Chapter 11
Implementations and Data Sets

11.1 Introduction

We discuss the implementations of various incremental clustering algorithms and provide some recommendations on the choice of their parameters. These algorithms are designed by combining the multi-start incremental algorithm with either NSO methods or variants of the k-means algorithm. All the NSO based incremental clustering algorithms involve the algorithm for finding initial cluster centers. Using numerical results, we discuss the choice of parameters for the latter algorithm. In addition, we describe some real-world data sets that are used to evaluate the clustering algorithms.

We start this chapter by introducing the algorithms applied in our numerical experiments and by providing details of their implementations. Then, in Sect. 11.3 we present data sets used in our experiments. Finally, in Sect. 11.4 we discuss the parameters selection for the algorithm that is used to find initial cluster centers (i.e., Algorithm 7.2 given in Sect. 7.4).

11.2 Implementations of Clustering Algorithms

We use the following algorithms in our numerical experiments:

- MS-KM—multi-start k-means algorithm;
- GKM—global k-means algorithm (Sect. 5.2.3);
- MGKM—modified global k-means algorithm (Sect. 8.2);
- LMB-CLUST—limited memory bundle method for clustering (Sect. 8.4);
- DG-CLUST—discrete gradient clustering algorithm (Sect. 8.5);
- IS-CLUST—smooth incremental clustering algorithm (Sect. 8.6);
- NDC-CLUST—incremental nonsmooth DC clustering algorithm (Sect. 9.2);

A. M. Bagirov et al., *Partitional Clustering via Nonsmooth Optimization*,
Unsupervised and Semi-Supervised Learning,
https://doi.org/10.1007/978-3-030-37826-4_11

- DCDB-CLUST—DC diagonal bundle clustering algorithm (Sect. 9.3);
- IDCA-CLUST—incremental DCA for clustering (Sect. 9.4).

The source codes (Fortran77 or Fortran95) of these algorithms are available at https://github.com/SnTa2019/Clustering-via-Nonsmooth-Optimization and also at http://napsu.karmitsa.fi/clustering/.

In the first three clustering algorithms, MS-KM, GKM, and MGKM, the underlying solver is the k-means algorithm (see Sect. 5.2). We use the standard implementation of the k-means algorithm in our experiments. The stopping criterion in this algorithm is formulated using the maximum number of data points which are allowed to change their clusters. More specifically, for some $\delta \geq 0$ the number $m_c = \delta m$ is defined, where m is the number of points in a data set. If the number of points changing their clusters is less than or equal to m_c, then the k-means algorithm terminates. In extra small, small, and medium sized data sets we set $\delta = 0$ and for large and very large data sets we use $\delta = 10^{-3}$.

Unlike other clustering algorithms listed above, the MS-KM is not an incremental algorithm and starting points for the clustering problem are generated using a simple multi-start randomized scheme. Note that, this algorithm does not provide any intermediate solutions for $1 < l < k$ and therefore, several runs need to be performed with different number of clusters. We denote by `MS-KM` the implementation of the MS-KM and set the maximum number of starting points in `MS-KM` to 1000.

`GKM` is an implementation of the incremental algorithm with the k-means algorithm, where each data point is used as a starting point for the kth cluster center and no auxiliary clustering problem is utilized. The original `GKM` utilizes the mixed integer programming formulation (4.2) of the clustering problem, however, the NSO formulation (4.3) of this problem can also be used. In its original design, `GKM` is only suitable for clustering relatively small data sets. In the implementation, we use only one starting point for the kth cluster center. More specifically, a data point which provides the largest decrease of the cluster function is utilized. This allows us to apply `GKM` to large data sets. Furthermore, we consider two different implementations of the GKM as follows:

- the affinity (or distance) matrix of a data set is computed before the application of the GKM. This implementation is applicable to small and medium sized data sets as the affinity matrix for large data sets cannot be stored in the memory of a computer and
- the affinity matrix is computed at each iteration. We can apply this implementation to large data sets.

Similar to other incremental clustering algorithms, `GKM` solves all the intermediate l-partition problems where $l = 1, \ldots, k$.

`MGKM` is an implementation of the MGKM. This is an incremental algorithm which utilizes the auxiliary clustering problem and the algorithm for finding starting cluster centers. For the parameters of the latter one, we refer to Sect. 11.4. At each iteration of the MGKM, the clustering problem is solved by applying the k-means

algorithm and the auxiliary clustering problem is solved by utilizing the partial k-means algorithm where all cluster centers are fixed but one. We use the stopping tolerance $\varepsilon = 0$ (see Algorithm 8.2 in Sect. 8.2) and therefore, `MGKM` always computes up to the maximum number of clusters provided by the user.

`LMB-Clust` is an implementation of the LMB-CLUST specifically developed for solving clustering problems in large and very large data sets. In our experiments, we use the stopping tolerance $\varepsilon_c = 10^{-3}$. For small and medium sized data sets, the stopping tolerance should be set smaller than 10^{-3}. In addition, we apply the modified version of the underlying optimization solver LMBM with the initial number of stored correction pairs (used to form the variable metric update) equal to 7 and the maximum number of stored correction pairs equal to 15. Otherwise, the default values of parameters of the LMBM are used.

`DG-Clust` is an implementation of the DG-CLUST. As mentioned before, the DG-CLUST exploits piecewise separability of the cluster and auxiliary cluster functions. It does not utilize the subgradients of the objective cluster functions, instead it approximates them using values of the objective functions. Thus `DG-Clust` is suitable also for solving clustering problems with the similarity measures d_1 and d_∞. The simplified scheme, described in Sect. 8.5, is applied to compute discrete gradients of the cluster and the auxiliary cluster functions which allows us to significantly reduce the number of distance function evaluations. The most important parameter in `DG-Clust` is $\lambda > 0$ for approximation of the subgradients. Its initial value is chosen $\lambda_1 = 1$ and at the hth ($h > 1$) iteration it is updated as $\lambda_h = \mu\lambda_{h-1}$ where $\mu = 0.2$. Other parameters are chosen as $\delta_h = 10^{-7}$ for all h, $\varepsilon = 10^{-6}$ and $\alpha = 1$.

`IS-Clust` is an implementation of the IS-CLUST. The smoothing (or precision) parameter is chosen as $\tau_h = \mu\tau_{h-1},\ h > 1$ where $\tau_1 = 1$ and $\mu = 0.1$. To solve the smooth optimization problems, `IS-Clust` applies the Quasi-Newton method with the BFGS update.

`NDC-Clust` is an implementation of the NDC-CLUST. This algorithm takes into account the DC representation of the objective cluster function. The main parameter in `NDC-Clust` is the step $\lambda > 0$ used to approximate the first DC component. This parameter is chosen as $\lambda_h = \mu\lambda_{h-1},\ h > 1$, where $\mu = 0.1$ and $\lambda_1 = 1$. Otherwise, the default parameters of the underlying optimization solver are used. In particular, $\delta_h = 10^{-7}$ for all h, and $\varepsilon = 10^{-6}$.

`DCDB-Clust` is an implementation of the DCDB-CLUST, where the DC structure of the clustering and the auxiliary clustering problems are utilized. This method is designed for solving clustering problems in large data sets. In our experiments, we use the stopping tolerance $\varepsilon_c = 10^{-3}$. Similar to `LMB-Clust`, the stopping tolerance should be set smaller for small and medium sized data sets. The underlying optimization solver DCD-BUNDLE uses seven correction pairs to form the diagonal updates. In addition, the default values of other parameters of the optimization solver are used.

`IDCA-Clust` is an implementation of the IDCA-CLUST. Since the objective functions in the subproblem (9.2) of the IDCA-CLUST are convex quadratic

functions with the simple structure their unique minima can be calculated explicitly without involving any optimization procedure or tuneable parameters.

All computational experiments were carried out in a PC with the CPU Intel(R) Core(TM) i5-8250U 1.60 GHz and RAM 8 GB working under Windows 10.

11.3 Data Sets

Data sets can be classified based on different parameters. Such parameters include the number of data points, the number of attributes, the number of classes if they are available (binary or multi-class), types of attributes (numeric, categorical, or mixture of both), completeness or incompleteness of data, in particular, absence of values of some attributes in some data points (missing values). We will use data sets with numeric attributes and no missing values.

The definition of the complexity of a data set is a problem and a model dependent. For instance, the complexity of a data set may be different from the perspective of the supervised data classification and clustering. Different models of a clustering problem contain different types and numbers of variables. Some of these models have constraints. Therefore, the complexity of a data set with respect to each of these models might be different. It is also not an easy task to determine how the complexity of a data set affects the time complexity of an algorithm. If a data set has well-separated clusters, then most clustering algorithms may require significantly less iterations than in data sets with not well-separated clusters.

In most optimization based clustering algorithms, considered in this book, optimization algorithms require the calculation of the value of the cluster functions and their one subgradient at each point. In the case of the MSSC problems, the objective is piecewise quadratic and quadratic functions can be represented as a sum of the squared Euclidean distances. In the case of the similarity measures d_1 and d_∞, the objective cluster function is piecewise linear. For such functions, the complexity of calculation of their values and subgradients depends on the number of points and attributes in a data set. This means that the complexity depends on the number of entries in a data set. Therefore, we use the number of entries to classify data sets as extra small, small, medium sized, large, and very large data sets.

11.3.1 Extra Small Data Sets

We group those data sets that contain no more than 3000 entries as extra small. The brief description of these data sets, used in our numerical experiments, and their references are given in Table 11.1.

Table 11.1 Brief description of extra small data sets

Data sets	Number of instances	Number of attributes	Number of classes	Number of entries	Refs.
German towns	59	2	–	116	[272]
Bavaria postal 1	84	3	–	252	[272]
Bavaria postal 2	84	4	–	336	[272]
Iris plant	150	3	3	450	[91]
TSPLIB1060	1060	2	–	2120	[246]
Liver disorder	356	6	2	2136	[91]

Table 11.2 Brief description of small data sets

Data sets	Number of instances	Number of attributes	Number of classes	Number of entries	Refs.
Heart disease	297	13	2	3861	[91]
TSPLIB3038	3038	2	–	6076	[246]
Pima Indians diabetes	768	8	2	6144	[91]
Breast cancer Wisconsin	683	9	2	6147	[91]
Ionosphere	351	34	2	11,934	[91]
Vehicle silhouettes	846	18	4	15,228	[91]

Table 11.3 Brief description of medium sized data sets

Data sets	Number of instances	Number of attributes	Number of classes	Number of entries	Refs.
D15112	15,112	2	–	30,224	[246]
Image segmentation	2310	19	7	43,890	[91]
Page blocks	5473	10	5	54,730	[91]
Pla85900	85,900	2	–	171,800	[246]
Pen-based recognition of handwritten digits	10,992	16	10	175,872	[91]

11.3.2 Small Data Sets

The data sets containing more than 3000 but less than 20,000 entries are classified as small. The brief description of such data sets, including their references, is given in Table 11.2.

11.3.3 Medium Sized Data Sets

Medium sized data sets are those containing more than 20,000 but less than 2.0×10^5 entries. These data sets are presented in Table 11.3 with their corresponding references.

11.3.4 Large Data Sets

We group the data sets containing more than 2×10^5 but less than 2×10^6 entries as large. The brief description of these data sets, including their corresponding references, is given in Table 11.4.

11.3.5 Very Large Data Sets

Very large data sets are those containing more than 2×10^6 entries. We give the brief description of these data sets and their corresponding references in Table 11.5.

Table 11.4 Brief description of large data sets

Data sets	Number of instances	Number of attributes	Number of classes	Number of entries	Refs.
EEG eye state	14,980	14	2	209,720	[91]
Landsat satellite	6435	36	6	231,660	[91]
Letter recognition	20,000	16	26	320,000	[91]
Optical recognition of handwritten digits	5620	64	10	359,680	[91] [91]
Person activity	164,860	3	11	494,580	[91]
Shuttle control	58,000	9	7	522,000	[91]
Skin segmentation	245,057	3	2	735,171	[91]
KEGG metabolic relation network	53,413	20	–	1,068,260	[91] [91]
3D road network	434,874	3	–	1,304,622	[91]
Gas sensor array drift	13,910	128	6	1,780,480	[91]

Table 11.5 Brief description of very large data sets

Data sets	Number of instances	Number of attributes	Number of classes	Number of entries	Refs.
Online news popularity	39,644	58	2	2,299,352	[91]
Sensorless drive diagnosis	58,509	48	11	2,866,941	[91]
ISOLET	7797	616	26	4,802,952	[91]
Covertype	581,012	10	7	5,810,012	[91]
MiniBooNE particle identification	130,064	50	2	6,503,200	[91] [91]
Gisette	13,500	5000	–	67,500,000	[91]

11.4 Parameters Selection in Finding Starting Cluster Centers

One of the most important components of the incremental clustering algorithms is the procedure for finding starting cluster centers. This procedure is given in Algorithm 7.2. It contains three parameters $\gamma_1, \gamma_2 \in [0, 1]$, and $\gamma_3 \in [1, \infty)$ whose optimal values depend on the size of a data set. More precisely, they depend more on the number of data points than on the number of attributes.

Assume that $l \geq 2$ and the solution to the $(l-1)$-clustering problem is known. Algorithm 7.2 consists of the following three main steps:

- first, each data point is considered as a candidate starting cluster center together with the solution of the previous $(l-1)$-clustering problem obtained by some incremental clustering algorithm. A data point which provides the largest decrease of the clustering function is determined. Then a threshold is computed by multiplying this maximum decrease by the parameter $\gamma_1 \in [0, 1]$. The data points which provide the decrease of the clustering function greater than this threshold are selected as potential starting cluster centers while the rest of the data points are removed;
- second, the selected data points are replaced by the centers of clusters around them, and the decrease of the clustering function is calculated using these centers. Similar to the previous step, a threshold is computed using the maximum decrease with the parameter $\gamma_2 \in [0, 1]$, and the centers providing the decrease greater than this threshold are kept as potential starting cluster centers; and
- in the last step, the auxiliary clustering problem is solved starting from each point left in the list of potential starting cluster centers, and the values of the auxiliary clustering function are computed at each (local) solution obtained. Then using the smallest value of this function, one more threshold parameter—$\gamma_3 \in [1, \infty)$—is defined. Solutions of the auxiliary clustering problem with the function values less than this threshold are included to the final list of starting cluster centers. Together with the solution of the previous $(l-1)$-clustering problem this list is used as a set of starting points for solving the l-clustering problem.

Note that there is no theoretical result which would help us to find exact values of the parameters γ_1, γ_2, and γ_3. We use numerical experiments on some data sets with different sizes to provide some recommendations on the values of these parameters. Let us denote by $\bar{A}_1(\gamma_1)$, the set $\bar{A}_1$ obtained from (7.12) using $\gamma_1 \in [0, 1]$; by $\bar{A}_3(\gamma_1, \gamma_2)$, the set $\bar{A}_3$ obtained from (7.14) using $\gamma_1, \gamma_2 \in [0, 1]$; and by $\bar{A}_5(\gamma_1, \gamma_2, \gamma_3)$, the set $\bar{A}_5$ obtained from (7.16) using $\gamma_1, \gamma_2 \in [0, 1]$, $\gamma_3 \in [1, \infty)$. It is clear that for all $\gamma_1 \leq \mu_1$, $\gamma_2 \leq \mu_2$, $\gamma_3 \geq \mu_3$ we have

$$\begin{aligned} &\bar{A}_1(\mu_1) \subseteq \bar{A}_1(\gamma_1), \\ &\bar{A}_3(\mu_1, \mu_2) \subseteq \bar{A}_3(\gamma_1, \gamma_2), \quad \text{and} \\ &\bar{A}_5(\mu_1, \mu_2, \mu_3) \subseteq \bar{A}_5(\gamma_1, \gamma_2, \gamma_3). \end{aligned} \tag{11.1}$$

Notice that the set $\bar{A}_1(0)$ contains all data points which are not cluster centers. This means that if $\gamma_1 = \gamma_2 = 0$ and γ_3 is sufficiently large, then the number of starting points in Step 5 of Algorithm 7.2 is the number of data points which are not cluster centers. This is the largest number of starting points for the clustering problem which can be obtained. The least number of starting points is obtained when $\gamma_1 = \gamma_2 = \gamma_3 = 1$.

The inclusions in (11.1) imply that an incremental clustering algorithm obtains its best solution when $\gamma_1 = \gamma_2 = 0$ and γ_3 is sufficiently large, and this solution cannot be improved using any other values of $\gamma_1, \gamma_2 \in [0, 1]$ and $\gamma_3 \in [1, \infty)$. Nevertheless, computational effort required by any incremental clustering algorithm reduces as parameters γ_1 and γ_2 increase and the parameter γ_3 decreases. Therefore, for a given data set we are interested in finding the largest values of γ_1 and γ_2, and the smallest value of γ_3 such that an incremental clustering algorithm can still obtain its best solution to the clustering problem and further increase of γ_1, γ_2 or decrease of γ_3 deteriorates the solution.

Let us denote by $\tilde{f}_k(\gamma_1, \gamma_2, \gamma_3)$ the value of the cluster function f_k, given in (4.4), obtained by an incremental clustering algorithm for the given values of $\gamma_1, \gamma_2, \gamma_3$, and k clusters. First, we set $\gamma_1 = \gamma_2 = 0$ and $\gamma_3 = 10$ (assuming that this value is sufficiently large) and find $\tilde{f}_k(0, 0, 10)$. This means that we compute the largest possible sets $\bar{A}_1$ and $\bar{A}_3$ and find the best possible solution by the incremental algorithm. Then we calculate $\tilde{f}_k(\gamma_1, \gamma_2, \gamma_3)$ for

$$\begin{aligned} &\gamma_1, \gamma_2 = 0.05i, && i = 1, \ldots, 20, \quad \text{and} \\ &\gamma_3 = 1 + 0.01(i-1), && i = 1, \ldots, 101. \end{aligned}$$

The largest values of γ_1, γ_2 and the smallest value of γ_3 satisfying the condition

$$\frac{\tilde{f}_k(\gamma_1, \gamma_2, \gamma_3) - \tilde{f}_k(0, 0, 10)}{\tilde{f}_k(0, 0, 10)} \leq \varepsilon \tag{11.2}$$

are accepted as an estimation of the optimal values of parameters γ_j, $j = 1, 2, 3$. Here, $\varepsilon \geq 0$ is a given tolerance.

Next, we demonstrate how to find estimations for the optimal values of γ_j, $j = 1, 2, 3$. For this aim, we apply `IS-Clust` on data sets Iris plant and the Breast cancer Wisconsin (see Tables 11.1 and 11.2 for details of these data sets). The results with different values of parameters are presented in Tables 11.6 and 11.7. We give first the results with $\gamma_1 = \gamma_2 = 0$ and $\gamma_3 = 10$, and then the results with the largest values of γ_1, γ_2 and the smallest value of γ_3 that satisfy condition (11.2). Finally, we present results with nondecreasing values of γ_1 and γ_2. In these tables, we use the following notations:

- k—the number of clusters;
- E—the error in % computed using (10.9); and
- α—the parameter defined by

$$\alpha = \frac{N_d(\gamma_1, \gamma_2, \gamma_3)}{N_d(0, 0, 10)},$$

where $N_d(\gamma_1, \gamma_2, \gamma_3)$ is the number of distance function evaluations by the incremental algorithm for given values of $\gamma_1, \gamma_2, \gamma_3$. It is clear that $N_d(0, 0, 10)$ is the number of distance function evaluations by the same algorithm when $\gamma_1 = \gamma_2 = 0$ and $\gamma_3 = 10$. The parameter α reflects the ratio of computational effort for given $\gamma_1, \gamma_2 \in [0, 1]$ and $\gamma_3 \geq 1$ with respect to that of for $\gamma_1 = \gamma_2 = 0$ and $\gamma_3 = 10$.

Since the data set Iris plant is extra small we take $\varepsilon = 0$ for it. From Table 11.6, we see that the largest values of γ_1, γ_2 and the smallest value of γ_3 satisfying the condition (11.2) are: $\gamma_1 = 0.50, \gamma_2 = 0.55, \gamma_3 = 1.10$. Results show that any increase of γ_1, γ_2 and any decrease of γ_3 deteriorate the best solution. It can also be observed that values of $\gamma_2 < \gamma_1$ does not improve the accuracy of the algorithm. Results given in columns at the bottom right corner of Table 11.6 confirm this claim. Results for the parameter α demonstrate that the selection of optimal values of γ_j, $j = 1, 2, 3$ allows us to significantly reduce the computational effort without deteriorating the clustering solution.

For the data set Breast cancer Wisconsin, we take $\varepsilon = 0.01$. The largest values of γ_1, γ_2 and the smallest value of γ_3 satisfying the condition (11.2) are: $\gamma_1 = 0.60, \gamma_2 = 0.80, \gamma_3 = 1.01$. Results in Table 11.7 show that any increase of γ_1, γ_2 or

Table 11.6 Results for different values of $(\gamma_1, \gamma_2, \gamma_3)$: Iris plant

k	E	α	E	α	E	α	E	α
	(0.00, 0.00, 10.00)		(0.50, 0.55, 1.10)		(0.50, 0.55, 1.05)		(0.55, 0.55, 1.10)	
2	0.00	1.00	0.00	0.11	0.00	0.11	0.00	0.09
3	0.00	1.00	0.00	0.34	0.00	0.34	0.00	0.30
4	0.00	1.00	0.00	0.32	0.00	0.31	0.00	0.28
5	0.00	1.00	0.00	0.28	0.00	0.25	0.00	0.23
6	0.00	1.00	0.00	0.36	0.00	0.33	0.00	0.31
7	0.00	1.00	0.00	0.28	0.01	0.26	0.01	0.24
8	0.00	1.00	0.00	0.20	0.25	0.19	0.25	0.17
9	0.00	1.00	0.00	0.17	0.27	0.17	0.27	0.15
10	0.00	1.00	0.00	0.15	0.00	0.16	0.00	0.14
	(0.55, 0.55, 2.00)		(0.50, 0.60, 1.10)		(0.50, 0.60, 2.00)		(0.55, 0.00, 2.00)	
2	0.00	0.38	0.00	0.11	0.00	0.48	0.00	0.38
3	0.00	0.39	0.00	0.34	0.00	0.45	0.00	0.39
4	0.00	0.33	0.00	0.32	0.00	0.39	0.00	0.33
5	0.00	0.26	0.00	0.27	0.00	0.32	0.00	0.26
6	0.00	0.26	0.00	0.26	0.00	0.29	0.00	0.26
7	0.01	0.25	0.01	0.25	0.01	0.28	0.01	0.25
8	0.25	0.21	0.25	0.22	0.25	0.24	0.25	0.21
9	0.27	0.21	0.27	0.23	0.27	0.24	0.27	0.21
10	0.00	0.23	0.00	0.25	0.00	0.26	0.00	0.23

Table 11.7 Results for different values of $(\gamma_1, \gamma_2, \gamma_3)$: Breast cancer Wisconsin

k	E	α	E	α	E	α	E	α
	(0.00, 0.00, 10.00)		(0.60, 0.80, 1.01)		(0.60, 0.80, 2.00)		(0.60, 0.80, 1.00)	
2	0.00	1.00	0.00	0.34	0.00	0.34	0.00	0.06
5	0.00	1.00	0.00	0.16	0.00	0.16	0.61	0.02
10	0.00	1.00	0.00	0.08	0.00	0.08	0.03	0.01
15	0.00	1.00	0.85	0.06	0.85	0.06	1.22	0.01
20	0.00	1.00	0.76	0.06	0.76	0.06	0.70	0.01
25	0.00	1.00	0.00	0.06	0.00	0.06	1.22	0.01
	(0.65, 0.80, 1.01)		(0.65, 0.80, 2.00)		(0.60, 0.85, 1.01)		(0.60, 0.85, 2.00)	
2	0.00	0.27	0.00	0.27	0.00	0.34	0.00	0.34
5	0.00	0.13	0.00	0.13	0.00	0.14	0.00	0.14
10	0.00	0.07	0.00	0.07	0.00	0.06	0.00	0.06
15	1.03	0.06	1.03	0.05	1.02	0.05	1.02	0.05
20	0.99	0.05	0.99	0.05	0.80	0.04	0.80	0.04
25	0.00	0.05	0.00	0.05	0.22	0.04	0.00	0.04

any decrease of γ_3 slightly deteriorates the best solution obtained by the incremental algorithm. Results for the parameter α demonstrate that the selection of estimations of the optimal values of $\gamma_j,\ j = 1, 2, 3$ allows us to significantly reduce the number of distance function evaluations in comparison with those required by the algorithm when $\gamma_1 = \gamma_2 = 0$ and $\gamma_3 = 10$. We can see that values of γ_1 and γ_2 are larger and γ_3 is smaller for this data set comparing to those for Iris plant.

Results presented in Tables 11.6 and 11.7, in general, lead to the following observations:

- the optimal values of $\gamma_1,\ \gamma_2 \in [0, 1]$ increase and the optimal value of $\gamma_3 \geq 1$ decreases as the size of a data set increases;
- we can select $\gamma_2 \in [\gamma_1, 1]$;
- the optimal value of γ_3 seems to be close to 1. This means that the solutions found by solving the auxiliary clustering problem are very close to the local minimizers of the clustering problem; and
- the values of α demonstrate that the use of the optimal values (or their estimations) of parameters leads to a significant decrease in computational effort. This decrease becomes even more significant as the number of clusters and the size of a data set increase.

To conclude, small values of $\gamma_1,\ \gamma_2$ and large values of γ_3 will increase computational time considerably in large data sets without any significant improvement in the quality of the solution to the clustering problem. Therefore, in the implementations of incremental clustering algorithms, we recommend to select the parameters $\gamma_j,\ j = 1, 2, 3$ as follows:

- for small and extra small data sets: γ_1 and γ_2 lie in the interval $[0.4, 0.6]$ while $\gamma_3 \in [1.1, 2]$;

- for medium sized data sets: we can set $\gamma_1 \in [0.55, 0.65]$, $\gamma_2 \in [0.75, 0.85]$, and $\gamma_3 \in [1.01, 1.05]$;
- for large data sets: we set $\gamma_1 \in [0.90, 0.95]$, $\gamma_2 \in [0.925, 0.975]$, and $\gamma_3 \in [1.005, 1.0075]$; and
- for extra large data sets: we set $\gamma_1 \in [0.975, 0.995]$, $\gamma_2 \in [0.99, 0.999]$, and $\gamma_3 = 1.001$.

Note that these values are not optimal and can only be considered as recommended values.