to iterate our process of finding and analysing singular vectors to refine our clustering. Such an iterative process needs to analyse different singular vectors at each step to obtain additional information. In [11, §4], we describe our implementation.

3 Cluster Identification

A key stage in our algorithm is to identify clusters using the largest singular vectors of the doubly-stochastic bi-irreducible matrix $\mathbf{P}$. We note that we are assuming no a priori knowledge regarding the number of clusters nor the row and column permutations that may reveal the underlying block structure.

Partitioning algorithms that work on vectors analogous to the Fiedler vector divide the matrix in two parts according to the signs of the vector entries. But

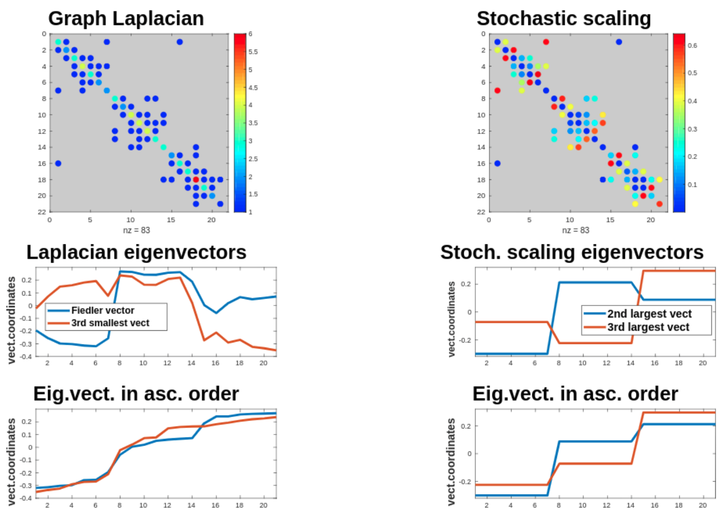

Fig. 1. Two matrix representations of a graph and their corresponding eigenvectors

thanks to our previous doubly-stochastic scaling we are able to see more than two blocks, as explained in Section 2. If $\mathbf{PP}^T$ or $\mathbf{P}^T\mathbf{P}$ has an underlying nearly block diagonal structure, its leading eigenvectors are expected to have nearly piecewise constant pattern when reordered according to the size of their entries, as in the right bottom plot of Figure 1. Our problem is thus to detect steps in the reordered vector, which we choose to view as a signal processing issue. In fact, this vector can be considered as a 1D signal whose steps are to be detected. Hence, we proceed in our cluster detection by applying tools from signal processing, in particular using a convolution product between the current singular vector and a specific filter. The peaks in the convolution product correspond to edges in the signal. These tools have the added bonus of not needing prior knowledge of the number of clusters.

We have chosen to use the Canny filter [4], which is widely used in both signal and image processing for edge detection. To optimise edge detection it combines three performance criteria, namely good detection, good localization, and the constraint that an edge should correspond to a peak in an appropriate convolution. In order to satisfy these criteria the Canny filter uses an operator constructed from the convolution product of the output Signal-to-Noise Ratio (SNR) and a localization function (the filter). The optimal localization function is the first derivative of a Gaussian kernel [4].

Much effort has gone into refining our implementation of edge detection to make it optimal for our particular application. When using the filter for the convolution we can vary a parameter that we call the sliding window size. The window size determines whether the filter has a narrow support and has a steep profile (as happens with a small window) or a broad support and a smoother profile (with a big window), as shown in Figure 2(a). A small window can create many peaks whereas a larger one will detect fewer edges.

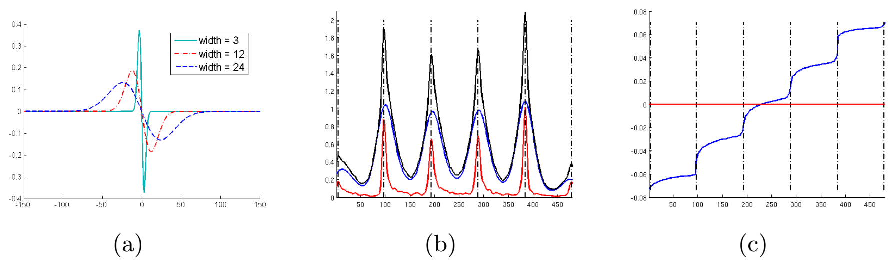

Fig. 2. (a) The filter for different sliding window sizes; (b) Convolution products; (c) Step detection

To avoid the side effects of the window size on the convolution product the step detection is performed with two window widths of size $n/30$ and $n/150$

respectively where n is the size of the vector. We then sum both convolution results in order to detect the principal peaks and to make sure that our steps are both sharp and of a relatively significant height.

Figure 2(b) shows an example of how the sliding window can affect the step detection using the 480×480 matrix `rbsb480` from the SuiteSparse collection [8]. We can see from this figure that the peaks corresponding to steps in the vector are better identified using this sum. In Figure 2(c), the associated right singular vector is plotted indicating the quality of the steps detected by the filters. In contrast, the horizontal red line in Figure 2(c) indicates the bipartitioning suggested by the Fiedler vector. This presumes that there are precisely two clusters and the separation is not aligned with any of the steps indicated by the singular vector.

It is possible for the edge detector to identify spurious clusters. In [11, §6] we give an illustrative example and also describe how to deal with this issue in general by validating peaks and also by incorporating additional information provided by other singular vectors to resolve the boundaries between blocks with high precision.

4 Cluster Improvement

We now explain how to merge the different clusters found for each individually analysed singular vector, as well as those found on rows and columns once the iterative process is over. We then present a bespoke measure to evaluate the quality of our clustering which allows us to amalgamate some small clusters. Using the measure, we develop a stopping criterion to determine convergence of the iterative process.

Throughout this section we illustrate our comments with the `rbsb480` matrix [8] because of its interesting fine block structure on both rows and columns. We observe that in Figures 3 and 4, we are not presenting a clustering in a formal sense, but simply an overlapping of the independent row and column partitions.

4.1 Overlapping the Clusterings

Each time our algorithm analyses a right (respectively left) singular vector of the matrix it may find a row (respectively column) partition of the matrix. Of course, the resulting partition may be different for every analysed vector and we need to merge them in an embedded manner. This process is illustrated in Figure 3, where the first detection step identifies only 4 row clusters and 5 column clusters, while the second left and right singular vectors suggest a different picture with 3 row clusters and 4 column clusters. These new clusters are complementary to the first set and the merged clustering gives 12 row clusters and 20 column clusters with fairly well separated sub-blocks in the resulting reordered matrix, as shown in the left-hand plot in Figure 4.

As we can see from our tests on `rbsb480`, it may be sensible to analyse several vectors in turn and merge these results. However, this can be overdone and we may produce a very fine partitioning of the original matrix. This issue is illustrated in

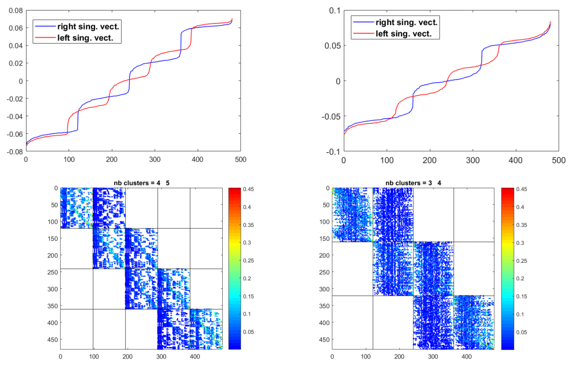

Fig. 3. Cluster identification using different singular vectors

the right-hand plot of Figure 4, where 3 steps have been merged to give a fine grain clustering (with 48 by 100 clusters) and motivates our development of an effective method for amalgamating blocks.

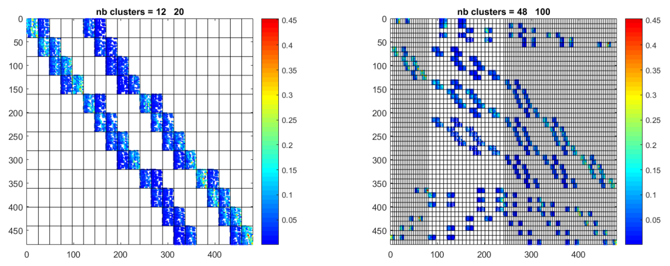

Fig. 4. Recursive cluster identification obtained by analysing several vectors in turn

4.2 Quality Measure

We base our cluster analysis on the modularity measure from [23]. Modularity can be interpreted as the sum over all clusters of the difference between the fraction of links inside each cluster and the expected fraction, by considering a random network with the same degree distribution.

We can evaluate our row clustering using the modularity of our doubly-stochastic matrix $\mathbf{PP}^T$ given by the formula

$$\mathcal{Q}_r = \frac{1}{n}\sum_{k=1}^{r_C}\left(\mathbf{v}_k^T\mathbf{PP}^T\mathbf{v}_k - \frac{1}{n}|\mathcal{J}_k|^2\right). \qquad (1)$$

Here, n is the number of rows in $\mathbf{P}$, r_C is the number of row clusters and $|\mathcal{J}_k|$ is the cardinality of cluster k. The equation is easily derived from the definition of modularity from [1], noticing that $2m = n$ in the doubly-stochastic case. In a similar way, we derive a quality measure $\mathcal{Q}_c$ for the column clustering by considering the matrix $\mathbf{P}^T\mathbf{P}$ and summing over the sets of column clusters.

In [11, §7.1] we prove that the quality measure falls between the bounds

$$0 \leq \mathcal{Q}_r \leq 1 - \frac{1}{r_C}\,, \tag{2}$$

with the upper bound being reached only when the r_C clusters have the same cardinality.

Despite issues with the so-called resolution limit [13], modularity is still one of the most widely used measures in the field of community detection and we are comfortable in employing it here. Furthermore, in the specific case of k-regular graphs, many measures including the Newman and Girvan modularity behave similarly [6]. Since doubly-stochastic matrices can be considered as adjacency matrices of weighted 1-regular graphs most common quality measures can be expected to give similar results.

To avoid tiny clusters as in the right plot of Figure 4, we recursively amalgamate the pair of clusters that gives the maximum increase in modularity measure. The algorithm stops naturally when there is no further improvement to be made by amalgamating pairs of clusters together. A proof that this method always provides a local maximum can be found in [11, §7.2] where it is also shown that it is computationally cheap.

The amalgamation process prevents an explosion in the number of clusters and gives some control over the amount of work for testing the $r_C(r_C-1)/2$ possibilities and is applied after overlapping the clusterings obtained independently for each singular vector.

As a stopping criterion, we compare the quality of the clustering updated by the amalgamation process, $\mathcal{Q}_r^{upd}$ with that obtained immediately beforehand, $\mathcal{Q}_r^{ref}$. There are four possibilities to consider.

- If $\mathcal{Q}_r^{upd} < \mathcal{Q}_r^{ref}$ we reject the update.
- If $\mathcal{Q}_r^{upd} > \mathcal{Q}_r^{ref}$ but the number of clusters increases we accept the update only if there is a sizeable jump in modularity taking into account the upper bound in (2). We compare the quality measure increase with its theoretical potential increase using the ratio

$$\rho = \frac{\mathcal{Q}_r^{upd} - \mathcal{Q}_r^{ref}}{\frac{1}{nbClust^{ref}} - \frac{1}{nbClust^{upd}}}.$$

  The threshold can be tuned to control the number of iterations we take and the granularity of the clustering. In tests, a threshold value between 0.01 and 0.1 seems to be effective.
- If $\mathcal{Q}_r^{upd} > \mathcal{Q}_r^{ref}$ and the number of clusters does not increase, we accept the update.
- If the clusterings are identical up to a permutation then we can terminate the algorithm, which is equivalent to rejecting the update.

5 Applications

We illustrate the performance of our algorithm in community detection in Section 5.1 and we test it on shape detection using the `Scikit-learn` package [25] in Section 5.2.

5.1 Community Detection

We first test our algorithm on simple networks. To ensure bi-irreducibility with minimal effect on the clusters we set the diagonal to 10^{-8}.

Before the detection phase, we remove dominant entries (> 0.55) from the scaled matrix since they give rise to near-canonical eigenvectors that tend to lack well defined steps. From the first iteration, we consider each dominant entry as an already identified block when looking at the projected eigenvector of the matrix, as explained in [11, §3]. These large entries are associated with structures not strongly linked to communities, such as hub or pendant nodes, and we incorporate them during the amalgamation process to optimise modularity once our algorithm has converged.

We have compared our algorithm (US) against four established community detection algorithms from the `igraph` package [7], namely Walktrap (WT) [27], Louvain (ML) [2], Fast and Greedy (FG) [5], and Leading Eigenvector (LE) [22]. WT and ML are specifically designed to work on graph structures directly whereas LE focuses (like ours) on exploiting spectral information. FG is a natural greedy algorithm to optimise the modularity. WT and ML have been shown to be among the three best community detection algorithms in a recent study [31].

We have applied these algorithms to four datasets of 500 node networks generated with the Lancichinetti–Fortunato–Radicchi benchmark (LFR) [19] which is widely used in community detection because its characteristics are close to those of real-world networks. Each dataset corresponds to a different average degree $\overline{k}$. More details are given in Table 1.

Table 1. Parameters for benchmark generation

Parameter	Value	
number of nodes	500	
average degree $\overline{k}$	$\{10,20,40,75\}$	
maximum degree	$2 \times \overline{k}$	
degree distribution exponent	-2	*Exponent of the power law used to generate degree distribution.*
community size distribution exponent	-1	*Exponent of the power law used to generate community size distribution.*

We test the behaviour of community detection algorithms using the network mixing parameter, which quantifies the strength of a community structure. Initially, it measures the strength of a node's community membership by computing the ratio between its links outside the community and its degree. The greater the mixing parameter for each node, the weaker the community structure. The network mixing parameter μ is the mean value of the nodal mixing parameters.

We measure the accuracy of the community structures returned by the algorithms by using the Normalised Mutual Information (NMI). This is a measure taken from information theory (see for instance [15]). It measures the deviation between two candidate partitions with no requirement that they have the same number of blocks and is widely used in the community detection field [31]. The NMI takes its values in $[0, 1]$, and is equal to 1 when the two partitions are identical.

The results are shown in Figure 5. Each plot corresponds to a different value of $\overline{k}$. In each plot, we have generated 50 networks for each μ value and display the mean value of the NMI. We remark that we can pair existing algorithms in terms of performance: ML and WT, where NMI is close to 1 even for large values of μ, and FG and LE where NMI decreases rapidly. Our algorithm falls between the two groups.

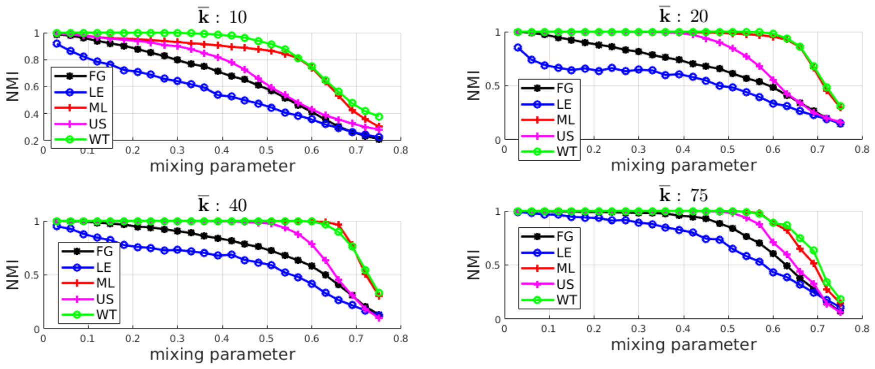

Fig. 5. Curves of NMI for FG, LE, ML, WT, US

We note that the two groups tend to behave similarly in terms of NMI when increasing $\overline{k}$. This can be explained by the constraints on the network that result in larger communities for large $\overline{k}$ values, as shown in Figure 6. Here, two adjacency networks from our dataset have been plotted with $\mu = 0.3$. The matrix in the left plot has an average degree $\overline{k} = 10$ and 40 communities, whereas for the matrix in the right plot $\overline{k} = 75$ and there are 5 communities. It is known that large communities are more accurately detected by most community detection algorithms. But it seems that it is not the only factor that makes our algorithm work better for large $\overline{k}$, as its NMI curves are more sensitive to $\overline{k}$ than FG and LE.

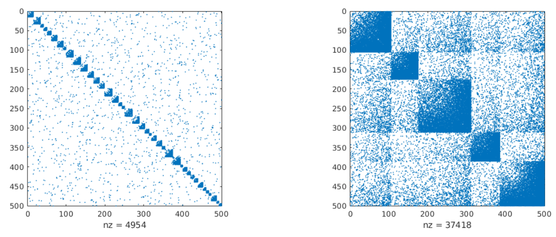

Fig. 6. Adjacency matrices of two networks

We have tested whether this behaviour may be due to the fact that, in very sparse networks, leading eigenvectors may be associated with features other than the community structure [14]. Thus, we have compared the accuracy of two versions of our algorithm. In the first we add 10^{-8} to the diagonal entries of $\mathbf{A}$ (USd). In the second we add a perturbation to the whole matrix (USw): this version works with $\mathbf{A} + \epsilon \mathbf{e}\mathbf{e}^T$ where $\mathbf{A}$ is the adjacency matrix of the network. We have set ϵ to 0.15 as is often suggested with Google's Pagerank [3]. The results of both algorithms can be viewed in Figure 7 for two different values of $\overline{k}$. We observe that the two versions behave similarly when $\overline{k} = 75$, whereas USw (blue curve) is much more accurate than USd (pink curve) for $\overline{k} = 20$.

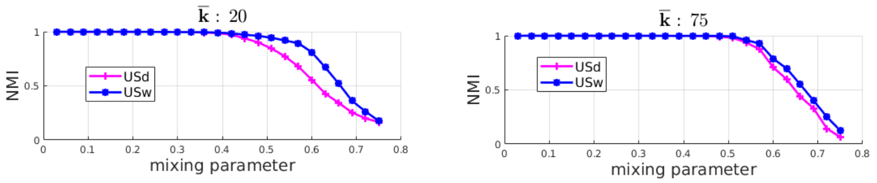

Fig. 7. Curves of NMI for USd and USw

To complete this study, we have investigated the behaviour of WT, ML, USd and USw with increasing $\overline{k}$. The results are shown in Figure 8. We see that ML, WT and USw first improve their accuracy, seem to reach a threshold and finally worsen for the largest value of $\overline{k}$. USd also improves its accuracy, but does not worsen when $\overline{k}$ reaches its maximum test value.

Hence, although our algorithm does not beat all existing community detections algorithms on very sparse networks, it is able to correctly detect communities better than some widely-used purpose-built algorithms. Moreover it seems to be

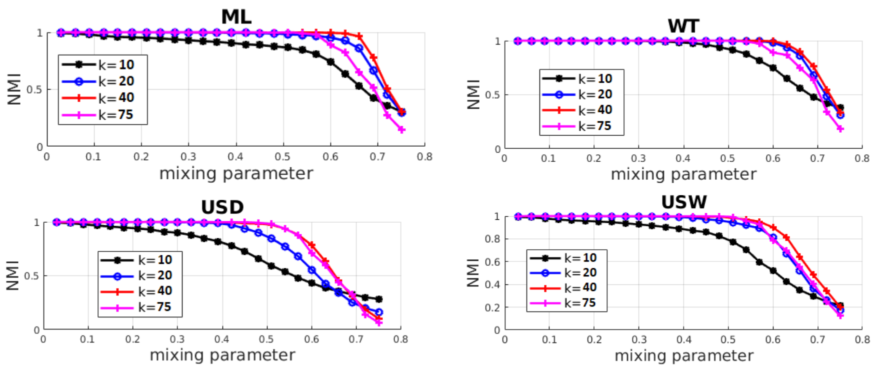

Fig. 8. NMI curves of ML, WT, USd, USw for some $\overline{k}$ values

a very encouraging alternative when the networks become denser as it retains its accuracy in these circumstances. Indeed, we will see in Section 5.2 that it is very good at detecting block structures in affinity matrices, and such matrices are nothing but adjacency matrices of weighted complete graphs.

Finally, our algorithm is not constrained to work with only symmetric matrices and so provides a versatile tool for community detection in directed graphs even when these graphs have unbalanced flows, i.e. an imbalance between links that enter and then leave a subgroup. Moreover, WT and ML, which both symmetrize the matrix by working on $A + A^T$, produce spurious results in this case. An example is shown in Figure 9, where two communities are connected in an asymmetric fashion, and the partitions suggested by the algorithms are shown. Thus by working on the normal equations of the scaled matrix we are able to perfectly detect the community structure of the graph, while both WT and ML fail.

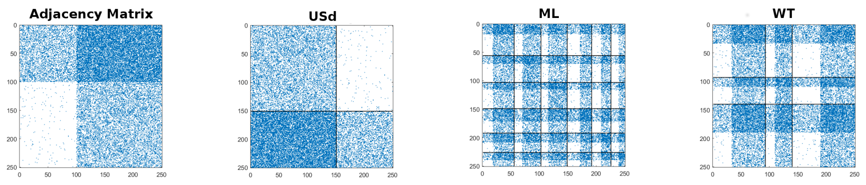

Fig. 9. Detecting asymmetric clusters

5.2 Shape Detection

To show the potential of our algorithm for other clustering tasks, we have tested it on datasets from the `Scikit-learn` package [25]. To enable a quick visual validation, we have used it to detect, through their affinity matrices, coherent clouds of points in two dimensions with 1500 points for each dataset. The affinity matrix [24] of a set of points $\{x_i \in \mathbb{R}^p, i = 1...n\}$ is the symmetric matrix with zeros on the diagonal and

$$\mathbf{A}_{i,j} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right),$$

elsewhere. The accuracy of our method strongly depends on the choice of the Gaussian parameter σ. We follow the prescription in [21] to take into account both density and dimension of the dataset. Since the affinity matrix has a zero diagonal we enforce bi-irreducibility as before by adding 10^{-8} to the diagonal. As for community detection, we preprocess by removing dominant entries in the scaled matrix. In postprocessing we reassign these entries to the cluster that contains their closest neighbour in Euclidean distance.

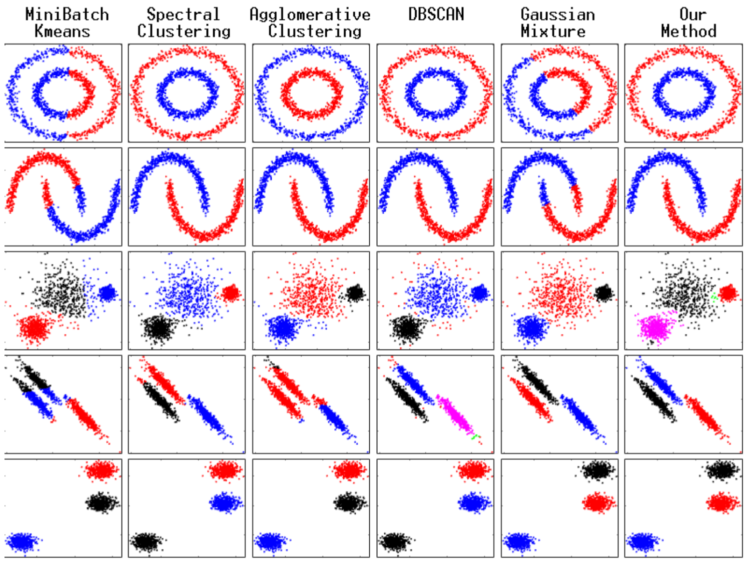

Fig. 10. Comparison between clustering algorithms

As a baseline, we have compared it with the clustering algorithms from the `Scikit-learn` package. The results for five of these algorithms are provided in Figure 10. Here each row corresponds to a specific two-dimensional dataset, and each column to an algorithm. The points are coloured with respect to the cluster they have been assigned by the algorithm. The rightmost column gives the results for our method when we stop after the analysis of a single eigenvector. It is clear that this vector provides enough information to roughly separate the data into coherent clusters for all the datasets whatever their shape. The NMI is given in Table 2, following the same order as in Figure 10 for the datasets. We see that our algorithm always achieves the best score except for two datasets: the fourth for which one element has been misplaced by the postprocessing and the third for which our algorithm still has a NMI larger than 0.9. We also note that, except for `DBSCAN`, all these algorithms need to know the number of clusters.

Table 2. NMI for the clusterings shown in Figure 10

MiniBatch Kmeans	Spectral Clust.	Agglo. Clust.	DB SCAN	Gaussian Mixture	Our method
$2.9 \times 10{-}4$	**1**	0.993	**1**	$1.3 \times 10{-}6$	**1**
0.39	**1**	**1**	**1**	0.401	**1**
0.813	0.915	0.898	0.664	**0.942**	0.902
0.608	0.942	0.534	0.955	**1**	0.996
1	**1**	**1**	**1**	**1**	**1**

6 Conclusions

We have developed a spectral clustering algorithm that is able to partition a matrix by means of only a few singular vectors (sometimes as few as one), mainly thanks to the spectral properties of the doubly-stochastic scaling. Moreover, our method does not need to know the number of clusters in advance and needs no artificial symmetrization of the matrix. We have illustrated the use of our algorithm on standard data analysis problems, where we are competitive with methods specifically designed for these applications. However, it is more versatile than those classical algorithms because it can be applied on matrices from very diverse applications, simply needing adaptive post- and pre-processing to be used on specific applications.

In future work we aim to customise our method for co-clustering since it naturally detects rectangular patterns in square matrices. In Figure 11 we compare our algorithm with the spectral co-clustering algorithm designed by Dhillon in [9]. We observe that Dhillon's algorithm misplaces elements of the 8 cluster matrix whilst our algorithm succeeds in recovering the ground truth structure. While we have to keep in mind that these co-clusterings have to be understood as the overlapping of independent clusterings on the matrix rows and columns, and that it is shown in [10] that the intertwining between row and column clusters

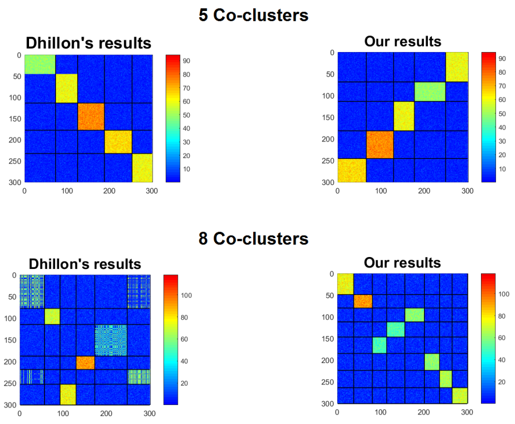

Fig. 11. Detection of co-clusters

is an important factor in co-clustering quality, our preliminary results are most encouraging. A study of the algorithm complexity is ongoing, as well as an efficient implementation. For now, a simple Matlab implementation can be found in the Data Mining section of the webpage `http://apo.enseeiht.fr/doku.php?id=software`.

References

1. Bagrow, J.P.: Communities and bottlenecks: Trees and treelike networks have high modularity. Physical Review E **85**(6), 066118 (2012)
2. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment **2008**(10), P10008 (2008)
3. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. **30**(1-7), 107–117 (Apr 1998)
4. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Analysis and Machine Intelligence **8**, 679–698 (1986)
5. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Phys. Rev. E **70**(6), 066111 (Dec 2004)
6. Conde-Cespedes, P.: Modélisation et extension du formalisme de l'analyse relationnelle mathématique à la modularisation des grands graphes. PhD Thesis, Université Pierre et Marie Curie (2013)
7. Csardi, G., Nepusz, T.: The igraph software package for complex network research. InterJournal **Complex Systems**, 1695 (2006)
8. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. ACM Transactions on Mathematical Software **38**(1), 1–14 (2011)
9. Dhillon, I.S.: Co-clustering documents and words using bipartite spectral graph partitioning. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 269–274. KDD '01, ACM (2001)

10. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 89–98. KDD '03, ACM (2003)
11. Duff, I., Knight, P., Le Gorrec, L., Mouysset, S., Ruiz, D.: Uncovering hidden block structure. Technical Report TR/PA/18/90, CERFACS (august 2018)
12. Fortunato, S.: Community detection in graphs. Physics reports **486**(3), 75–174 (2010)
13. Fortunato, S., Barthélemy, M.: Resolution limit in community detection. Proceedings of the National Academy of Sciences **104**(1), 36–41 (2007)
14. Fortunato, S., Hric, D.: Community detection in networks: A user guide. CoRR **abs/1608.00163** (2016)
15. Fred, A.L.N., Jain, A.K.: Robust data clustering. In: CVPR (2). pp. 128–136. IEEE Computer Society (2003)
16. Fritzsche, D., Mehrmann, V., Szyld, D.B., Virnik, E.: An SVD approach to identifying metastable states of Markov chains. Electronic Transactions on Numerical Analysis **29**, 46–69 (2008)
17. Frobenius, G.: Ueber matrizen aus nicht negativen elementen. Sitzungsber. Königl. Preuss. Akad. Wiss p. 456–477 (1912)
18. Knight, P.A., Ruiz, D.: A fast algorithm for matrix balancing. IMA Journal of Numerical Analysis **33**(3), 1029–1047 (2013)
19. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Physical Review E **78**(4), 046110 (Oct 2008)
20. Lei, J., Rinaldo, A., et al.: Consistency of spectral clustering in stochastic block models. The Annals of Statistics **43**(1), 215–237 (2015)
21. Mouysset, S., Noailles, J., Ruiz, D.: Using a global parameter for gaussian affinity matrices in spectral clustering. In: VECPAR. Lecture Notes in Computer Science, vol. 5336, pp. 378–390. Springer (2008)
22. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. Physical review E **74**(3), 036104 (2006)
23. Newman, M.: Analysis of weighted networks. Physical Review E **70**, 056131 (2004)
24. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic. pp. 849–856. NIPS'01, MIT Press (2001)
25. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
26. Perron, O.: Zur theorie der matrices. Mathematische Annalen **64**(2), 248–263 (1907)
27. Pons, P., Latapy, M.: Computing communities in large networks using random walks (long version). ArXiv Physics e-prints (Dec 2005)
28. Schaeffer, S.E.: Graph clustering. Computer Science Review **1**(1), 27–64 (2007)
29. Sinkhorn, R., Knopp, P.: Concerning nonnegative matrices and doubly stochastic matrices. Pacific J. Math. **21**, 343–348 (1967)
30. Von Luxburg, U.: A tutorial on spectral clustering. Statistics and computing **17**(4), 395–416 (2007)
31. Yang, Z., Algesheimer, R., Tessone, C.J.: A comparative analysis of community detection algorithms on artificial networks. Scientific Reports **6**, 30750 EP – (Aug 2016)