Parallel implementation of Spectral Clustering

Sandrine Mouysset[1] and Ronan Guivarch[2]

[1] University of Toulouse, IRIT-UPS, France
[2] University of Toulouse, INP(ENSEEIHT)-IRIT, France

Abstract. Test
Spectral clustering is one of the most relevant unsupervised method able to gather data without a priori information on shapes or locality. A parallel strategy based on domain decomposition with overlapping interface is considered. By investigating sparsification techniques and introducing sparse structures, this parallel method is adapted to treat very large data set in fields of Pattern Recognition and Image Segmentation.

1 Introduction

Spectral clustering selects dominant eigenvectors of a parametrized affinity matrix in order to build a low-dimensional data space wherein data points are grouped into clusters [1]. The main difficulties of this method could be summarized by the two following questions: how to automatically separate clusters one from the other and how to perform clustering on large dataset for example on image segmentation. This means that we look for some full-unsupervising process with parallelization. Several studies exist for defining a parallel implementation which exploits linear algebra [3], [4] for the affinity computation of the whole data set [2]. But the input parameters which are the affinity parameter and the number of clusters limit these methods. To address this limitation, a fully unsupervised parallel strategy based on domain decomposition was proposed in [5] which preserves the quality of global partition thanks to overlapping interface. From the first results, we have observed that the main part of the time is spent in the spectral clustering step and we encountered memory limitation with large problems.

In this paper, we study the robustness of this parallel spectral clustering by investigating sparsification techniques and introducing sparse structures and adapted eigensolvers in order to treat larger problems.

2 Parallel Spectral Clustering

Let's consider a data set $S = \{x_i\}_{i=1..n} \in \mathbb{R}^p$. Assume that the number of targeted clusters k is known. First, the spectral clustering consists in constructing the affinity matrix based on the Gaussian affinity measure between points of the dataset S. After a normalization step, the k largest eigenvectors are extracted. So every data point x_i is plotted in a spectral embedding space of $\mathbb{R}^k$ and the clustering is made in this space by applying $K-means$ method. Finally, thanks to an equivalence relation, the final partition of data set is defined from the clustering in the embedded space. Algorithm 1 presents the different steps of spectral clustering.

Algorithm 1 Spectral Clustering Algorithm

```
Input: data set $S$, number of clusters $k$
1. Form the affinity matrix $A \in \mathbb{R}^{n \times n}$ defined by:
   $A_{ij} = \begin{cases} \exp\left(-\frac{\|x_i - x_j\|^2}{(\sigma/2)^2}\right) & \text{if } i \neq j, \\ 0 \text{ otherwise}, \end{cases}$   (1)
2. Construct the normalized matrix: $L = D^{-1/2} A D^{-1/2}$ with $D_{i,i} = \sum_{j=1}^{n} A_{ij}$,
3. Assemble the matrix $X = [X_1 X_2 .. X_k] \in \mathbb{R}^{n \times k}$ by stacking the eigenvectors associated with the $k$ largest eigenvalues of $L$,
4. Form the matrix Y by normalizing each row in the $n \times k$ matrix X,
5. Treat each row of Y as a point in $\mathbb{R}^k$, and group them in $k$ clusters via the $K$-means method,
6. Assign the original point $x_i$ to cluster $j$ when row $i$ of matrix Y belongs to cluster $j$.
```

This spectral clustering method could be adapted for parallel implementation (see Figure 1). By dividing the data set S in q sub-domains, each processor applies independently the spectral clustering algorithm on the subsets and provide a local partition. Based on these local partitions and overlapping interface, a gathering step ensures the connection between subsets of data and determines a global partition. For each subdomain, a quality measure which exploits the block structure of indexed affinity matrix per cluster is used to determine the number of clusters.

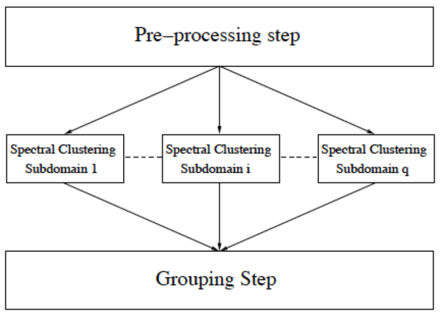

Pre–processing step
Spectral Clustering Subdomain 1
Spectral Clustering Subdomain i
Spectral Clustering Subdomain q
Grouping Step

Fig. 1. Principle of the parallel spectral clustering

We can summarize this Master-Slave implementation with Algorithm 2 and Algorithm 3.

Algorithm 2 Parallel Algorithm: Master

```
1: Pre-processing step
   1.1 Read the global data and the parameters
   1.2 Split the data into $q$ subsets
   1.3 Compute the affinity parameter $\sigma$ with the formula given in paper [5];
       the bandwidth of the overlapping is fixed to $3 \times \sigma$
2: Send the sigma value and the data subsets to the other processors (MPI_SEND)
3: Perform the Spectral Clustering Algorithm on its subset
   3.1 Computation of the spectrum of the affinity matrix (1): classical routines
       from LAPACK library [6] are used to compute selected eigenvalues,
       eigenvectors of the normalized affinity matrix $A$ for its subset of data points
   3.2 Number of clusters: the number of clusters $k$ with the heuristic [5]
   3.3 Spectral embedding: the centers for K-means initialization in the spectral
       embedding are chosen to be the furthest from each other along a direction
4: Receive the local partitions and the number of clusters from each processor
   (MPI_RECV)
5: Grouping Step
   5.1 Gather the local partitions in a global partition thanks to the transitive relation
   given in paper [5]
   5.2 Output a partition of the whole data set $S$ and the final number of clusters $k$
```

Algorithm 3 Parallel Algorithm: Slave

```
1: Receive the sigma value and its data subset from the Master processor (MPI CALL)
2: Perform the Spectral Clustering Algorithm on its subset
3: Send the local partition and its number of clusters to the Master processor (MPI
   CALL)
```

We can notice that when we split the original data set into overlapping sub-pieces of data set, we gain on two aspects:

– memory consumption: the local spectral clustering analysis of each sub-piece involves the creation of a local affinity matrix. The size of the matrix is n^2, n being the cardinal of the data subset. The sum of the memory needs for all these local affinity matrix is much less than that needed for the affinity matrix covering the global data set. The consequence is that we can manage bigger data set, data set whose size cannot permit us to run with only one processor.
– floating point operations: the analysis of each subproblem is made from the extraction of eigenvectors in the scaled affinity sub-matrix: one extracted eigenvector for each identified cluster of the data subset. In that respect, the

parallel approach enables us to decrease drastically the cost of this eigenvector computation: each subproblem will include a number of clusters much less than the total number of clusters in the whole data set.

Nevertheless, as we want to be able to consider larger and larger data sets, as, for instance, in image segmentation (see 3.3) or genomic applications, we still encounter memory limitation when the number of points in a local data subset is too much for the memory capacity of one processor.

3 Sparsification of Spectral Clustering

Despite the domain decomposition, the most time consuming is dedicated to the spectral clustering algorithm. To address this limitation and the memory consumption ones, we investigate a thresholding as sparsification technique.

3.1 Thresholding

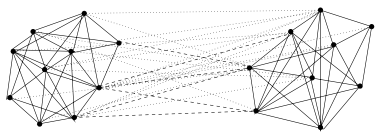
From the definitions of both the Gaussian affinity A_{ij} between two data points x_i and x_j and the Heat kernel $K_t(x) = (4\pi t)^{-\frac{p}{2}} \exp\left(-\|x\|^2/4t\right)$ in free space $\mathbb{R}_+^* \times \mathbb{R}^p$, we can interpret the gaussian affinity matrix as discretizations of heat kernel by the following equation:

$$A_{ij} = (2\pi\sigma^2)^{\frac{p}{2}} K_t\left(\sigma^2/2, x_i - x_j\right). \qquad (2)$$

So, we can prove that eigenfunctions for bounded and free space Heat equation are asymptotically close [7]. With Finite Elements theory, we can also prove that the difference between eigenvectors of A and discretized eigenfunctions of K_t is of an order of the distance between points include inside the same cluster. This means that applying spectral clustering into subdomains resumes in restricting the support of these L^2 eigenfunctions which have a geometrical property: their supports are included in only one connected component. In fact, the domain decomposition by overlapping interface does not alter the global partition because the eigenvectors carry the geometrical property and so, the clustering property.

Let's now interpret a thresholding of the affinity matrix on the clustering result. This leads to restrict the approximation to the finite elements which satisfy homogeneity mesh condition in the interpretation. In other words, this means that it strengthens the piece-wise constancy of the dominant eigenvectors from the normalized Gaussian affinity matrix. But the threshold should be well-chosen and should be coherent according to the data distribution. So it should be defined function of both dimension of the data and number of data as defined in [7].

From another point of view, the affinity matrix could be also interpreted as a Gaussian weighted adjacency graph. The thresholding will control the width of the neighborhoods. This parameter chosen according to the affinity parameter plays a similar role as the parameter ϵ in case of the ϵ-neighborhood graph. A


(a) Without thresholding

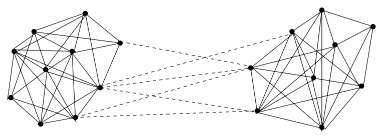
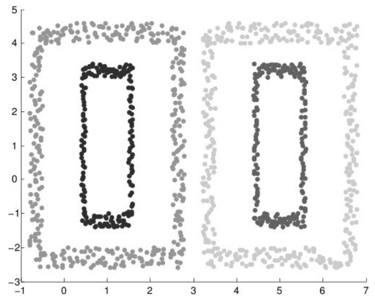
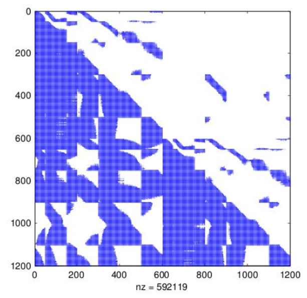

(b) With thresholding

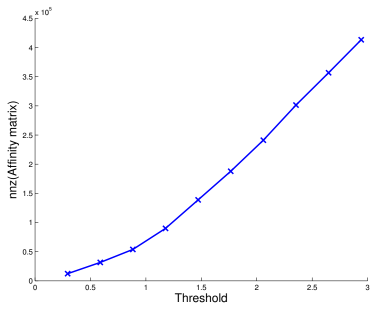
Fig. 2. Thresholding of the weighted adjacency graph.

thresholding of the largest distances is equivalent to cancel edges which connect data points very distant from each other as represented in Figure 2. So it strengthens the affinity between points among the same cluster and the separability between clusters.

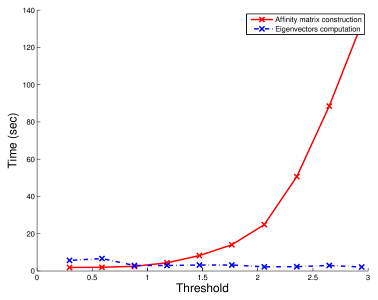

(a) Clustering result with thresholding



(b) Affinity matrix : lower triangular without threshold, upper triangular with threshold



(c) Memory cost function of the threshold

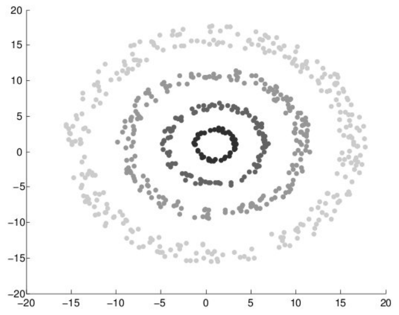


(d) Timings function of the threshold

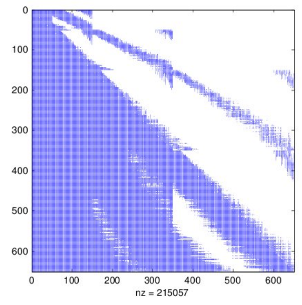
Fig. 3. Example 1: data set, sparsity of the affinity matrix, memory cost and timings

3.2 First validations

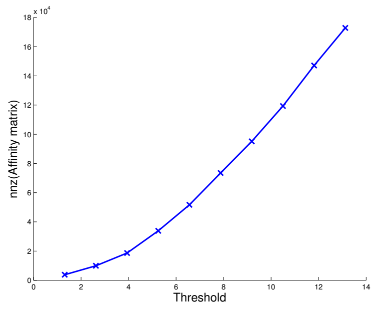
For first validations, we consider two examples represented in Fig. 3 (a) and Fig. 4 (a) in which the clusters could not be separated by hyperplanes: the first one with four rectangles of $n = 1200$ points and a target of $n = 600$ points. The eigenvectors were provided by the reverse communication required by the Fortran library ARPACK [8].



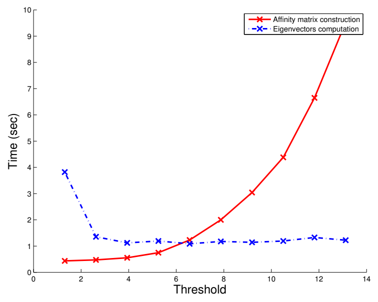
(a) Clustering result with thresholding



(b) Affinity matrix : lower triangular without threshold, upper triangular with threshold



(c) Memory cost function of the threshold



(d) Timings function of the threshold

Fig. 4. Example 2: data set, sparsity of the affinity matrix, memory cost and timings

We measure the timings in seconds, in function of the threshold, of the construction of the affinity matrix and of the computation of eigenvectors. The memory cost is evaluated in function of the threshold by the number of nonzeros elements in the affinity matrix.

We can notice on (c) sub-figures that we gain a lot of memory when we decrease the threshold i.e. when we drop the connections of points at a distance larger than it.

We also remark on (d) sub-figures that the time to construct the affinity matrix decreases in this case. Indeed, the computation of the component A_{ij} requires to compute an exponential (1). So because the selection of the connections we keep is done only with the distance, we don't compute the non-useful components and save a lot of floating point operations.

So we have a response to the memory consumption and timing limitations we mentioned previously.

If we look the timing for the extraction of the eigenvectors, the time remains the same for acceptable values of the threshold. But we encounter a limit to the sparsification technique with example 2: a strong threshold could imply a very sparsified affinity matrix with a bad condition number. In this case, the eigenvector computation becomes the most time consuming task in the sense that the algorithm from Arnoldi method does not converge.

3.3 Another application: image segmentation

For image segmentation, we include both 2D geometrical information and 1D brightness (or 3D color levels) information in the spectral clustering method in the sense that there does not exist some privileged directions with very different magnitudes in the distances between points along theses directions. The step between pixels and brightness (or color levels) are about the same magnitude. Thus, a new distance in the affinity measure is defined for image. In the same way, a global heuristic for the Gaussian affinity parameter is proposed in which both dimension of the problem as well as the density of points in the given 3D (or 5D for colored image) are integrated. By considering the size of the image I, the Gaussian affinity A_{ir} is defined as follows:

$$A_{ir} = \begin{cases} \exp\left(-\frac{d(I_{ij},I_{rs})^2}{(\sigma/2)^2}\right) \text{ if } (ij) \neq (rs), \\ 0 \text{ otherwise,} \end{cases}$$

with the distance between the pixel (ij) and (rs) defined by:

$$d\left(I_{ij}, I_{rs}\right) = \sqrt{\left(\frac{i-r}{l}\right)^2 + \left(\frac{j-s}{m}\right)^2 + \left(\frac{I_{ij}-I_{rs}}{256}\right)^2} \qquad (3)$$

Parallel spectral clustering was used for image segmentation [5] and the sparsified parallel spectral clustering will be applied for further investigations on image segmentation and presented in the final paper.

4 Conclusion and ongoing works

As we mentioned in the conclusion of our work at the previous VECPAR conference [5], we have begun to study sparsification techniques in the construction of affinity matrix by dropping some components that correspond to points at a distance larger than a threshold. We validate this approach in matlab by showing that the number of non zero of the affinity matrix decreases with still some good results in terms of spectral clustering and even some gains in the time spent to compute the affinity matrix.

We have now to implement these sparsification techniques in our parallel FORTRAN codes by using sparse data structure to store the matrix and well-suited numerical libraries to compute the eigenvectors (ARPACK) in order to show the benefits of our approach in the case of real images on parallel computers.

References

1. Ng, A. Y., Jordan, M. I. and Weiss, Y. On spectral clustering: analysis and an algorithm. *Proc.Adv.Neural Info.Processing Systems*, 2002.
2. Chen, W-Y., Yangqiu, S., Bai H., Lin C-J. and Chang E. Y. Parallel Spectral Clustering in Distributed Systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*,2010.
3. Song, Y. and Chen, W.Y. and Bai, H. and Lin, C.J. and Chang, E.Y. Parallel spectral clustering, *Processing of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2008.
4. Fowlkes, C. and Belongie, S. and Chung, F. and Malik, J., Spectral grouping using the Nystrom method, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.
5. Mouysset, S. and Noailles, J., Ruiz, D. and Guivarch, R.. On a strategy for Spectral Clustering with parallel computation, *High Performance Computing for Computational Science: 9th International Conference*, 2010.
6. Anderson, E. and Bai, Z. and Bischof, C. and Blackford, S. and Demmel, J. and Dongarra, J. and Du Croz, J. and Greenbaum, A. and Hammarling, S. and McKenney, A. and others LAPACK Users' guide, *Society for Industrial Mathematics*, 1999.
7. Mouysset, S. and Noailles, J. and Ruiz, D. On an interpretation of Spectral Clustering via Heat equation and Finite Elements theory, *International Conference on Data Mining and Knowledge Engineering*, 2010.
8. Lehoucq, R.B. and Sorensen, D.C. and Yang, C., ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods, *SIAM*,1998.