Parallel Spectral Clustering with FEAST Library

Saad Mdaa, Anass Ouali Alami, Ronan Guivarch(✉), and Sandrine Mouysset

University of Toulouse - IRIT, Toulouse, France
Ronan.Guivarch@toulouse-inp.fr

Abstract. Spectral clustering is one of the most relevant unsupervised learning methods capable of classifying data without any a priori information. At the heart of this method is the computation of the dominant eigenvectors of an affinity matrix in order to work on a low-dimensional data space in which the clustering is made. We propose in this paper a study of the integration of the FEAST library to compute these eigenvectors in our parallel spectral clustering method by domain decomposition. We also show that this library allows to add a second level of parallelism in addition to the domain decomposition level.

Keywords: Kernel methods · FEAST library · Clustering · Parallel computing · Overlapping domain decomposition · Two-level parallelism

1 Introduction

Many fields from Social Science to Medicine and Biology generate a large amount of data to analyze. With the emergence of new technologies, expertise is not always available and data annotation can not be provided. As such unsupervised method are privileged [14]. In particular, spectral clustering is one of the most relevant unsupervised learning methods capable of classifying data without any a priori information on shape or locality [12]. This method consists in selecting the dominant eigenvectors of a matrix called affinity matrix, in order to define a low-dimensional data space in which the data are easily clustered. The most computationally demanding step in this algorithm remains the extraction of the dominant space from the full dense Gaussian affinity matrix. Some parallel approaches can be considered to treat large amount of data. For example, the strategies can be based on either low rank approximations of large matrices, such as Nyström methods [3,13] or subdomain decomposition [5].

In this paper, we investigate the eigensolver library called FEAST that proposes a new transformative numerical approach [9]. To take full advantage of this library, we consider a sparsification of the affinity matrix that allows a second level of parallelism on each subdomain.

The paper is organized as follows. In Sect. 2 we summarize the spectral clustering. Then we present in Sect. 3, the FEAST library, a solver for the generalized eigenvalue problem in order to perform this search of the dominant

T. Guarda et al. (Eds.): ARTIIS 2022, CCIS 1675, pp. 127–138, 2022.
https://doi.org/10.1007/978-3-031-20319-0_10

eigenvectors. In Sect. 4, we include FEAST in spectral clustering by considering both full and sparsified gaussian affinity matrix. In Sect. 5, we present how we incorporate FEAST routines in our parallel spectral clustering method. As a proof of concept, we show in Sect. 6, through some experiments, the good behavior of this approach by comparison with a classical approach using LAPACK/SCALAPACK libraries. Finally, we conclude in Sect. 7.

2 Spectral Clustering

Algorithm 1 presents the different steps of spectral clustering by assuming that the number k of targeted clusters is known. First, the spectral clustering consists in building the affinity matrix based on the Gaussian affinity measure between points of the data set S. After a normalization step, the k eigenvectors associated to the k largest eigenvalues are extracted (*step 3*). 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: Spectral Clustering Algorithm

Input : data set $S = \{x_i\}_{i=1..n} \in \mathbb{R}^p$, number of clusters k, affinity parameter σ

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: $L = D^{-1/2}AD^{-1/2}$ 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

There are several ways to compute, at *step 3*, the eigenvectors associated with the k largest eigenvalues of L. In previous works [5], we used `DSYEV` routine of the LAPACK library [1]. More recently, we replaced the LAPACK routines with an implementation of the Subspace Iteration method [7].

We want to verify the opportunity to use the FEAST library to perform this step. We present in the next section the basic principles of this library.

3 FEAST Library

Introduced by Polizzi [9], FEAST is a solver for the generalized eigenvalue problem $Ax = \lambda Bx$ where A and B are two square matrices of size n [10].

FEAST belongs to the family of iterative solvers based on the integration over a contour of the density matrix, which is a representation used in quantum mechanics. The FEAST algorithm decribed in Algorithm 2 requires as input:

- an interval $I_\lambda = [\lambda_{min}, \lambda_{max}]$ for the search for eigenvalues
- and M, the number of eigenvalues in this interval.

By supplying I_λ and M to FEAST, it first calculates the integral

$$U = \frac{1}{2\pi i} \oint_{\mathcal{C}} (zB - A)^{-1} BY \, \mathrm{d}z$$

where $Y \in \mathbb{C}^{n \times M}$ is made up of M random vectors and $\mathcal{C}$ is a curve in the complex plane enclosing the interval I_λ.

The calculation of this integral is carried out by using a numerical integration scheme (Gaussian quadrature for example), i.e. by using the approximation

$$U \approx \frac{1}{2\pi i} \sum_{i=1}^{m} w_k (z_k B - A)^{-1} BY$$

where $\{z_k\}_{k=1..m}$ are points of the curve $\mathcal{C}$. For each integration node z_k, a linear system $(z_k B - A)U_k = BY$ of size n and with M right-hand sides must be solved.

The result of the integration is used in the Rayleigh-Ritz method, which results in a dense and small eigenvalue problem, whose size is of the order of the number of eigenvalues in the search interval M.

Algorithm 2: FEAST algorithm for solving the generalized eigenvalue problem $Ax = \lambda Bx$

Input : $I_\lambda = [\lambda_{min}, \lambda_{max}]$: the search interval
M: the number of eigenvalues in the search interval

Output: $M_0 \leq M$ eigenpairs

1 Compute $U = \frac{1}{2\pi i} \oint_{\mathcal{C}} (zB - A)^{-1} BY \, \mathrm{d}z$ where $Y \in \mathbb{C}^{n \times M}$ consists of M random vectors and $\mathcal{C}$ is a curve in the complex plane encompassing the interval I_λ ;
2 Compute Rayleigh quotients: $A_U = U^H AU$ and $B_U = U^H BU$;
3 Solve the generalized eigenvalue problem of size M: $A_U W = B_U W \Lambda$;
4 Compute the approximate Ritz pairs: $\Lambda, X = U.W$;
5 If convergence is not achieved, go back to the first step with $Y = X$

Regarding parallelism, the FEAST library can exploit three levels:

- **L1**: Search intervals can be processed separately without overlap;
- **L2**: The linear systems associated with the quadrature nodes in the complex contour can be solved independently;
- **L3**: Each linear system with several second members can be solved in parallel.

4 Spectral Clustering with FEAST

In this section, FEAST is included in spectral clustering by considering both full and sparsified gaussian affinity matrix.

4.1 A First Approach Considering the Dense Affinity Matrix

In the framework of spectral classification, we look for the k largest eigenvalues of the normalized affinity matrix L (*step 3* of Algorithm 1).

L is a real, dense and symmetric matrix. So according to the FEAST manual, the routine to use in this case is **dfeast_syev**. We also need to provide the search interval $I_\lambda = [\lambda_{min}, \lambda_{max}]$ and M the number of eigenvalues in this interval.

As L is a stochastic matrix, we have $\lambda_{max} = 1$. In order to determine λ_{min} and M, one can use a stochastic estimation of the eigenvalues inside the search contour by setting the flag **fpm(14) = 2** according to Algorithm 3:

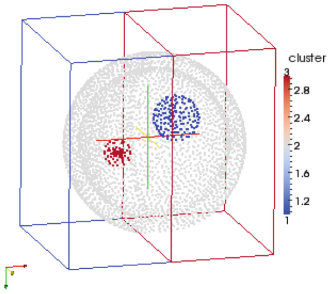
Algorithm 3: Method to compute λ_{min} and M

```
1  λmax = 1;
2  λmin = value close to 1;
3  loop
4  |  Stochastic estimation of M on [λmin, λmax] (by making a first call to
   |   the routine dfeast_syev with the option flag fpm(14)=2);
5  |  if M < k then
6  |  |  λmin = λmin - step;
7  |  else
8  |  |  exit;
9  |  end
10 end
```

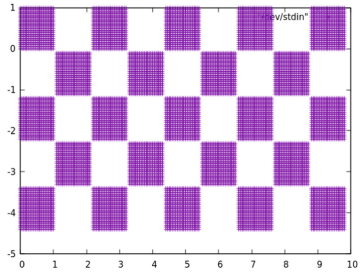
After having determined λ_{min} and M, we make a second call to **dfeast_syev**, to compute the eigenpairs.

To test this approach on some clustering benchmark [11], we select 4 data sets available in our toolbox and we choose to solve the problem with only 1 subdomain in order focus on the eigenvalue problem. Two data sets are respectively plotted in Fig. 1(a) and Fig. 3(a).

The benchmark results obtained using L2-level parallelism with 4 processors are summarized in Table 1.



(a) **sphere2a** data set



(b) **Chequerboard_5x3** data set

Fig. 1. Examples of clustering benchmark

Table 1. FEAST execution time obtained on dense matrices with 4 processors

Data set	Size (n)	Dimension (p)	Number of clusters (k)	Time (t)
Toy	640	2	2	0.39 s
Target	650	2	4	0.75 s
Sphere2	1905	3	2	8.19 s
Sphere2a	3560	3	2	55.31 s

From these first tests, we can remark that estimating M and λ_{min} takes a long time and it is difficult to choose a good step. Moreover, The performance in terms of execution time is not satisfactory for processing large data. Further investigations should be led to identify the most computational steps in this FEAST implementation on spectral clustering. But considering a full dense affinity matrix may strongly impact the execution time. So, in the following, we consider the sparsification of the Gaussian affinity matrix.

4.2 A Second Approach with the Sparsification of the Affinity Matrix

Spectral classification is an expensive algorithm, especially for large data, as it requires computing eigenpairs of a dense matrix of size $n \times n$. To overcome this limitation and memory consumption, sparsification with a threshold can be used.

Indeed, the affinity matrix can be interpreted as a weighted adjacency graph. Thus, the thresholding will control the width of the neighbourhood, and will therefore cancel the edges that connect data points that are very far from each other as shown in Fig. 2. Thus, this reinforces the affinity between points in the same cluster and the separability between clusters [6].

(a) Without thresholding (b) With thresholding

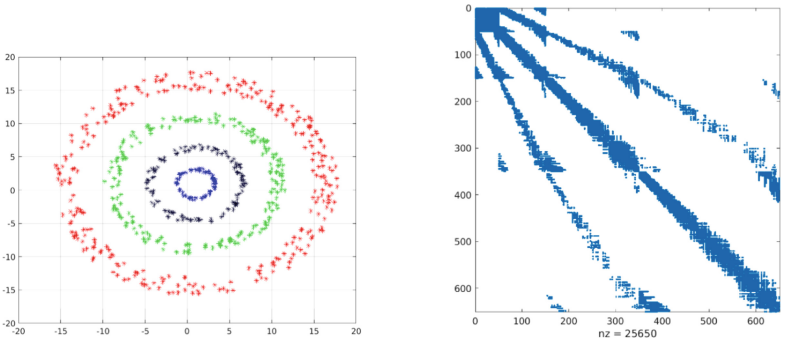
Fig. 2. Thresholding of the weighted adjacency graph

The sparsification of the matrix L is obtained by a threshold proportional to the Gaussian affinity parameter σ (see Algorithm 1):

$$threshold = \alpha \times \sigma.$$

The σ value (and so the threshold) can be heuristically defined to build an automatic sparsified matrix. To do so, we start by considering an uniform distribution of n points in this enclosing p-th dimensional box. This uniform distribution is reached when dividing the box in n smaller boxes all of the same size, each with a volume of order $D^p_{\max}/n$ where D_{max} is the maximum of the distance between two data point x_i and x_j, $\forall i, j \in \{1, .., n\}$. The corresponding edge size is defined by $D_{\max}/n^{\frac{1}{p}}$. The thresholding will be function of this factor which represents a reference distance for any kind of distribution of data S [8].

For sparse matrices, FEAST offers interfaces to obtain the eigenvectors associated with the largest k eigenvalues without specifying $[\lambda_{min}, \lambda_{max}]$.



(a) Clustering result (b) Sparse affinity matrix ($\alpha = 7$)

Fig. 3. Spectral Clustering with sparsification on the Target data set ($n = 650$ and $k = 4$)

Table 2. FEAST execution time obtained on sparse matrices with 4 processors

Data set	Size (n)	Dimension (p)	Number of clusters (k)	α	Time (t)
Target	650	2	4	7	0.5 s
Sphere2	1905	3	2	3	0.2 s
Sphere2a	3560	3	2	3	0.5 s
Cross	5120	2	5	3	10.6 s
Chequerboard2	5000	2	25	3	11.3 s

The results given in Table 2 are obtained using L2-level parallelism with 4 processors. Compared to the first approach (see Table 1), we save a lot of memory and execution time while having a correct classification. So this approach that considers sparse version of the affinity matrix, provides promising results and should be preferred.

5 Strategies of Parallelization

To parallelize the spectral clustering, we first use a domain decomposition strategy and recently implemented a new strategy to exploit more parallelism.

5.1 Domain Decomposition Principle

Our first strategy is based on domain decomposition with overlaps. Lets consider a data set $S = \{x_i\}_{i=1..n} \in \mathbb{R}^p$. This data set is included in a domain. We divide the domain in q sub-domains, thus defining q sub-sets that can have a different amount of data. By assigning a sub-domain to a processor, it applies independently the clustering algorithm on the corresponding sub-set and provides a local partition.

This grouping step is dedicated to link the local partitions from the sub-domains thanks to the overlap and the following transitive relation:

$\forall x_{i_1}, x_{i_2}, x_{i_3} \in S$,

$$\begin{aligned} \text{if} \quad & x_{i_1}, x_{i_2} \in C^1 \text{ and } x_{i_2}, x_{i_3} \in C^2 \\ \text{then} \quad & C^1 \cup C^2 = P \text{ and } x_{i_1}, x_{i_2}, x_{i_3} \in P \end{aligned} \tag{2}$$

where C^1 and C^2 are two distinct clusters and P is a larger cluster which includes both C^1 and C^2. By applying this transitive relation (2) on the overlap, the connection between sub-sets of data is established and provides a global partition.

We can implement this algorithm using a Master-Slave paradigm as summarized in Algorithms 4 and 5.

Algorithm 4: Parallel Algorithm: Master

1: Pre-processing step
   1.1 read the global data and the parameters
   1.2 compute the uniform distance δ
   1.3 compute the overlapping bandwidth α
   1.4 split the data into q sub-sets
2: Send δ and the data sub-sets to the slaves
3: Perform the Clustering Algorithm on its sub-set
4: Receive the local partitions and the number of clusters from each slave
5: Grouping Step
   5.1 Gather the local partitions in a global partition with the transitive relation (2)
   5.2 Output a partition of the whole data set S and the final number of clusters k

Algorithm 5: Parallel Algorithm: Slave

1: Receive δ and its data sub-set from the Master
2: Perform the Clustering Algorithm on its sub-set
3: Send its local partition and its number of clusters to the Master

5.2 A Second Level of Parallelism

The decomposition of the domain in different sub-domains with overlaps consists of a first level of parallelism.

We can also consider that each clustering problem on each sub-domain can be solved using a parallel method. This allows us to use a large number of processors without splitting the domain into many sub-domains, which can be penalizing in some situations.

We have the possibility to use this second level of parallelism with the spectral clustering method. We can use SCALAPACK [2] and its routine `PDSYEV` in order to compute all the eigenvectors of the matrix L.

We have a group of leading processors that will handle the first level of parallelism. Then, for each sub-domain, there is a pool of processors, including the leading processor of the sub-domain, that perform the computation via SCALAPACK.

This organization is illustrated in the Fig. 4 for 4 sub-domains. M_1, M_2, M_3 and M_4 are the leading processors.

Or we also can exploit the different levels of parallelism in FEAST. On each sub-domain, we call FEAST using L2-level parallelism with the dedicated pool of processors. We present in the next section the comparison between LAPACK/SCALAPACK and FEAST.

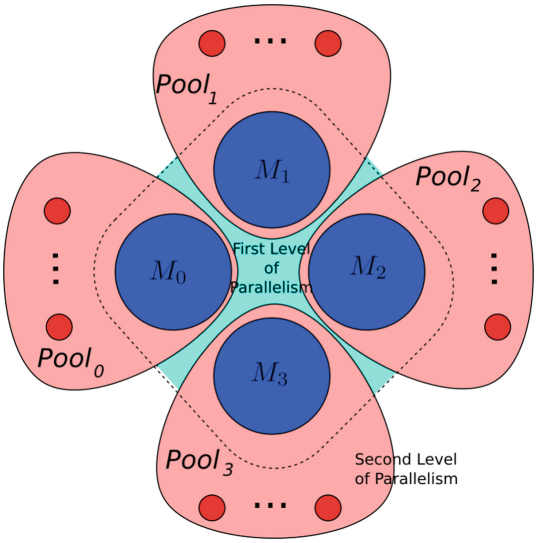


Fig. 4. Two-level Parallelism

6 Numerical Results: Proof of Concept

As proof of concept, we first performed some calculations with small data sets to compare FEAST to LAPACK/SCALAPACK. Then we give first results on larger data sets.

6.1 Clustering Benchmark

In the following, n is the size of the data, p is the dimension of the problem ($2D$ or $3D$), and **nb_sd** is the number of sub-domains. The Table 3 summarizes the time to solve the eigenvalue problem and the total time.

The number of processors in a pool is 1 or 2. With 1 processor, we use LAPACK or FEAST with no parallelization to solve a problem on a sub-domain. With 2 processors, we use SCALAPACK and FEAST with L2-level parallelism.

The following measure was used to quantify the performance of FEAST:

$$\textbf{efficiency } \% = \frac{\text{time with 1 processor}}{2 \times \text{time with 2 processors}} \times 100$$

We see that for all these problems, the results with FEAST using one processor per subdomain or two (i.e. not using, or using the second level of parallelism) are better than those with LAPACK and SCALAPACK. The times for the two problems of size n close to 10000 (time for the resolution of the problem and the total time) are better by a factor 10. We also note, that even for small problems, we benefit from the second level of parallelism, which is not the case with the

Table 3. Results on clustering benchmark

3SPHERES ($n = 4361$, $p = 3$ and **nb_sd** $= 8$)					
Solver	LAPACK	SCALAPACK	FEAST		
Size of the pool	1	2	1	2	Efficiency %
Eigenvalue problem time	3.16 s	3.90 s	1.00 s	**0.81 s**	61.72%
Total time	3.35 s	4.10 s	1.11 s	**0.91 s**	60.98%
3SPHERESA ($n = 9700$, $p = 3$ et **nb_sd** $= 8$)					
Solver	LAPACK	SCALAPACK	FEAST		
Size of the pool	1	2	1	2	Efficiency %
Eigenvalue problem time	34.68 s	46.45 s	3.42 s	**3.10 s**	55.16%
Total time	35.68 s	47.44 s	4.71 s	**4.39 s**	53.64%
SPHERE2B ($n = 10717$, $p = 3$ et **nb_sd** $= 8$)					
Solver	LAPACK	SCALAPACK	FEAST		
Size of the pool	1	2	1	2	Efficiency %
Eigenvalue problem time	37.5 s	50.30 s	3.23 s	**2.72 s**	59.37%
Total time	38.52 s	51.27 s	3.55 s	**3.05 s**	58.19%

LAPACK/SCALAPACK implementation. We show in previous works that for the latter approach, it is only interesting to use SCALAPACK for much larger problems. The efficiency are between 50% and 60% which is promising for small problems.

6.2 Tests on Larger Data Sets

In order to test the different approaches with larger problems and in particular to observe how the speed-up behaves, we consider the **Chequerboard_5 × 3** data sets plotted in Fig. 1(b). We can change the density of the points of each dark square, in order to increase the number of points of the problem and form, for this experiment, five data sets with $\{10,143; 42,527; 94,208; 197,568; 258,458\}$ points. We divide our domain into 16 sub-domains.

Table 4. Total execution time with LAPACK/SCALAPACK and FEAST on **Chequerboard_5 × 3_XX data sets**

Solver	Nb of points	Nb of points / subdomain	FEAST		
Data set	**Pool's size**				
	-	-	1	2	Efficiency %
Chequerboard_5x3_21	10,143	833	0.59 s	**0.55 s**	53.63%
Chequerboard_5x3_42	42,527	3,354	6.19 s	**4.97 s**	62.27%
Chequerboard_5x3_63	94,208	7,360	37.39 s	**25.17 s**	74.27%
Chequerboard_5x3_84	197,568	14,705	178.97 s	**124.42 s**	71.92%
Chequerboard_5x3_105	258,428	24,556	451.22 s	**330.20 s**	68.32%

With the three biggest problems, the speed-up is close to 70% which confirms the interest of using the second level of parallelism.

We have not indicated in this table the time of the LAPACK/SCALAPACK versions but they are always much slower than FEAST. For example, for the problem of size 258,428, it takes more than two hours to get the result.

7 Conclusion

In this paper, we have shown that the use of FEAST to compute the eigenvectors in the spectral clustering method is advantageous compared to its implementation with the eigenpair computation routines of LAPACK.

This advantage remains when we consider the parallel spectral clustering with domain decomposition and allows a second level of parallelism by activating the L2-level parallelism of FEAST.

This L2-level parallelism of FEAST, when we are able to sparsify the affinity matrix, allows, with consequent problem sizes, to obtain speed-ups close to 70% compared to FEAST with no parallelism.

In the future, we plan to validate our approach on real data (images, social science, medicine and biology data) on supercomputers. Also, because we sparsify the affinity matrix, it would be interesting to compare FEAST with the routines of the ARPACK library [4] which are designed to handle sparse matrices.

References

1. Anderson, E., et al.: LAPACK Users' Guide. USA, third edn, SIAM, Philadelphia, Pennsylvania (1999)
2. Blackford, L.S., et al.: ScaLAPACK users' guide. SIAM (1997)
3. Fowlkes, C., Belongie, S., Chung, F., Malik, J.: Spectral grouping using the Nystrom method. IEEE Trans. Pattern Anal. Mach. Intell. **26**(2), 214–225 (2004)
4. Lehoucq, R.B., Sorensen, D.C., Yang, C.: ARPACK: solution of large scale eigenvalue problems by implicitly restarted Arnoldi methods. www.netlib@ornl.gov (1997)
5. Mouysset, S., Noailles, J., Ruiz, D., Guivarch, R.: On a strategy for spectral clustering with parallel computation. In: High Performance Computing for Computational Science: 9th International Conference (2010)
6. Mouysset, S., Guivarch, R.: Sparsification on Parallel Spectral Clustering. In: Daydé, M., Marques, O., Nakajima, K. (eds.) VECPAR 2012. LNCS, vol. 7851, pp. 249–260. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38718-0_25
7. Mouysset, S., Guivarch, R.: ParKerC: Toolbox for parallel kernel clustering methods. In: 10th International Conference on Pattern Recognition Systems - ICPRS 2019. IET: Institution of Engineering and Technology, Tours, France (2019). https://doi.org/10.1049/cp.2019.0253
8. Mouysset, S., Noailles, J., Ruiz, D.: On an interpretation of spectral clustering via heat equation and finite elements theory. Lecture Notes in Engineering and Computer Science (2010)

9. Polizzi, E.: Density-matrix-based algorithm for solving eigenvalue problems. Phys. Rev. B **79**(11), 115112 (2009)
10. Polizzi, E.: Density-matrix-based algorithm for solving eigenvalue problems. Phys. Rev. B **79**, 115112 (2009). https://doi.org/10.1103/PhysRevB.79.115112
11. Thrun, M.C., Ultsch, A.: Clustering benchmark datasets exploiting the fundamental clustering problems. Data Brief **30**, 105501 (2020)
12. Von Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007)
13. Wang, S., Gittens, A., Mahoney, M.W.: Scalable kernel k-means clustering with Nyström approximation: relative-error bounds (2017)
14. Xu, D., Tian, Y.: A comprehensive survey of clustering algorithms. Annals Data Sci. **2**(2), 165–193 (2015)