Iterated Matrix Reordering

Gauthier Van Vracem and Siegfried Nijssen ✉

UCLouvain - ICTEAM/INGI, Louvain-la-Neuve, Belgium
siegfried.nijssen@uclouvain.be

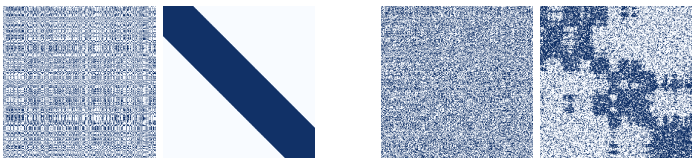
Abstract. Heatmaps are a popular data visualization technique that allows to visualize a matrix or table in its entirety. An important step in the creation of insightful heatmaps is the determination of a good order for the rows and the columns, that is, the use of appropriate matrix reordering or seriation techniques. Unfortunately, by using artificial data with known patterns, it can be shown that existing matrix ordering techniques often fail to identify good orderings in data in the presence of noise. In this paper, we propose a novel technique that addresses this weakness. Its key idea is to make an underlying base matrix ordering technique more robust to noise by embedding it into an iterated loop with image processing techniques. Experiments show that this iterative technique improves the quality of the matrix ordering found by all base ordering methods evaluated, both for artificial and real-world data, while still offering high levels of computational performance as well.

Keywords: Matrix reordering · Matrix seriation · Pattern mining · Clustering · Convolution

1 Introduction

Heatmaps are a popular data visualization technique that allows to easily visualize a matrix or table in its entirety. In the process of creating a heatmap an important step is however the determination of the order of the rows and columns used in the visualization; depending on the order chosen, the visualization may highlight different structures or patterns, such as clusters, that are of interest to understand the data being visualized. The challenge is that this specific order is not necessarily known and it is necessary to implement different *matrix reordering* algorithms that aim to find a permutation that best displays the data.

Many methods have been proposed in the literature that allow to permute a matrix in order to highlight a particular pattern (see [2] for a survey). However, in practice these methods are very sensitive to noise. This is illustrated in Fig. 1 for a banded Boolean matrix. On the right-hand side of Fig.1(a) we show the output of a reordering algorithm, a TSP solver in this case, applied on a random permutation of a banded matrix without noise, shown on the left-hand side; the algorithm successfully identifies the banded structure. However, if we flip each entry of the matrix with a 25% probability, the reordering method no longer recovers the banded structure, as shown in Fig. 1(b). Similar observations also



(a) Noiseless (b) 25% noise

Fig. 1: Impact of noise on the ordering of the same matrix. On the left, a random permutation. On the right, the matrix after applying the same ordering method

hold for other matrices and ordering methods, as we will see in the experimental section of this work.

In this paper we address this problem. We propose a novel type of method that allows to reorder data such that structures are also recovered in the presence of higher levels of noise. The key idea of this novel type of method is to reuse less effective methods by integrating them into an iterative loop that abstracts from noise by using convolution. *Convolution* is a technique that can be used to blur an image in order to reduce noise. We use convolution for different purposes: on the one hand, to reduce the noise of the matrix in a thresholding process; on the other hand to asses the quality of the order produced. The key idea is therefore to temporarily transform the data by removing the noise and then to reorder this simpler noiseless matrix with conventional methods. The developed technique allows to quickly order sparse or dense binary data while eliminating the noisiness. The advantage of the framework is that it allows to use any existing method (as a black box), and, as we will show experimentally, enhances the quality of the outputs of these existing algorithms.

The paper is structured as follows: in Section 2, we discuss the state-of-the-art methods. In Section 3 we introduce our framework. In Section 4 we conduct different experiments comparing existing methods and the framework on both artificial and real data. Finally, in Section 5, we lay the foundations for possible future improvements.

2 Related work

Even though our technique can also be used on non-binary data, in this work we focus on binary data, as many ordering techniques already exist for such data and hence we can evaluate our contributions well for such data. We will distinguish two types of binary matrices. The most generic form of binary data is a matrix of size $m \times n$, where m and n are not necessarily equal. Such *tabular* or *two-way two-mode* data can also be seen as a bipartite graph; the rows and columns of such matrices can be ordered independently from each other. A specific form of binary data are $n \times n$ matrices which represent *graphs* or *two-way one-mode* data (adjacency matrix). For such data the same order is typically used for both the rows and the columns.

To order rows and columns, we will distinguish three families of methods: *distance-based*, *structure-based* and *convolution-based* methods.

Distance-based methods aim to put together rows and columns that are similar. To do this, the original matrix is transformed into a distance or *dissimilarity matrix* $\mathbf{D}$. Each entry $\mathbf{D}_{i,j}$ represents the distance between row i and j (Euclidean or Manhattan distance for example). For *network* matrices, only one dissimilarity matrix is created. For *tabular* data two matrices are created (one for the columns and one for the rows), which results in two ordering problems. Once these matrices are built, their elements are reordered. This problem is known as the *Seriation problem* and is a combinatorial task that aims at ordering a one-dimensional set of objects $\{O_1, ..., O_n\}$ in order to optimize a merit/loss function. The *Seriation library* [8] in R offers a wide range of methods and heuristics. Behrisch et al. [2] provide a survey detailing and comparing these methods.

The advantage of dissimilarity matrices is that they allow the problem to be viewed from different perspectives. One way of solving it is to solve a *TSP* (traveling salesman problem) on the dissimilarity matrix [4], where a path over all the rows (or columns) of the matrix is found that maximized the similarity between each pair of adjacent rows.

One of the most used methods is hierachical clustering, in which a dendogram is built over data; using various algorithms, from this dendogram a seriation can be created that respects the dendogram.

All the methods described above, being based on the distance between rows, are efficient when there is little noise in the matrix. They also have the advantage of being applicable on both binary and numerical data. However, the introduction of noise can strongly disturb this similarity to the point that these methods can no longer produce a relevant order. This issue is similar for sparse matrices.

Structure-based methods aim to swap rows/columns in such a way that the resulting image looks most like a specific structure. Contrary to the previously described methods, the ordering of rows and columns are no longer two independent problems, but are solved in a combined process. Two basic methods aim at discovering different structures [3,7] and work exclusively on binary data:

Nested ordering consists in counting respectively for the columns and rows the number of 1s and then sorting them by these values. This allows to discover structures where there is a hierarchy between the different matrix entries.

The *barycenter* heuristic method aims at revealing a banded structure. For each row a barycentric index is computed which corresponds to the average position for all the 1. The rows are reordered according to this index, after which the process is iterated by alternating columns and rows until convergence.

These methods suffer from similar disadvantages as distance-based methods; they assume that there is a specific pattern to discover and are also very sensitive to noise.

The third type of method consists of methods based on ***convolution***, such as ConvoMap [1]. This method optimizes an evaluation criterion that measures the quality of an order by computing the difference between the image in this

order and a blurred version of the same image. The intuition is that if applying a blurring filter does not change an image significantly, this indicates that there are not many black-white transitions in the image; there are clear black and white parts in the image that remain unmodified by the blur. It was shown that this criterion indeed scores best those orders in which a clear structure visible. The main weakness of the ConvoMap method is that its local search algorithm is inefficient; in earlier work execution times of multiple hours were reported [1]. In this paper, we propose to reuse this evaluation criterion, but we integrate it in a very different approach: a framework that aims to reuse any ordering method as a black box component and to boost it by taking advantage of convolution.

3 Iterative Ordering using Convolution

We consider a binary matrix $\mathbf{M}$ of size $m \times n$ with $\mathbf{M}_{i,j} \in \{0, 1\}$, or an $n \times n$ matrix for networks, with $\mathbf{M}_{i,j} = \mathbf{M}_{j,i}$ and $\mathbf{M}_{i,i} = 0$.

The key idea of our iterative method is that we embed a *base ordering method* φ_b in an iterative loop to obtain a better output ordering, similar in spirit to how boosting methods improve the classification accuracy of an underlying base learning algorithm. We make the following two assumptions on the underlying base ordering method:

1. on data with a perfect, noiseless structure (such as a banded structure, a nested structure, a block structure, and so on), the ordering method will recover this structure;
2. on noisy data the ordering method will recover some of the structure in the data, although in an imperfect manner.

We will refer to the imperfect, incomplete patterns identified by the base method as *embryonic patterns*. Table 2 shows that most existing methods in literature are able to generate these fragments of dense patterns.

An overview of our iterative method is provided in Figure 2. Initially, the base method φ_b is used to produce a preliminary order; we assume that this order is informative, but imperfect due to noise. Subsequently, we apply convolution and thresholding to remove this noise from the image. The resulting data is once more put in the base ordering method, exploiting the assumption that for data without noise the method will produce a better ordering. The resulting order is used to reorder the original data (renoising), after which the process is repeated, until a stopping criterion indicates the process has converged.

We optimize the process by including an additional smoothing step in which we use local search to locally improve the solution.

The details of our approach are provided below.

Blur. The objective of convolution and thresholding is to obtain a clearer image with less variation; this should make it easier to order the data by the base algorithm. To achieve this goal a *convolution* operation is performed on the matrix. *Convolution* is an image processing technique that consists in filtering an

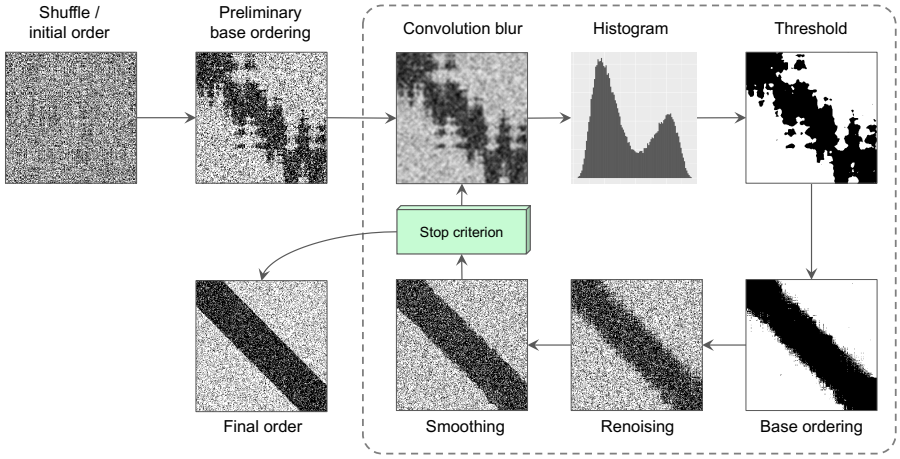


Fig. 2: Overview of the iterative framework with thresholding enabled

image by applying a *kernel* matrix $\mathbf{K}$ of size $k_m \times k_n$. Each pixel of the image is multiplied by the value of the surrounding pixels weighted by a specific weight. These weights are determined by the chosen kernel. We define the convolution operation as follows:

$$(\mathbf{M} * \mathbf{K})_{i,j} = \sum_{r=1}^{k_m} \sum_{c=1}^{k_n} \mathbf{K}_{r,c} \cdot \mathbf{M}_{i+r-\lceil k_m/2 \rceil,\, j+c-\lceil k_n/2 \rceil} \tag{1}$$

The effect of the convolution depends on the kernel used; some kernels have the effect of sharpening the image, others blur the picture. In this paper we only consider kernels that fall in this second category. The larger the kernel is, the stronger the removal of noise will be. In our algorithm we allow for several predefined kernels, but we will limit ourselves to one type of kernel in this paper: the linear kernel of size $k \times k$, defined as follows: $\mathbf{K}_{r,c} = k_{mid} - max\big(\big|k_{mid} - r - 1\big|,\ \big|k_{mid} - c - 1\big|\big) + 1$ with $k_{mid} = \lfloor k/2 \rfloor$.

Threshold (optional). After applying the blur convolution, the initial matrix is transformed from a binary one into a continuous one $\mathbf{N}$ with $\mathbf{N}_{i,j} \in [0, 1]$. Two strategies are therefore possible at this point. The first one consists in directly reordering the convoluted matrix with the base method. This is only possible for *distance-based* methods. In this case the thresholding step is **optional**. The second option consists in reconverting the blurred matrix into a binary one by applying a *thresholding* operation. The *thresholded* version of a Matrix $\mathbf{N}$ is defined by:

$$threshold(\mathbf{N},\, t)_{i,j} = \begin{cases} 1 & \text{if } \mathbf{N}_{i,j} > t \\ 0 & \text{if } \mathbf{N}_{i,j} \leq t \end{cases} \tag{2}$$

Algorithm 1: Framework

```
   input : M, K_c, κ, φ_b
 1 M ← φ_b(M)
 2 repeat
 3 |   e_0 ← error(M, M ∗ K_c)
 4 |   for ∀ K_i ∈ κ do
 5 |   |   N ← M ∗ K_i
 6 |   |   T ← threshold(N)
 7 |   |   M' ← φ_b(T) or φ_b(N)
 8 |   |   S ← smooth(M, M')
 9 |   |   e_1 ← error(S, S ∗ K_c)
10 |   |   if e_1 < e_0 then
11 |   |   |   M ← S
12 |   |   |   break
13 until e0 ≤ e1;
14 return M
```

Algorithm 2: Smoothing

```
   input : M, T
 1 repeat
 2 |   changed ← false
 3 |   h ← height(M)
 4 |   for i ← 1 to h − 1 do
 5 |   |   for j ← 2 to h do
 6 |   |   |   Δ_0 ← d(M_i, T_j) + d(M_j, T_i)
 7 |   |   |   Δ_1 ← d(M_i, T_i) + d(M_j, T_j)
 8 |   |   |   if Δ_0 < Δ_1 then
 9 |   |   |   |   swap(M_i, M_j)
10 |   |   |   |   changed ← true
11 |   M ← M^t, T ← T^t
12 until ¬changed;
13 return M
```

This filter consists in choosing a threshold $t \in [0, 1]$, which is used to partition the values of the matrix into two classes. To achieve this, the $OTSU$ method is used. This is a popular algorithm [6] that constructs a histogram of the intensities of the matrix values and then partitions it into two classes in order to minimize their intra-class variance. This method is based on the shape of the histogram and makes the assumption that the distribution of the values is modeled by a bi-modal distribution. This assumption is partially satisfied if the original image contains a well defined pattern divided into different dense areas.

Smoothing refinement. Once the convolution and optionally the threshold are applied, a noiseless binary matrix is obtained, which contains only the shape of the *embryonic pattern*. It is then easier to reorder this new simpler matrix. The embedded base algorithm is then executed and the produced order of rows and columns is applied to the initial matrix. This operation is called *renoising*.

Nevertheless some defects can remain, notably for the quality of the edges as shown in Table 5. To address this issue, we apply a refinement operation that we refer to as *smoothing*. The pseudo-code for this operation is provided in Algorithm 2. This technique consists in permuting the columns and rows of the matrix in order to minimize the difference between the renoised matrix $\mathbf{M}$ and its thresholded version $\mathbf{T}$ computed previously. We define the function $d(\mathbf{M}_i, \mathbf{T}_j)$ as the Hamming distance between row i of the matrix $\mathbf{M}$ and row j of the matrix $\mathbf{T}$. Essentially, we locally optimize the order of the renoised data such that this data more closely resembles the thresholded version.

Stopping criterion. The operations described above are repeated iteratively. For each iteration, a new order is produced but the challenge is to measure and quantify the quality of this produced image and inferring an evaluation or stopping criterion. Here we reuse the scoring function used in the ConvoMap algorithm [1], where the goal is to ensure that the image looks as similar as possible to its blurred version. The scoring function used as criterion is defined as the difference between a matrix $\mathbf{M}$ and its convoluted version after applying

a blurring kernel $\mathbf{K}$: $error(\mathbf{M}, \mathbf{M} * \mathbf{K})$. The difference between two matrix is defined as follows:

$$error(\mathbf{M}, \mathbf{M}') = \sum_{r=1}^{m} \sum_{c=1}^{n} \left| \mathbf{M}_{r,c} - \mathbf{M}'_{r,c} \right| \tag{3}$$

Framework overview. Pseudo-code for the framework is provided in Algorithm 1. Within the algorithm, we use a unique kernel $\mathbf{K}_c$ as evaluation criterion and a sequence of kernels κ for the convolution/threshold process. In the most simple setting, $|\kappa| = 1$. *Thresholding* and *smoothing* are steps that can be enabled or disabled. At the end of each iteration, the quality of the new ordered matrix is evaluated using $\mathbf{K}_c$. If the order of this candidate is improved compared to the previous one, the execution continues with it. If no improvement is found, we can continue the process with the next kernel in κ; this is useful for instance to repeat the process with a more pronounced blur. The execution ends eventually if all kernels fail to improve the last solution. The order of κ is not modified during the execution for the sake of simplicity and to make the results more predictable.

4 Experiments

In this section, we will evaluate the performance of the framework as well as the quality of the results produced for different types of datasets. The experiments will be guided by the following questions:

Q1 How does the iterative framework improve existing methods in terms of quality, for tabular data (Q1a) and for sparse network data (Q1b)?
Q2 How does the iterative framework compare to existing methods with respect to execution time?
Q3 What is the impact of noise on the ordering?
Q4 How well does the smoothing refinement improve the results?
Q5 For which type of data is thresholding necessary?

4.1 Experimental setup

The framework is configured as follows. We have defined a sequence of kernels (κ) containing linear kernels of size $n \times n$ with respectively $n \in \{3, 5, 7, 9, 15, 25\}$. A linear kernel of size 49×49 is considered as evaluation criterion K_c. We have set the maximum number of iterations at 50. Except if specified otherwise, *smoothing* is enabled for all our experiments, *thresholding* is active for tabular data and inactive for network data.

Any matrix ordering method can be embedded into the framework. We made the choice to benchmark different methods belonging to the *R seriation package* [8] as distance-based methods. We have selected the following methods: TSP, OLO, GW, MDS, Spectral, QAP, HC. The barycentric method [7] will also be evaluated as well as the ConvoMap algorithm [1]. These methods are integrated into the framework in order to evaluate how our approach improves the quality of the order.

4.2 Datasets

The experiments were conducted on 2 types of data:

Tabular data: We consider 4 dense artificial datasets, each featuring a different pattern: *Pareto*, *Banded*, *Blocks* and *Triangles*. These datasets are generated for a size of 300×300. Noise is then added to these matrices by flipping each bit with a probability of p. We also evaluate the framework on real world data. *Mammals* [5] is a dataset that records the presence of mammals species (124 columns) at locations across Europe (2200 rows). A cell in the table is positive if a particular species has been seen at a specific location.

Networks: here we consider 4 artificial and 4 real world datasets that represents sparse networks. We consider 4 types of popular structures as artificial data: *Assortative*, *Core-periphery*, *Ordered* and *Disassortative* as presented by [9]. The specificity of these matrices is that they are partitioned in clusters or communities. This ground-truth information will be used for the evaluation of the different methods. In addition, we will evaluate the framework on real world data: (1) *political blog networks* for the US 2004 elections where the communities (ground-truth) are known [10]. In this work, we treated these graphs as undirected graphs, and removed nodes with less than 5 connections. (2) *A mail network* [11] between email addresses of a large European institution. The dataset originally contains 42 departments, but has been preprocessed to remove nodes with less than 5 connections and keep the 6 biggest communities. (3) *Indian village data* [12] representing an Indian village. The relationships inside the matrix represent the interaction links between different households. The caste of these households is known and is used as ground-truth information. (4) Finally we use *news data* [13] corresponding to the adjacency matrix of news documents divided into 3 categories. The links model the similarity between these documents. We applied a threshold value of 0.05. In all these cases, we simplified the data, as the visualization would otherwise be too large.

4.3 Results

We answer ***Q1a*** by comparing the order produced by the conventional methods (*basic*) and its embedded version in our *iterative* framework.

We benchmark all the methods listed above for the four dense datasets with a noise level $p = 0.2$. Before reordering the matrices, the columns and rows are shuffled. Evaluating the quality of an order is not straightforward, as a number of different orders could yield images of similar quality as an image created for the original order of the data (including, for instance, flipping the image horizontally or vertically). As a proxy of the quality of an image, we choose to use the ConvoMap evaluation criterion, as this was shown in previous work to be a reasonably proxy for orders of good quality [1]. The obtained scores are summarized in Table 1, where we also show the score of the original, unshuffled matrix as a reference to facilitate the comparison. Table 2 shows the produced images for some of these methods, allowing to also visually verify that the evaluation score used is a reasonable proxy for the quality of an image.

Table 1: Error scores comparison. The scores better than the original order are highlighted. The best score for a given dataset is indicated in **bold**

Method	Pareto 20%		Banded 20%		Blocks 20%		Triangles 20%	
	basic	iterative	basic	iterative	basic	iterative	basic	iterative
original	30,123		30,839		31,732		31,213	
ConvoMap	30,174	-	**30,759**	-	31,151	-	31,259	-
barycentric	36,087	30,168	33,549	30,819	33,851	31,057	32,908	31,711
TSP	32,469	30,155	34,076	30,831	32,814	31,040	32,313	30,650
HC	32,823	31,403	34,636	33,309	32,563	31,775	32,926	31,900
GW	31,258	30,115	32,686	30,822	31,758	31,145	32,272	30,753
MDS	30,690	30,118	35,698	35,698	34,475	34,465	32,721	32,079
MDS angle	30,898	30,870	31,079	30,817	31,673	**31,006**	33,509	31,523
Spectral	30,344	30,091	34,765	32,502	35,932	33,437	35,587	31,437
QAP 2SUM	30,298	30,116	34,553	31,386	34,338	32,533	35,834	33,057
QAP BAR	30,162	**30,077**	31,511	30,775	32,029	31,691	30,735	**30,500**
OLO	31,275	30,115	32,081	30,820	31,439	31,420	31,951	30,743

Since the framework uses the *basic* solution as a starting point/initial solution in order to enhance it, it is expected that the score of the *iterative* version is systematically better. We observe that the *basic* method is unable to produce a good order (only pieces of *embryonic patterns* that are difficult to interpret). However, *iterative* is able to recover the original pattern for almost all methods and datasets; the ConvoMap score of the solution that is found by the iterative method is in fact slightly better than the score of the original data in more than 50% of the cases. It should be noted that a method is only able to recover the original pattern if it is able to do so on noiseless matrices. The hierachical clustering (HC) approach is not able to structure a linear pattern like in the banded dataset, so it cannot be enhanced by the framework.

We also compare the results of the framework with the ConvoMap algorithm in Table 3. The produced ordered matrices are similar. This is not surprising since both rely on convolution. However, our framework produces these results in a much shorter time, several orders of magnitude shorter. This is explained by the fact that ConvoMap uses the evaluation criterion as a scoring function through a local search algorithm that tries to optimize this criterion through random moves. In our framework we use faster methods, i.e. the base methods, repeatedly on simplified noise-free matrices.

We show the results for the mammals dataset in Figure 3. We use two *structure-based* algorithms (nested and barycentric). As expected the nested method gives the best results: it is known that in warmer areas the diversity of mammals increases compared to colder areas. This nested pattern is however more clearly visible in the output of the iterative method. Moreover, in the output of the iterative method we also notice a cluster of species outside the nested structure, occurring in areas where the number of other species is relatively

Table 2: Comparative results between the original algorithm and its embedded version into the iterative approach

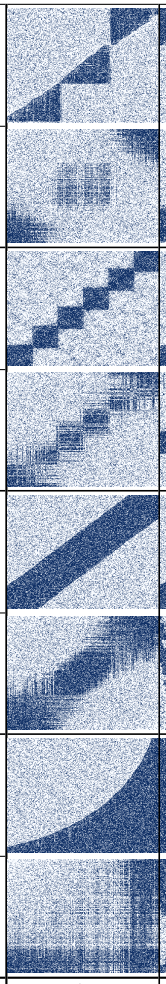
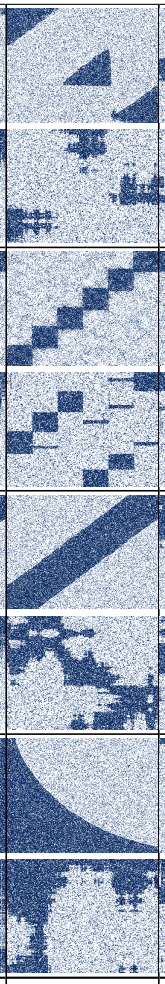
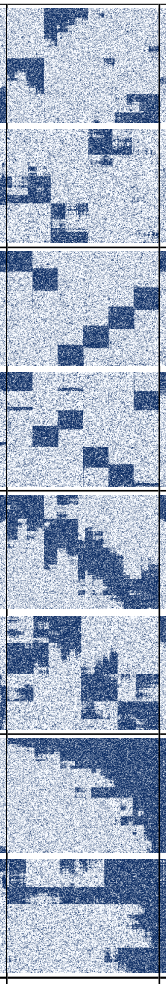
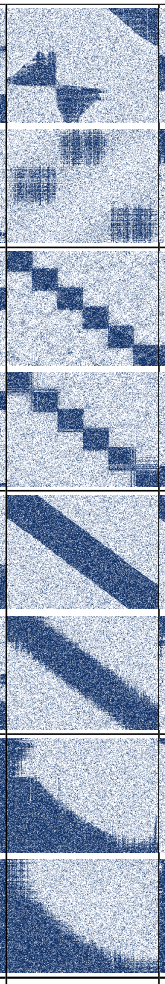
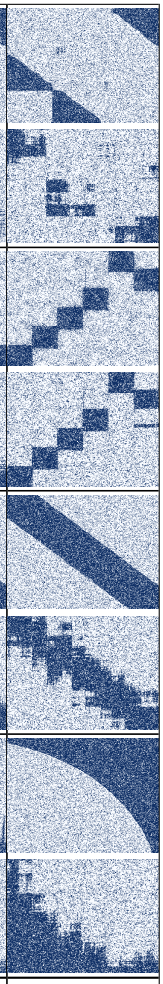
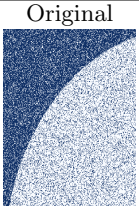
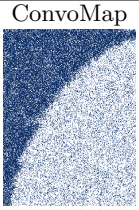
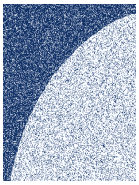
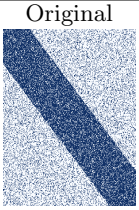
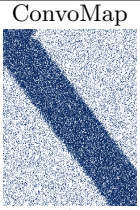
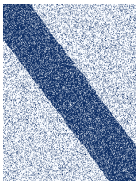
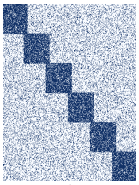
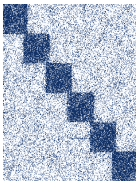
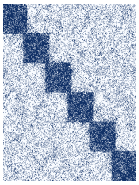
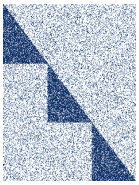
Methods	Pareto		Banded		Blocks		Triangles	
	basic	iterative	basic	iterative	basic	iterative	basic	iterative
Barycentric								
TSP								
HC								
MDS angle								
GW								

Table 3: Comparative results for ConvoMap and iterated (TSP) algorithm

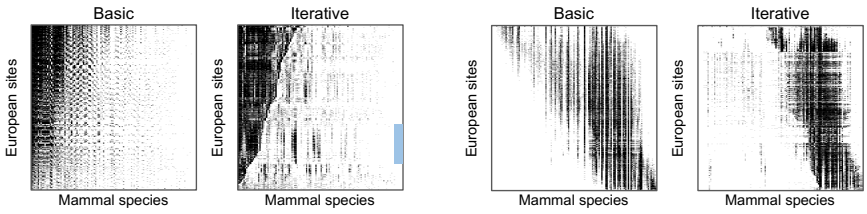
Pareto			Banded		
Original	ConvoMap	Iterative	Original	ConvoMap	Iterative
					
30 123	30 174 20 841 s	30 155 98 s	30 839	**30 759** 22 754 s	30 831 109 s
Blocks			Triangles		
Original	ConvoMap	Iterative	Original	ConvoMap	Iterative
					
31 732	31 151 40 489 s	**31 040** 75 s	31 213	31 259 18 551 s	**30 650** 148 s

small. This cluster corresponds to species that are almost exclusively located in northern Europe (Scandinavia, blue label) (*Gulo-Gulo*, *Alces-Alces*, ...). This useful information is not visible in the output of the original methods.

We address ***Q1b*** by applying the same experimental process to the network data described above. The quality of the ordering will be judged (1) subjectively by looking at the image produced (2) with an accuracy score. Each row/node of the adjacency matrix is associated with a label corresponding to the community. The better the ordering, the more likely these labels will be ordered together. To calculate the accuracy, we determine for each position the most common label among the *10-nearest* rows, if this majority label corresponds to the label of the row it is a *match*. The ratio $\frac{maches}{nodes}$ corresponds to the accuracy score.

Table 4 summarizes the accuracy scores for the different communities by comparing the basic and iterated version. We observe that almost all methods are improved with respect to the accuracy score on both synthetic and real data. Figure 4a to 4d show the produced matrices for synthetic networks with the labels of the different communities. We use HC (hierarchical clustering) as base method but other methods produce visually comparable results. The experiment show that that the *basic* version is not effective, but the iterative method identifies the clusters accurately.

Figure 4e to 4h show the results for real data. The base methods used are those that give the best results for their iterated version. Once more the iterated method correctly partitions the matrix into the corresponding clusters, in contract to the



(a) Nested ordering (b) Barycentric ordering

Fig. 3: European mammals dataset: comparison between basic and iterative

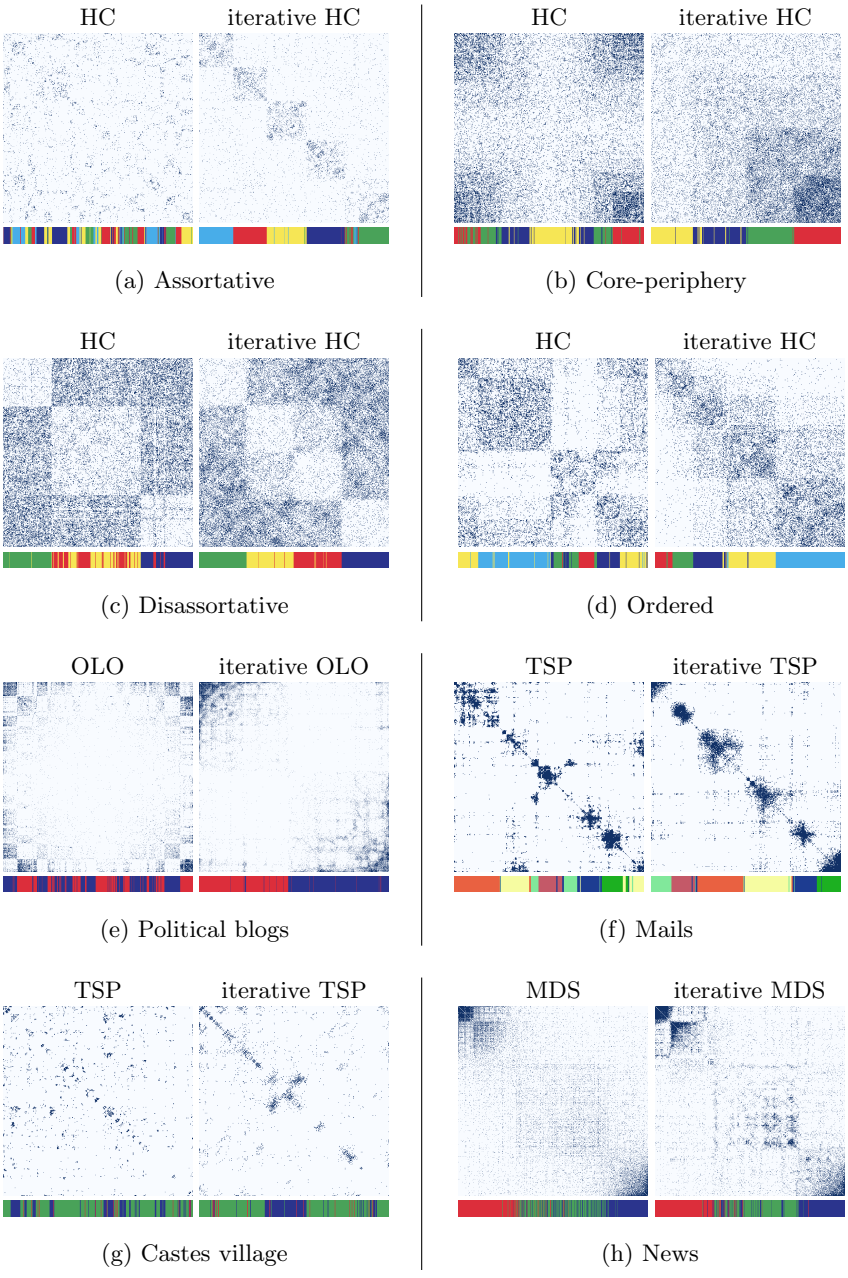
base method. Another feature is that the granularity of the noise is significantly smoothed.

To address ***Q2***, we already show in Table 3 the computational time for different datasets. Experiments show that the type of data (sparse, noise) has little impact and that it is essentially (*1*) the base method used (*2*) the size and number of kernels used for the convolution that are the most determining factors. The execution time is intrinsically linked to the number of iterations (Fig 5) and to the thresholding and smoothing. The methods that have been benchmarked (from the R seriation library) are optimized and have more or less similar times so that the average execution time of the framework for data of medium size (300×300) rarely exceeds 3 minutes of computation time. It should be noted that the more the execution progresses, the smaller the improvement gain will be. Approaches for early stopping could potentially provide shorter execution times.

Table 5 compares the same dataset (band) for different levels of noise ($p = 0.1, 0.2, 0.3, 0.35$). The objective of this experiment is to answer questions ***Q3*** and ***Q4***. The method studied is the *barycentric* one. The first row of the table corresponds to the results produced by the *basic* version. The second row is the iterative method but without the smoothing enable. As expected, the stronger

Table 4: Experiments for synthetic and real network reordering: community labels (k) accuracy assessment between the basic (b) and iterated version (i)

Dataset	n	k	HC		OLO		GW		MDS		TSP	
			b	i	b	i	b	i	b	i	b	i
a) Artificial data												
Assortative	500	5	81.2	**93.8**	84.6	**96.2**	50.2	**95.0**	64.2	**93.4**	87.0	**94.4**
Core-periphery	400	4	86.2	**95.2**	86.5	**94.5**	93.2	**96.7**	95.2	**95.7**	65.7	**89.5**
Dissasortative	400	4	89.0	**96.5**	90.7	**97.7**	81.2	**97.5**	85.0	**96.5**	87.5	**96.5**
Ordered	350	5	90.8	**95.1**	93.1	**98.8**	90.0	**98.0**	88.2	**96.2**	84.6	**94.5**
b) Real data												
Political blogs [10]	852	2	94.1	**95.5**	90.2	**97.2**	92.2	**96.1**	**97.3**	96.5	95.8	**96.4**
EU-emails [11]	329	6	78.7	**87.2**	89.0	**92.7**	82.7	**84.0**	63.2	**91.7**	90.9	**94.2**
Village castes [12]	356	3	81.7	**85.1**	82.8	**86.2**	74.4	**87.1**	**87.9**	87.6	82.0	**87.9**
News [13]	600	3	83.7	**87.2**	85.7	**88.7**	79.8	**88.2**	81.3	**88.3**	**89.0**	87.1



(a) Assortative (b) Core-periphery

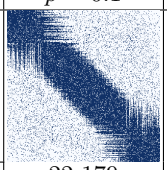
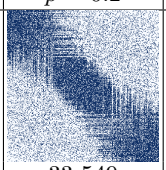
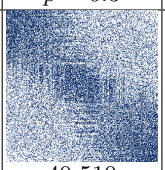
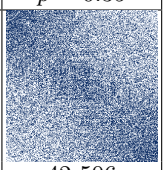
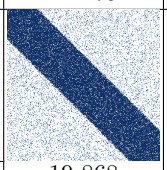
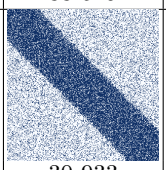
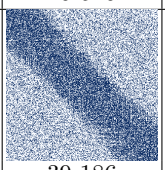
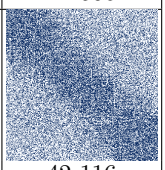
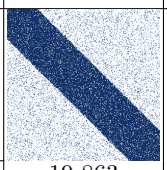
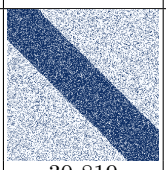
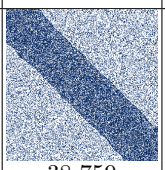
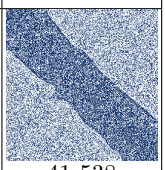
(c) Disassortative (d) Ordered

(e) Political blogs (f) Mails

(g) Castes village (h) News

Fig. 4: Results for synthetic and real networks with ground truth community labels

Table 5: Noise variation for barycentric methods on the banded dataset

	$p = 0.1$	$p = 0.2$	$p = 0.3$	$p = 0.35$
barycentric				
criterion	22 179	33 549	40 510	42 506
iterative simple				
criterion	19 868	30 933	39 186	42 116
iterative smooth				
criterion	19 863	30 819	38 759	41 538

the noise, the worse the ordering produced by the *basic* version is. On the other hand, the framework is able to handle noise up to 30-35%. The advantage of the smoothing refinement is to slightly adjust the order or rows and columns by minimizing the difference between the matrix and its convoluted or threshold. The impact of this refinement is more marked when the noise is high.

To further investigate ***Q4***, we studied in Figure 5 the impact of the smoothing refinement on the execution time and the number of runs of the base algorithm. The experiments were conducted on a 25% band dataset for different methods. A key finding is that the smoothing refinement has not only an impact on the quality but also on the speed of the algorithm. The solution converges much faster, which improves the execution speed by about 50%.

Finally, we study in Figure 6 the impact of *thresholding* and *smoothing* on dense tables and sparse networks. The interest of this experiment is to answer ***Q5*** and to determine when *thresholding* is necessary. The results show that this refinement is recommended for tabular data, while it is more efficient to reorder the blurred matrix directly for networks. The intuition is that *thresholding* only coarsens the dots for sparse data while it allows better pattern extraction and noise reduction when the data is dense. The choice of using *thresholding* is therefore not motivated by the data type, but by the sparsity.

5 Conclusions and future work

We have introduced a new framework for boosting the quality of the output produced by matrix reordering or seriation methods, for a large variety of datasets,

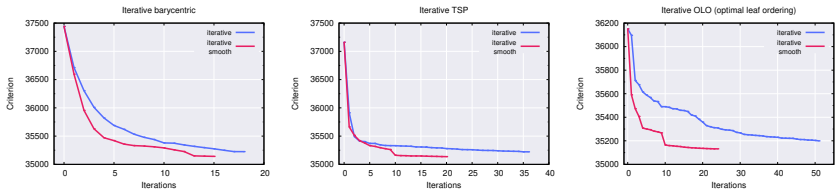
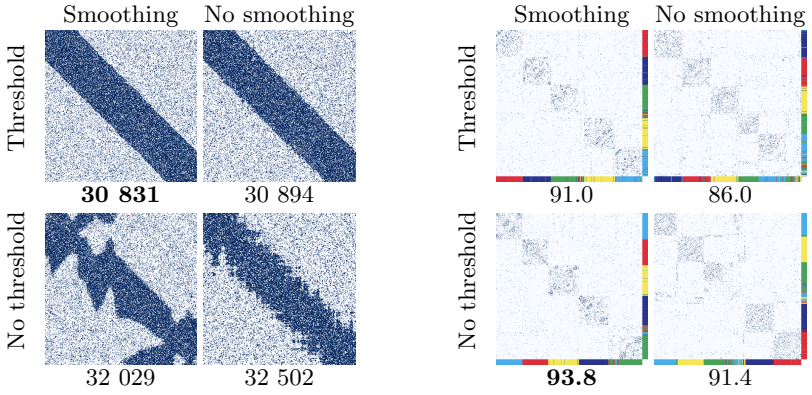


Fig. 5: Smoothing impact on number of iterations for a band dataset ($p = 0.25$)

and in particular those that are noisy and sparse. The key idea is to blur the finer structure in a dataset away, hence allowing the reordering to focus on the high level structure; the blurring is progressively reduced in successive iterations. Our results on a wide range of datasets show that the method outperforms existing methods both in terms of the quality of the output and in terms of the time necessary to find the ordering.

Many adjustments and improvements can nevertheless be added to the framework. For example, we have chosen to use a single base embedded method, but a variant would be to consider a set of different methods that would be applied in parallel in order to diversify the order produced; we could also apply a different method for the initial ordering.

We choose a fixed the set of kernels used in this work. An alternative would be to generate the kernels automatically by analyzing the noisiness or sparsity of the data. We can also consider a more refined method of kernel replacement strategy, by reordering the κ set (for example by penalizing kernels that failed to improve the order).



(a) Tables (b) Networks

Fig. 6: Impact of thresholding and smoothing. TSP method used on (a) with criterion and hierarchical clustering on (b) with label accuracy

We have seen the base method as a black box that can be easily replaced, but the same principle could be applied for the evaluation criterion component, which we fixed to the ConvoMap score in this framework.

The proposed framework is currently focused on binary data. However, we believe it is robust enough to be easily adapted to numerical data (biological datasets, gene expression) or a mix of categorical and numerical data.

Finally, while we focused on seriation as a step in the creation of black/white images for data, a more extensive evaluation could be carried out to determine whether this form of iterative loop is also useful when solving clustering or classification tasks. for instance, in combination with deep learning techniques also based on convolution.

References

1. Bollen, T., Leurquin, G., Nijssen, S. (2018). ConvoMap: Using Convolution to Order Boolean Data. *17th International Symposium, IDA 2018.*
2. Behrisch, M., Bach, B., Henry Riche, N., Schreck, T., Fekete, J-D. (2016). Matrix Reordering Methods for Table and Network Visualization. *Computer Graphics Forum.* 35. 10.1111/cgf.12935.
3. Garriga, G. Junttila, E., Mannila, H. (2008). Banded structure in binary matrices. *Knowledge and Information Systems.* 28. 197-226.
4. Hubert, L.J. Some applications of graph theory and related nonmetric techniques to problems of approximate seriation: The case of symmetric proximity measures. *British Journal of Mathematical Statistics and Psychology*, 27:133–153, 1974.
5. Mitchell-Jones, A. J., Mitchell, J., Amori, G., Bogdanowicz, W., Spitzen- berger, F., Krystufek, B., Vohralik, V., Thissen, J., Reijnders, P., Ziman, J., et al. (1999). The atlas of European mammals, volume 3. *Academic Press London.*
6. Sezgin, M. and Sankur, B. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging.* 2004.
7. Makinen, E. and Siirtola, H. (2005). The barycenter heuristic and the reorderable matrix. *Informatica (Slovenia)*, 29(3):357–364.
8. Hahsler, M., Buchta, C., Hornik, K. (2020). Seriation: Infrastructure for Ordering Objects Using Seriation. R package version 1.2-9.
9. Faskowitz, J., Yan, X., Zuo, XN. et al. Weighted Stochastic Block Models of the Human Connectome across the Life Span. *Sci Rep 8*, 12997 (2018).
10. Adamic, L.A. Glance, N. The political Blogosphere and the 2004 U.S. election: divided they blog. *in Proceedings of the 3rd international workshop on Link discovery*, LinkKDD '05 (ACM, New York, NY, USA, 2005) pp. 36–43
11. Yin, H., Benson, A.R., Leskovec, J., Gleich, D.F. Local Higher-order Graph Clustering. *In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017.*
12. Banerjee, A., Chandrasekhar, A.G., Duflo, E., Jackson, M.O., 2013, The Diffusion of Microfinance, https://doi.org/10.7910/DVN/U3BIHX, Harvard Dataverse, V9.
13. Ding, C.H., He, X., Zha, H., Gu, M., Simon, H.D. (2001). A min-max cut algorithm for graph partitioning and data clustering. *Proceedings IEEE International Conference on Data Mining (ICDM), 2001*, pages 107–114. IEEE.