On a strategy for Spectral Clustering with parallel computation

Sandrine Mouysset, Joseph Noailles, Daniel Ruiz and Ronan Guivarch[1]

University of Toulouse, IRIT-ENSEEIHT, France
{sandrine.mouysset,jnoaille,daniel.ruiz,ronan.guivarch}@enseeiht.fr

Abstract. Spectral Clustering is one of the most important method based on space dimension reduction used in Pattern Recognition. This method consists in selecting dominant eigenvectors of a matrix called affinity matrix in order to define a low-dimensional data space in which data points are easy to cluster. By exploiting properties of Spectral Clustering, we propose a method where we apply independently the algorithm on particular subdomains and gather the results to determine a global partition. Additionally, with a criterion for determining the number of clusters, the domain decomposition strategy for parallel spectral clustering is robust and efficient.

1 Introduction

Clustering aims to partition a data set by grouping similar elements into subsets. Two general main issues concern, on the one hand, the choice of a similarity criterion and, on the other hand, the way to separate clusters the one from the other. Spectral methods, and in particular the spectral clustering algorithm introduced by Ng-Jordan-Weiss (NJW) [1], are useful when considering non-convex shaped subsets of points. These methods are widely used in Pattern Recognition and in particular in Bioinformatics and image segmentation. The number of targeted clusters k is usually assumed to be known. From the spectral elements of an affinity normalized matrix, data points are clustered in a low-dimensionnal space made by the first eigenvectors of the normalized affinity matrix. Several approaches about parallel Spectral Clustering [5], [6], [2] were recently suggested, mainly focused on linear algebra techniques to reduce computational costs. However, the authors do not get rid of the construction of the complete affinity matrix and the problem of determining the number of clusters is still open.

In this paper, we propose to cluster on subdomains by breaking up the data set into data subsets with respect to their geometrical coordinates in a straighforward way. With an appropriate Gaussian affinity parameter and a method to determine the number of clusters, each processor applies independently the spectral clustering algorithm on subsets of data points and provide a local partition on these data subsets. Based on these local partitions, a gathering step ensures

[0] Candidate to the Best Student Paper Award

the connection between subsets of data and determines a global partition. We analyze in particular two different approaches of the type and we experiment on a geometrical particular example and on an image segmentation example. We identify the potential for parallelism of the algorithm as well as numerical behaviour and limitations.

2 Parallel Spectral Clustering: algorithm and justification

Spectral clustering uses eigenvectors of a matrix, called Gaussian affinity matrix, in order to define a low-dimensional space in which data points can be clustered (see algorithm 1).

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:

$$A_{ij} = \begin{cases} \exp\left(-\frac{\|x_i - x_j\|^2}{\sigma/2}\right) & \text{if } i \neq j, \\ 0 \text{ otherwise,} \end{cases} \tag{1}$$

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.

The Gaussian affinity matrix defined by (1) could be interpreted as a discretization of the Heat kernel [3]. And in particular, it is shown in [8] that this matrix is a discrete representation of the L^2 heat operator onto appropriate connected domains in $\mathbb{R}^p$. Thanks to properties of the heat equation, eigenvectors of this matrix are an asymptotical discrete representation of L^2 eigenfunctions with support included in only one connected component.

Clustering in subdomains resumes in restricting the support of these L^2 particular eigenfunctions. Therefore, we can apply Spectral Clustering on subdomains to identify connected components. The subdomains can be defined in a straightforward way by subdividing original data set according to their geometrical coordinates and a partition can be extracted independently and in parallel from each subset. Then, at the grouping level, spectral clustering algorithm is made on a subset with geometrical coordinates close to the boundaries of the previous subdomains. This partitionning will connect together clusters which belong to different subdomains thanks to the transitive relation: $\forall x_{i_1}, x_{i_2}, x_{i_3} \in S$,

$$\text{if } x_{i_1}, x_{i_2} \in C^1 \text{ and } x_{i_2}, x_{i_3} \in C^2 \text{ then } C^1 \cup C^2 = P \text{ and } x_{i_1}, x_{i_2}, x_{i_3} \in P \quad (2)$$

where S is a data set, C^1 and C^2 two distinct clusters and P a larger cluster which includes both C^1 and C^2.

Two main problems arise from this divide and conquer strategy: the difficulty to choose a Gaussian affinity parameter σ and the number of clusters k which remains unknown and may even vary from one subdomain to the other. We propose two ways to overcome these drawbacks. In the following, let us consider a p-dimensional data set $S = \{x_1, .., x_n\} \subset \mathbb{R}^p$. In the next section, we shall address the proper choice of the parameter σ and in section 2.2, we propose a way to overcome the problem of not knowing the number of clusters a priori.

2.1 Choice of the affinity parameter σ

The Gaussian affinity matrix (1) is widely used and depends on a free parameter σ. It is known that this parameter affects the results in spectral clustering and spectral embedding. A global heuristic for this parameter was proposed in [4] in which both the dimension of the problem as well as the density of points in the given p-th dimensional data set are integrated. With an assumption that the p-dimensionnal data set is isotropic enough, the data set S is included in a p-dimensionnal box bounded by D_{max} the largest distance between pairs of points in S: $D_{\max} = \max_{1 \leq i,j \leq n} \|x_i - x_j\|$.
So a reference distance noted σ could be defined: this distance represents the case of an uniform distribution in the sense that all pair of points are separated by the same distance σ in the box of edge size D_{max}:

$$\sigma = \frac{D_{\max}}{n^{\frac{1}{p}}}. \quad (3)$$

From this definition, clusters may exist if there are points that are at a distance no more than a fraction of σ. We could define such parameter for each subdomain. However, with a straighforward decomposition as the one proposed, one can find easily that a local σ in each subdomain will be close to the value of a global σ defined on the whole data set in the same way. This avoids local computations. However, this is not the case for the interface data set where a local σ must be considered. To conclude, we only need to compute two values of σ: one for the interface where the topology of the volume changes drastically, and one common to all the other "cubic" subdomains.

2.2 Number of clusters k

The problematic of the right choice of k is all the more accurate that this number may vary from one subdomain to the other in such a domain decomposition strategy. We therefore consider in each subdomain a quality measure based on

ratios of Frobenius norms, see for instance [4]. For instance, after indexing data points by cluster as followed, for $k = 3$:

$$\hat{L} = \begin{bmatrix} L^{(11)} & L^{(12)} & L^{(13)} \\ L^{(21)} & L^{(22)} & L^{(23)} \\ L^{(31)} & L^{(32)} & L^{(33)} \end{bmatrix},$$

the off-diagonal blocks will represent the affinity between clusters and the diagonal ones the affinity within clusters. The ratios between the Frobenius norm of the off-diagonal blocks and that of the diagonal ones could be evaluated:

$$r_{ij} = \frac{\|L^{(ij)}\|_F}{\|L^{(ii)}\|_F}.$$

By definition, the appropriate number of clusters k corresponds to a situation where points which belong to different clusters have low affinity between each other whereas points in same clusters have higher affinity. Among various values for k, the final number of cluster is defined so that the affinity between clusters is the lowest and the affinity within clusters is the highest as followed:

$$k = argmin \sum_{i \neq j} r_{ij}. \tag{4}$$

In Fig. 1, the ratio $\eta = \sum_{i \neq j} r_{ij}$, function of the number of cluster k, is plotted on two examples with various densities among clusters. The gap observed on the ratio η between two consecutive values of k indicates a strong change in the average links between clusters.

Moreoever, dividing the whole data set in subdomains may lead to situations in which a subdomain contains only one cluster. If the number of clusters k which satisfy (4) is equal to 2 in one subdomain, we then compare the numerator of ratio η to its denominator. Based on a threshold β, if the ratio $\frac{\|L_{12}\|_F}{\|L_{11}\|_F}$ is larger than β, we set the value k to 1 instead of 2.

3 Implementation: Algorithm components

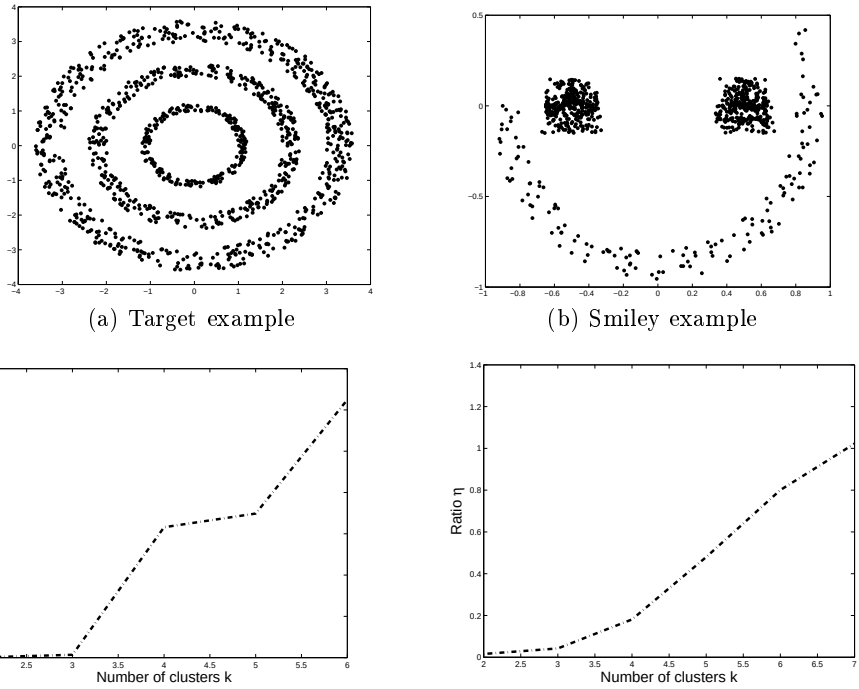
We shall now detail the different steps, described in Fig. 2, of the algorithm with respect to the strategy proposed previously.

3.1 Pre-processing step: Partition S in q subdomains

Let us include all data points in a box of edge l_i for the ith-dimension, $i = \{1, .., p\}$ where:

$$l_i = max_{1 < i_1, i_2 \leq n} |x_{i_1}(i) - x_{i_2}(i)|, \forall i \in \{1, .., p\}. \tag{5}$$

According to the maximum length on each dimension, the box is divided in q subboxes where $q = \Pi_{i=1}^{p} q_i$ and q_i denotes the number of subdivisions on the i-th dimension. Then, the affinity parameter σ is computed as indicated in (3). The number of processors is fixed to $nbproc = q + 1$.



(a) Target example (b) Smiley example

(c) Ratio η function of k (d) Ratio η function of k

Fig. 1. Examples for determining the number of cluster

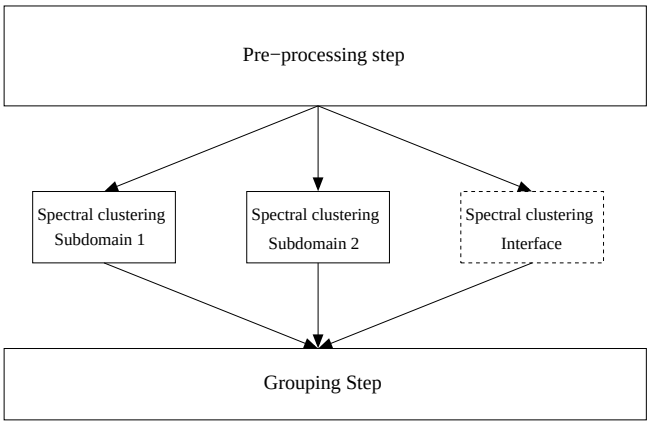


Fig. 2. Principle of parallel Spectral clustering for $q = 2$

3.2 Domain decomposition: Interface and subdomains

Interface It includes all points with a maximum norm distance to the boundaries less than a given γ bandwidth. This interface should help to reconnect together clusters with points in more than one subdomain. Picking up a bandwidth value $\gamma = 3\sigma$ enables to group together points in the same cluster. As the interface layer does not cover the same volume as the other "cubic" subdomains, the isotropic assumption is not anymore satisfied, and a particular affinity parameter σ^* must be considered. We therefore follow the same idea as in section 2.1 but with an adequate volume measure for the interface:

$$\sigma^* = \frac{Vol(interface)}{n_{interface}^{\frac{1}{p}}}$$

where $Vol(interface)$ represents the real volume of the interface and $n_{interface}$ the number of data points in the interface. The volume of the interface is function of bandwidth γ, the number of cut-size q and $l_1, ..l_p$ the edges of the box in each direction as followed:

$$Vol(interface) = \sum_{i=1}^{p}(q_i - 1)\gamma^{p-1}l_i - \gamma^p \Pi_{i=1}^{p}(q_i - 1). \quad (6)$$

Subdomains Each processor from 1 to *nbproc* has a data subset S_i, $i = 1..nbproc$ which coordinates are included in a geometrical subbox. The affinity of all the subdomains have the same global parameter σ defined by (3).

3.3 Spectral clustering on subdomains

Some elements of Algorithm 1 are now precised.

Computation of the spectrum of the affinity matrix (1) Classical routines from LAPACK library [7] are used to compute selected eigenvalues and eigenvectors of the normalized affinity matrix A for each subset of data points.

Number of clusters The number of cluster k is chosen to satisfy (4).

Spectral embedding The centers for k-means in the spectral embedding are initially chosen to be the furthest from each other along a direction.

3.4 Grouping step

The final partition is formed by grouping partitions from the $nbproc - 1$ independent spectral clustering analyses. The grouping is made with the interface

partition and the transitive relation (2). If a point belongs to two different clusters, both clusters are then included in a larger one. As output of the parallel method, a partition of the whole data set S and the final number of clusters k are given.

An example on how our method is applied on a target data set splitted in $q = 4$ subboxes (see Fig. 3). On the left, the clustering result for the interface is plotted. Each color represents a cluster. On the right, the clustering results on the 4 respective subdomains are plotted.

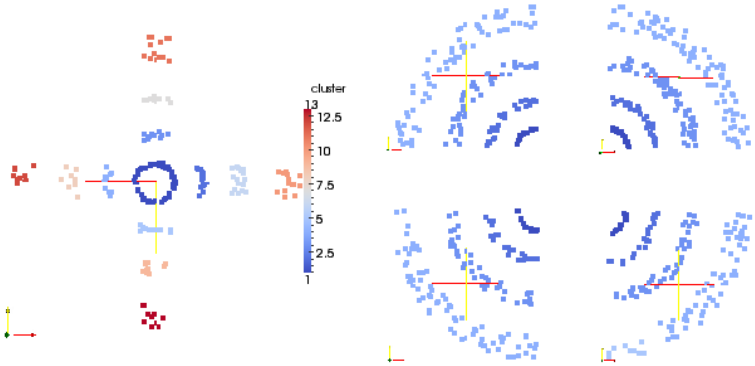


Fig. 3. Target example: interface and subdomains

4 Parallel experiments

As numerical example, this parallel spectral clustering is tested on a 3D geometrical case which represents 2 non-concentric truncated spherical areas included in a larger one as shown in the left of Fig. 4. On the right of the same figure, one zoom around each included truncated sphere is plotted. It shows the proximity between the small spheres and the big one.

The numerical experiments were carried out on the Hyperion supercomputer[1]. Hyperion is the latest supercomputer of the CICT (Centre Interuniversitaire de Calcul de Toulouse). With its 352 bi-Intel "Nehalem" EP quad-core nodes it can develop a peak of 33TFlops. Each node has 4.5 GB memory dedicated for each of the cores with an overall of 32 GB fully available memory on the node. We vary the number of points of this geometrical example from $n = 4361$ to $n = 15247$ points.

[1] http://www.calmip.cict.fr/spip/spip.php?rubrique90

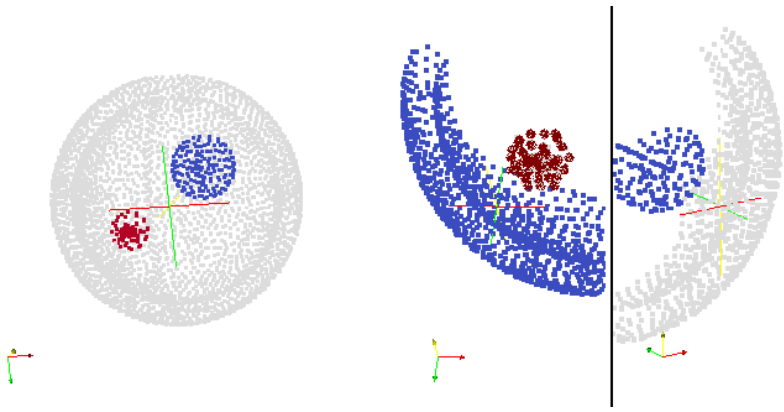


Fig. 4. Geometrical example and zooms: $n = 4361$

For our tests, the domain is successively divided in $q = \{1, 3, 5, 13\}$ subboxes. The timings for each step of parallel Spectral clustering are measured. We give in Table 1, for each problem size and each distribution the number of data in the interface, the total time and the percentage of this time spent in the spectral clustering computation on the subdomains.

n	Number of processors	Number of data in the interface	Total Time (sec)	% of Total Time for spectral clustering
4361	1	-	251.12	99.9
	5	1596	17.69	97.3
	9	2131	32.22	97.7
	13	2559	57.73	98.6
9700	1	-	2930.3	99.9
	5	3601	214.47	99.4
	9	4868	354.77	99.3
	13	5738	628.81	99.6
15247	1	-	> 3h	-
	3	5532	695.41	99.6
	9	7531	1289.43	99.6
	13	8950	2394.01	99.8

Table 1. 3 truncated spheres with interface

We can retain from these results the following information:

- the main part of our algorithm is the spectral clustering on subdomains;
- the time spent in this part is the time of the processor which gets the more data: there is a synchronization point at the end of this part, before the grouping step;
- with this example, the interface gets the maximum number of data;
- the speed-up is larger than the ratio between the total number of points to the maximum data on one subdomain. For example, with $n = 4361$ points and 5 processors, the ratio is 2.73 and the speed-up is 14.12. This can be explained by the non-linearity of our problem with the computation of eigenvectors from Gaussian the affinity matrix.
- the spectral clustering on subdomains is faster than considering the whole data set. Computation of parameters σ, σ^* and the grouping step doesn't penalize our strategy; the time spent in these parts is negligible (less than 2% of total time).

As remarks, the loop implemented to test several values of k in spectral clustering algorithm until satisfying (4) become less and less costly when the number of processors increase. This is due to eigenvectors computation which is less costly with smaller dense affinity matrices. Also, subdividing the whole data set implicitely reduces the Gaussian affinity to diagonal subblocks (after permutations). However when the data set is subdivided in larger numbers of subdomains, the data set of the interface becomes the most time consuming computational task.

We shall investigate its influence and study the trade-off between subdivisions and interface size.

5 Discussion and alternative

As shown in the previous examples, using interface which connects all the partitions could present some limitations. In fact, the more the domain is subdivided, the larger is the set in the interface. So to limit this drawback, a threshold, noted τ, should be defined for the number of subdomains in each axis. This threshold τ represents the ratio between the volume covered by the interface and the total volume.

$$\tau = \frac{Vol(interface)}{Vol} \tag{7}$$

where $Vol(interface)$ is defined by (6) and Vol is the total volume function of l_i defined by (5) for $i = \{1, .., p\}$: $Vol = \Pi_{i=1}^{p} l_i$.

To overcome this drawback of considering the interface as a distinct subdomain, the data set of interface could be included in the others subdomains. In fact, the whole data set is subdivided in q subboxes which have a non-empty intersection. This leads to reduce the number of processors ($nbproc = q$) and avoid computing a special parameter σ^* for the interface. The main advantage

is that the Spectral clustering method is used on all subdomains with the same topology of volume and does not break the isotropic distribution. However the threshold τ is still preserved in order to reduce the time in grouping step. So the volume of the intersection between subdomains is upbounded by a fraction of the volume of the whole data set. Thus, this strategy with intersection is resumed in Fig. 5 for $q = 2$.

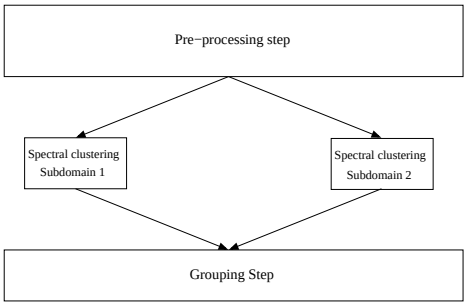


Fig. 5. Principle of alternative parallel Spectral clustering with intersection for $q = 2$

In the same way, Fig. 6 illustrates this alternative on the previous target example divided in $q = 4$ subboxes. On the left, the final clustering results, after the grouping step, is plotted.

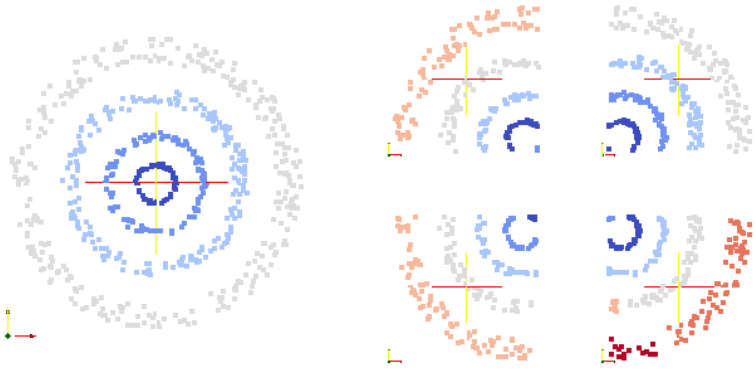


Fig. 6. Target example: subdomains with intersection

5.1 Numerical experiments: Geometrical example

The same examples than in section 4 are tested with this new strategy. In the same way, the results are resumed in the Table 2 with the timings for respective steps of parallel spectral clustering with intersection.

n	Number of processors	Maximum of data by processor	Total Time (sec)	% of Total Time for spectral clustering
4361	1	-	251.12	99.9
	4	1662	27.88	98.4
	8	984	6.25	91.0
	12	1004	6.76	88.5
9700	1	-	2930.3	99.9
	4	3712	304.71	99.6
	8	2265	70.35	98.1
	12	2283	67.27	96.6
15247	1	-	$>$ 3h	-
	4	5760	1034.09	99.8
	8	3531	247.16	98.9
	12	3517	231.71	97.9

Table 2. 3 truncated spheres with intersection between subdomains

We can observe that this alternative has the same main behaviours than the one with interface:

- very good speed-up, much larger than the ratio of the total number of data to the maximum number of data on a subdomain;
- the main part of the time is spent in the spectral clustering step;
- the time of the spectral clustering step is the time of the processor with the maximum number of data.

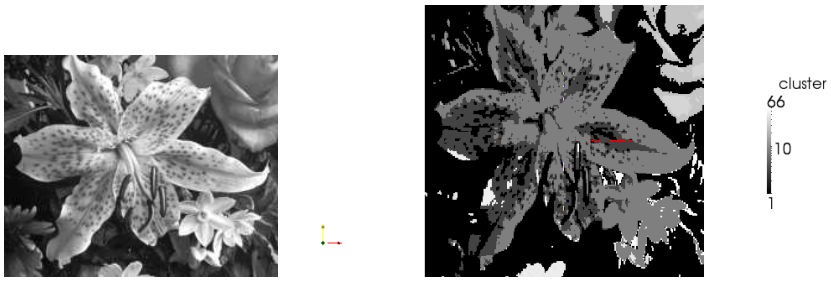
We can express some specific remarks for this strategy:

- the times are better than the interface strategy times with an equivalent number of processor: for example, with $q = 12$ and $n = 15247$, the total time is divided by 10;
- the time is decreasing when the number of subdivisions increases at the condition that the maximum number of data on a processor decreases. We observe, for example, that with $n = 4361$ points and $q = 12$, the processor with the maximum number of points has more points that the equivalent one with $q = 8$. That explains the larger time with $q = 12$ than $q = 8$.

The last remark opens some reflexion about how to divide the domain: a splitting that balances the number of data among the processors will give better results than an automatic splitting of the geometry.

5.2 An image segmentation example

An image segmentation in grayscale is now considered. This kind of example is well designed to the parallel strategy thanks to an uniform distribution with respect to the geometrical coordinates per processor. The affinity matrix is defined as a 3-dimension rectangular box [4] which includes both geometrical coordinates and brightness. The steps between pixels and brightness are about the same magnitude. This means that the image data can be considered as isotropic enough. This approach is tested on an image representing flowers. This image is a 186×230 picture *i.e* $n = 42780$ data points. Due to the large number of data, the parallel spectral clustering is applied on $q = 20$ processors.



(a) Original data set (b) Clustering result

Fig. 7. Example of image segmentation tested on Hyperion

In Fig. 7, the original data set is plotted on the left and the final clustering results on the right. The spectral clustering result has determined 66 clusters. Compared to the original data set, the shapes of the different flowers are well-described. Moreover, the details on the lily can be recognized. The total time spent is equal to 675.67 sec for $n = 42780$ which confirms the computational performance with this parallel spectral clustering with intersection.

6 Conclusion and ongoing works

By exploiting the property of connected components, Spectral clustering could be independently applied on geometrical subdomains without altering the final partition. With an independant way of determining the number of targeted clusters k in each subdomain, the method is completely unsupervised. However, considering an isolated data set for interface presents some limit. It depends on the trade-off between dividing and grouping. The alternative which consists in including this interface in all the subdomains improves the parallel approach.

Futhermore, the strategy could be improved with techniques for distributing uniformily the data per processor and some techniques for sparsifying Gaussian

affinity matrix. On sparse data sets, sparse in the sense of the distribution in the enclosing volume, we may also benefit from techniques of graph partitioning, such as Metis techniques. Applied to the graph of nearest neighbours in the data set, we partition in a more equilibrated way the data points in subsets. Some sparsification techniques, such as thresholding the affinity between data points, could also be introduced to speed up the algorithm when the subdomains are still large enough. It will permit reducing the time dedicated to spectral clustering in subdomains.

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. Belkin, M. and Niyogi, P. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. *Advances in Neural Information Processing Systems*, 2002.
4. Mouysset, S. and Noailles, J. and Ruiz, D. Using a Global Parameter for Gaussian Affinity Matrices in Spectral Clustering, *High Performance Computing for Computational Science: 8th International Conference*, 2008.
5. 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.
6. 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.
7. 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.
8. 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, (to appear).