Chapter 8
Nonsmooth Optimization Based Clustering Algorithms

8.1 Introduction

In Chap. 4, we formulated different optimization models of the clustering problem. Using these models, one can apply various heuristics or optimization methods to solve clustering problems. Algorithms considered in this chapter are based on the NSO model of these problems. More specifically, we consider incremental clustering algorithms where at each iteration the clustering and the auxiliary clustering problems are solved by applying either heuristics like the k-means or NSO algorithms [19, 22, 24, 26, 29, 33, 36, 170, 171].

We start with the description of the modified global k-means algorithm in Sect. 8.2. This algorithm is an improvement of the GKM. The main difference between these two algorithms is that the GKM uses only data points to find starting cluster centers whereas the modified global k-means algorithm solves the auxiliary clustering problem to compute them. In Sect. 8.3, we describe a further improvement of the modified global k-means algorithm called the fast modified global k-means algorithm. In addition, we discuss various procedures to reduce the computational complexity of the modified global k-means algorithm.

Then, we introduce three incremental clustering algorithms where the LMBM, the DGM, and the HSM are used to solve the clustering and the auxiliary clustering problems. More precisely, the limited memory bundle method for clustering is described in Sect. 8.4; the discrete gradient clustering algorithm is presented in Sect. 8.5; and the smooth incremental clustering algorithm is given in Sect. 8.6.

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_8

8.2 Modified Global *k*-Means Algorithm

In this section, we present the *modified global k-means algorithm* (MGKM) to solve the clustering problem (7.2) where the similarity measure d_2 is used [19, 21]. The algorithm is the modified version of the GKM, and it is based on the incremental approach. The flowchart of the MGKM is illustrated in Fig. 8.1.

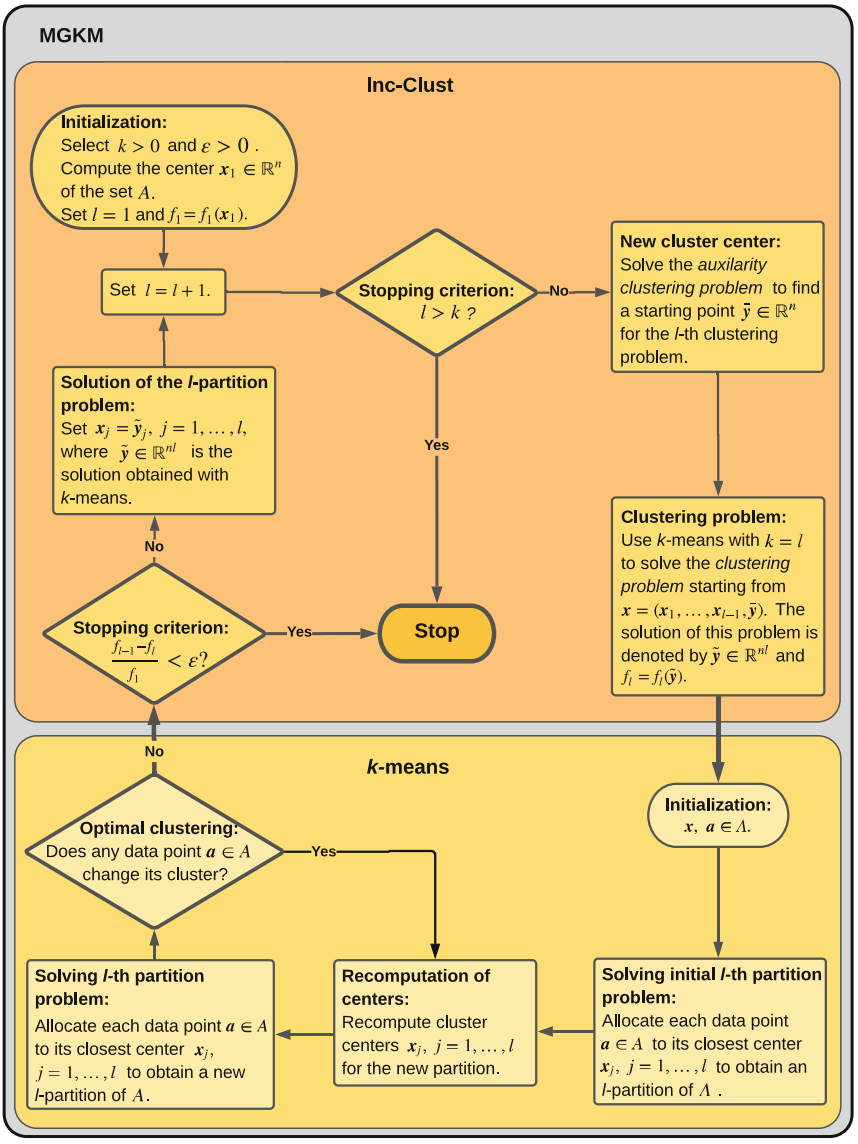

Fig. 8.1 Modified global *k*-means algorithm (MGKM)

The MGKM starts with the computation of the centroid of the whole data set. Then a new cluster center is added at each iteration. More precisely, the auxiliary clustering problem (7.4) is solved to compute a starting point for the lth center. The new center together with the previous $l-1$ cluster centers is taken as a starting point for solving the lth partition problem. The k-means Algorithm 5.1 is applied starting from this point to find the lth partition of the data set.

Assume that $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1})$, $l \geq 2$, be a solution to the $(l-1)$th clustering problem. Let $p = 2$. Recall the sets $\bar{S}_1$ and $\bar{S}_2$ defined in (7.7) and (7.8), respectively. Take any $\boldsymbol{y} \in \bar{S}_2$ and consider the sets $\bar{B}_{12}(\boldsymbol{y})$ and $\bar{B}_3(\boldsymbol{y})$ given in (7.9). The algorithm for finding a starting point for the lth cluster center involves Algorithm 5.1 and proceeds as follows.

Algorithm 8.1 Finding a starting point

Input: Data set A and the solution $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1})$ to the $(l-1)$th clustering problem, $l \geq 2$.
Output: Starting point for the lth cluster center.

1: For each $\boldsymbol{a} \in \bar{S}_2 \cap A$, compute the set $\bar{B}_3(\boldsymbol{a})$, its center $\boldsymbol{c}$ and the value $\bar{f}_{l,\boldsymbol{a}} = \bar{f}_l(\boldsymbol{c})$ of the auxiliary cluster function $\bar{f}_l$ at the point $\boldsymbol{c}$.

2: Compute

$$\bar{f}_{l,\min} = \min_{\boldsymbol{a} \in \bar{S}_2 \cap A} \bar{f}_{l,\boldsymbol{a}}, \quad \bar{\boldsymbol{a}} = \operatorname*{argmin}_{\boldsymbol{a} \in \bar{S}_2 \cap A} \bar{f}_{l,\boldsymbol{a}},$$

and select the corresponding center $\bar{\boldsymbol{c}}$.

3: Compute the set $\bar{B}_3(\bar{\boldsymbol{c}})$ and its center.

4: Reassign data points and update the center of the set $\bar{B}_3(\bar{\boldsymbol{c}})$ until no more points escape or return to this set. Let $\bar{\boldsymbol{y}}$ be the final value of $\bar{\boldsymbol{c}}$. Then $\bar{\boldsymbol{y}}$ is a starting point for the lth cluster center.

In Steps 1 and 2 of Algorithm 8.1, a starting point is found to minimize the auxiliary cluster function $\bar{f}_l$, given in (7.5). This point is chosen among all data points that can attract at least one data point (see Step 1). For each such data point $\boldsymbol{a}$, the set $\bar{B}_3(\boldsymbol{a})$ and its center are computed. Then the function $\bar{f}_l$ is evaluated at these centers, and the center that provides the lowest value of the function $\bar{f}_l$ is selected as a starting point to minimize the function $\bar{f}_l$.

In Step 4 of Algorithm 8.1, we apply Algorithm 5.1 to minimize the auxiliary cluster function $\bar{f}_l$. In this case the first $l-1$ cluster centers are fixed and only the lth cluster center is updated at each iteration.

Remark 8.1 Algorithm 8.1 is a special case of Algorithm 7.2 when we select $\gamma_1 = 0$ and $\gamma_2 = 1$.

Proposition 8.1 *Let $\bar{\boldsymbol{y}}$ be a cluster center generated by Algorithm 8.1. Then this point is a local minimizer of the auxiliary cluster function $\bar{f}_l$.*

Proof Recall the sets $B_i(\boldsymbol{y}),\ i = 1, 2, 3$ defined in (4.30). Since $\bar{\boldsymbol{y}}$ is a cluster center we have $B_2(\bar{\boldsymbol{y}}) = \emptyset$. This is due to the fact that in the hard clustering problem, each data point belongs to only one cluster. Then the function (7.5) can be rewritten as

$$\bar{f}_l(\bar{\boldsymbol{y}}) = \frac{1}{m}\left(\sum_{\boldsymbol{a}\in B_1(\bar{\boldsymbol{y}})} r_{l-1}^{\boldsymbol{a}} + \sum_{\boldsymbol{a}\in B_3(\bar{\boldsymbol{y}})} d_2(\bar{\boldsymbol{y}}, \boldsymbol{a})\right).$$

It is clear that $\bar{\boldsymbol{y}}$ is a minimum point of the convex function

$$\varphi(\boldsymbol{x}) = \frac{1}{m}\sum_{\boldsymbol{a}\in B_3(\bar{\boldsymbol{y}})} d_2(\boldsymbol{x}, \boldsymbol{a}),$$

that is $\varphi(\bar{\boldsymbol{y}}) \le \varphi(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^n$. There exists $\varepsilon > 0$ such that

$$d_2(\boldsymbol{y}, \boldsymbol{a}) > r_{l-1}^{\boldsymbol{a}} \quad \text{for all} \quad \boldsymbol{a} \in B_1(\bar{\boldsymbol{y}}) \quad \text{and} \quad \text{for all} \quad \boldsymbol{y} \in B(\bar{\boldsymbol{y}}; \varepsilon), \quad \text{and}$$
$$d_2(\boldsymbol{y}, \boldsymbol{a}) < r_{l-1}^{\boldsymbol{a}} \quad \text{for all} \quad \boldsymbol{a} \in B_3(\bar{\boldsymbol{y}}) \quad \text{and} \quad \text{for all} \quad \boldsymbol{y} \in B(\bar{\boldsymbol{y}}; \varepsilon).$$

Then for any $\boldsymbol{y} \in B(\bar{\boldsymbol{y}}; \varepsilon)$ we have

$$\begin{aligned}
\bar{f}_l(\boldsymbol{y}) &= \frac{1}{m}\left(\sum_{\boldsymbol{a}\in B_1(\bar{\boldsymbol{y}})} r_{l-1}^{\boldsymbol{a}} + \sum_{\boldsymbol{a}\in B_3(\bar{\boldsymbol{y}})} d_2(\boldsymbol{y}, \boldsymbol{a})\right) \\
&= \frac{1}{m}\sum_{\boldsymbol{a}\in B_1(\bar{\boldsymbol{y}})} r_{l-1}^{\boldsymbol{a}} + \varphi(\boldsymbol{y}) \\
&\ge \frac{1}{m}\sum_{\boldsymbol{a}\in B_1(\bar{\boldsymbol{y}})} r_{l-1}^{\boldsymbol{a}} + \varphi(\bar{\boldsymbol{y}}) \\
&= \bar{f}_l(\bar{\boldsymbol{y}}).
\end{aligned}$$

Therefore, $\bar{f}_l(\boldsymbol{y}) \ge \bar{f}_l(\bar{\boldsymbol{y}})$ for all $\boldsymbol{y} \in B(\bar{\boldsymbol{y}}; \varepsilon)$. This completes the proof. □

Next, we give the step by step form of the MGKM.

Algorithm 8.2 Modified global k-means algorithm (MGKM)

Input: Data set A, the number of clusters k to be computed and a tolerance $\varepsilon \geq 0$.
Output: The l-partition of the set A with $l = 1, \ldots, k$.

1: *(Initialization)* Compute the center $\boldsymbol{x}_1 \in \mathbb{R}^n$ of the set A. Let f_1 be the corresponding value of the clustering function (7.3). Set $l = 1$.

2: *(Stopping criterion)* Set $l = l + 1$. If $l > k$, then **stop**—the k-partition problem has been solved.

3: *(Computation of the next cluster center)* Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}$ be the cluster centers for the $(l-1)$th partition problem. Apply Algorithm 8.1 to find a starting point $\bar{\boldsymbol{y}} \in \mathbb{R}^n$ for the lth cluster center.

4: *(Refinement of all cluster centers)* Select $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}, \bar{\boldsymbol{y}})$ as a new starting point, apply the k-means Algorithm 5.1 to solve the clustering problem (7.2) for $k = l$. Let $(\tilde{\boldsymbol{y}}_1, \ldots, \tilde{\boldsymbol{y}}_l)$ be a solution to this problem and f_l be the corresponding value of the clustering function.

5: *(Stopping criterion)* If

$$\frac{f_{l-1} - f_l}{f_1} \leq \varepsilon,$$

then **stop**, otherwise set $\boldsymbol{x}_j = \tilde{\boldsymbol{y}}_j,\ j = 1, \ldots, l$ and go to Step 2.

Algorithm 8.2 has two stopping criteria. The algorithm stops when either it solves all l-partition problems, $l = 1, \ldots, k$ or the stopping criterion in Step 5 is satisfied. Note that $f_l^* = \inf\{f_l(\boldsymbol{x}),\ \boldsymbol{x} \in \mathbb{R}^{nk}\} \geq 0$ for all $l \geq 1$ and the sequence $\{f_l^*\}$ is decreasing, that is,

$$f_{l+1}^* \leq f_l^* \quad \text{for all} \quad l \geq 1.$$

This means that the stopping criterion in Step 5 will be satisfied after a finite number of iterations and therefore, Algorithm 8.2 computes as many clusters as the data set A contains with respect to the tolerance $\varepsilon > 0$. Note that the choice of this tolerance is crucial for Algorithm 8.2: large values of ε can result in the appearance of large clusters whereas small values can lead to small and artificial clusters.

In Step 3 of Algorithm 8.2, a starting point for the lth cluster center is computed. This is done by applying Algorithm 8.1 and minimizing the auxiliary cluster function. This algorithm requires the calculation of the distance or affinity matrix of the data set A. The matrix can be computed before the application of Algorithm 8.1 in small and medium sized data sets. However, it cannot be done for large data sets as such a matrix is too big to be stored in the memory of a computer. This means that in the latter case, the affinity matrix is computed at each iteration of the MGKM.

8.3 Fast Modified Global k-Means Algorithm

Algorithm 8.2 is time-consuming in large data sets as it requires the computation of the affinity matrix at each iteration. The *fast modified global k-means algorithm* (FMGKM) [29] is an improved version of Algorithm 8.2 and does not rely on the affinity matrix to compute starting points. Instead, the FMGKM uses some weights within the auxiliary cluster function for generating starting points from different parts of the data set. This leads to the elimination of computing or sorting the whole affinity matrix and therefore, to the reduction of computational effort and the memory usage. The flowchart of the FMGKM is similar to that of the MGKM given in Fig. 8.1.

Next, we describe the FMGKM. Let

$$U = \{u_1, \ldots, u_s\}$$

be a finite set of positive numbers. For $u \in U$, the auxiliary cluster function $\bar{f}_l$, given in (7.5), is modified as follows:

$$\bar{f}_l^u(\boldsymbol{y}) = \frac{1}{m} \sum_{\boldsymbol{a} \in A} \min \left\{ r_{l-1}^{\boldsymbol{a}}, u\, d_2(\boldsymbol{y}, \boldsymbol{a}) \right\}. \tag{8.1}$$

If $u = 1$, then $\bar{f}_l^u(\boldsymbol{y}) = \bar{f}_l(\boldsymbol{y})$ for all $\boldsymbol{y} \in \mathbb{R}^n$. Take $u \in U$ and define the set

$$\tilde{S}_2^u = \left\{ \boldsymbol{y} \in \mathbb{R}^n : r_{l-1}^{\boldsymbol{a}} > u\, d_2(\boldsymbol{y}, \boldsymbol{a}) \ \text{ for some } \ \boldsymbol{a} \in A \right\},$$

and for any $\boldsymbol{y} \in \tilde{S}_2^u$ consider the set

$$\tilde{B}_3^u(\boldsymbol{y}) = \left\{ \boldsymbol{a} \in A : r_{l-1}^{\boldsymbol{a}} > u\, d_2(\boldsymbol{y}, \boldsymbol{a}) \right\}.$$

The set $\tilde{S}_2^u$ is similar to the set $\bar{S}_2$ defined in (7.8) and the set $\tilde{B}_3^u(\boldsymbol{y})$ is similar to the set $\bar{B}_3(\boldsymbol{y})$ described in (7.9). The set $\tilde{B}_3^u(\boldsymbol{y})$ contains all data points attracted by the point $\boldsymbol{y} \in \tilde{S}_2^u$ with a given weight $u > 0$.

The following algorithm is a modified version of Algorithm 8.1 and computes a starting point for the lth cluster center.

Algorithm 8.3 Finding a starting point

Input: Data set A, the solution $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1})$ to the $(l-1)$th clustering problem, $l \geq 2$ and the set $U = \{u_1, \ldots, u_s\}$.
Output: Starting point for the lth cluster center.

1: Set $t = 1$.
2: Take $u_t \in U$. For each $\boldsymbol{a} \in \tilde{S}_2^{u_t} \cap A$ compute the set $\tilde{B}_3^{u_t}(\boldsymbol{a})$, its center $\boldsymbol{c}$ and the value $\bar{f}_{l,\boldsymbol{a}}^{u_t} = \bar{f}_l^{u_t}(\boldsymbol{c})$ of the function $\bar{f}_l^{u_t}$ at the point $\boldsymbol{c}$.
3: Compute

$$\bar{f}_{l,\min}^{u_t} = \min_{\boldsymbol{a} \in \tilde{S}_2^{u_t} \cap A} \bar{f}_{l,\boldsymbol{a}}^{u_t} \quad \text{and} \quad \bar{\boldsymbol{a}} = \operatorname*{argmin}_{\boldsymbol{a} \in \tilde{S}_2^{u_t} \cap A} \bar{f}_{l,\boldsymbol{a}}^{u_t},$$

and select the corresponding center $\bar{\boldsymbol{c}}_t$.
4: Compute the set $\tilde{B}_3^{u_t}(\bar{\boldsymbol{c}}_t)$ and its center.
5: Reassign data points until no more points escape or return to this set. Let $\bar{\boldsymbol{y}}(u_t)$ be the final value for the center $\bar{\boldsymbol{c}}_t$. Compute the value $\bar{f}_{l,t}$ of the auxiliary function $\bar{f}_l$, given in (7.5), at the point $\bar{\boldsymbol{y}}(u_t)$.
6: Set $t = t + 1$. If $t \leq s$, then go to Step 2.
7: Compute

$$\bar{f}_{l,\min} = \min_{t=1,\ldots,s} \bar{f}_{l,t},$$

and let

$$\bar{f}_{l,t_0} = \bar{f}_{l,\min} \quad \text{for} \quad t_0 \in \{1, \ldots, s\}.$$

Set $\bar{\boldsymbol{y}} = \bar{\boldsymbol{y}}(u_{t_0})$ as a starting point for the lth cluster center.

In order to solve the auxiliary clustering problem (7.4) in Step 5 of Algorithm 8.3, we apply Algorithm 5.1. Here, only one cluster center is updated, other cluster centers are known from previous iterations and they are fixed. Since Algorithm 5.1 is only able to find local solutions to this problem more than one starting points are used to compute high quality solutions.

Starting points are computed using the function (8.1) with different values of u. If u is sufficiently small, then the starting point will be close to other cluster centers, most likely near the center of the largest cluster. If $u = 1$, then we get the same starting point as in the case of Algorithm 8.1. As the value of u is increased the starting points become more isolated data points. This leads to the finding of starting points from different parts of the data set.

The FMGKM is presented in step by step form as follows.

Algorithm 8.4 Fast modified global k-means algorithm (FMGKM)

Input: Data set A, the number of clusters k to be computed and a tolerance $\varepsilon > 0$.
Output: The l-partition of the set A with $l = 1, \ldots, k$.

1: *(Initialization)* Compute the center $\boldsymbol{x}_1 \in \mathbb{R}^n$ of the set A. Let f_1 be the corresponding value of the clustering function (7.3). Set $l = 1$.

2: *(Stopping criterion)* Set $l = l + 1$. If $l > k$, then **stop**—the k-partition problem has been solved.

3: *(Computation of the next cluster center)* Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}$ be the cluster centers for the $(l-1)$th partition problem. Apply Algorithm 8.3 to find a starting point $\bar{\boldsymbol{y}} \in \mathbb{R}^n$ for the lth cluster center.

4: *(Refinement of all cluster centers)* Select $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}, \bar{\boldsymbol{y}})$ as a new starting point, apply Algorithm 5.1 to solve the clustering problem (7.2) for $k = l$. Let $(\tilde{\boldsymbol{y}}_1, \ldots, \tilde{\boldsymbol{y}}_l)$ be a solution to this problem and f_l be the corresponding value of the clustering function.

5: *(Stopping criterion)* If

$$\frac{f_{l-1} - f_l}{f_1} < \varepsilon,$$

then **stop**, otherwise set $\boldsymbol{x}_j = \tilde{\boldsymbol{y}}_j,\ j = 1, \ldots, l$ and go to Step 2.

The most time-consuming step in Algorithm 8.4 is Step 3, where Algorithm 8.3 is applied to minimize the auxiliary cluster function (8.1) for different $u \in U$ and to find the starting point for the lth cluster center. In its turn, Step 2 of Algorithm 8.3 is time-consuming as in this step, clusters are computed for each data point $\boldsymbol{a} \in \tilde{S}_2^{u_t} \cap A$. This requires the partial computation of the affinity matrix. In addition, centers of those clusters and values of the function $\bar{f}_l^u$ at these centers need to be computed. Since for each data point only one center is obtained the complexity of the computation of the function $\bar{f}_l^u$ is the same as the complexity of the computation of the affinity matrix.

In [29], two different approaches are introduced to reduce the computational complexity of Step 2 in Algorithm 8.3. Both approaches exploit the incremental nature of the algorithm. In these approaches a matrix, consisting of the distances between data points and cluster centers is used instead of the affinity matrix. Since the number of clusters is significantly less than the number of data points the former matrix is much smaller than the latter one. More precisely, in these approaches data points which are close to cluster centers from the $(l-1)$th partition are excluded. Therefore, these data points are removed from the list of points which can attract large clusters and also from the list of points which can be attracted by non-excluded data points.

Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}$, $l \geq 2$ be known cluster centers. Assume v_{iq} is the squared Euclidean distance between the data point $\boldsymbol{a}_i$, $i = 1, \ldots, m$ and the cluster center $\boldsymbol{x}_q$, $q = 1, \ldots, l-1$, that is

$$v_{iq} = d_2(\boldsymbol{a}_i, \boldsymbol{x}_q).$$

Let $\boldsymbol{v}_q = \left(v_{1q}, \ldots, v_{mq}\right) \in \mathbb{R}^m$, $q = 1, \ldots, l-1$. Consider a matrix V of the size $m \times (l-1)$, whose columns are vectors $\boldsymbol{v}_q$, $q = 1, \ldots, l-1$:

$$V = [\boldsymbol{v}_{iq}], \quad i = 1, \ldots, m, \ q = 1, \ldots, l-1.$$

Let also $\boldsymbol{r} = (r_{l-1}^1, \ldots, r_{l-1}^m)$ be a vector of m components where r_{l-1}^i is the squared Euclidean distance between the data point $\boldsymbol{a}_i$ and its cluster center in the $(l-1)$th partition (see (7.6)). Note that the matrix V and the vector $\boldsymbol{r}$ are available after the $(l-1)$th iteration of the incremental clustering algorithm.

Now, we are ready to describe the following two approaches to reduce the computational complexity of Step 2 of Algorithm 8.3.

1. *Reduction of the number of pairwise distance computations.* Let a data point $\boldsymbol{a}_j \in A$ be given and $\boldsymbol{x}_{q(j)}$ be its cluster center. Here $q(j) \in \{1, \ldots, l-1\}$. For a given $u \in U$ and the data point $\boldsymbol{a}_i$ if

$$\left(1 + \frac{1}{\sqrt{u}}\right)^2 r_{l-1}^j \leq v_{iq(j)},$$

then $\boldsymbol{a}_j \notin \tilde{B}_3^u(\boldsymbol{a}_i)$. Indeed, we have

$$\|\boldsymbol{a}_i - \boldsymbol{a}_j\| \geq \|\boldsymbol{a}_i - \boldsymbol{x}_{q(j)}\| - \|\boldsymbol{a}_j - \boldsymbol{x}_{q(j)}\| \geq (1/\sqrt{u})\|\boldsymbol{a}_j - \boldsymbol{x}_{q(j)}\|.$$

Thus, we get $ud_2(\boldsymbol{a}_i, \boldsymbol{a}_j) \geq r_{l-1}^j$, and therefore $\boldsymbol{a}_j \notin \tilde{B}_3^u(\boldsymbol{a}_i)$. This condition allows us to reduce the number of pairwise distance computations. This reduction becomes substantial as the number of clusters increases. Define the set

$$\tilde{R}^u(\boldsymbol{a}_i) = \left\{\boldsymbol{a}_j \in A : \left(1 + \frac{1}{\sqrt{u}}\right)^2 r_{l-1}^j > v_{iq(j)}\right\}.$$

It is clear that

$$\tilde{B}_3^u(\boldsymbol{a}_i) \subseteq \tilde{R}^u(\boldsymbol{a}_i).$$

The set $\tilde{R}^u(\boldsymbol{a}_i)$ can be used instead of the set A to compute the value of the function $\bar{f}_l^u$ in Step 2 of Algorithm 8.3. In this case, one may not get the exact value of this function; however, it gives a good approximation to the exact value. Furthermore, take

$$w \in \left(1, \left(1 + \frac{1}{\sqrt{u}}\right)^2\right],$$

and consider the set

$$\tilde{R}_w^u(\boldsymbol{a}_i) = \left\{\boldsymbol{a}_j \in A : \ wr_{l-1}^j > d_2(\boldsymbol{a}_i, \boldsymbol{a}_j)\right\}.$$

Then replace A by $\tilde{R}_w^u(\boldsymbol{a}_i)$ for the computation of the function $\bar{f}_l^u$. This will further reduce the amount of computations in Step 2 of Algorithm 8.3.

2. *Reduction of the number of starting cluster centers.* This approach is similar to that of considered in Algorithm 7.4. More specifically, data points which are very close to previous cluster centers are not considered for being starting points to minimize the auxiliary cluster function (8.1). At the $(l-1)$th iteration a squared averaged radius

$$\bar{d}_{av}^q = \frac{1}{|A^q|} \sum_{\boldsymbol{a} \in A^q} d_2(\boldsymbol{x}_q, \boldsymbol{a}),$$

and a squared maximum radius

$$\bar{d}_{\max}^q = \max_{\boldsymbol{a} \in A^q} d_2(\boldsymbol{x}_q, \boldsymbol{a})$$

of each cluster $A^q, \ q = 1, \ldots, l-1$ are computed. Consider the numbers

$$\alpha_q = \frac{\bar{d}_{\max}^q}{\bar{d}_{av}^q} \geq 1 \quad \text{and} \quad \beta_q = \varepsilon(\alpha_q - 1),$$

where $\varepsilon > 0$ is a sufficiently small number. Let

$$\gamma_{ql} = 1 + \beta_q(l-1), \quad q = 1, \ldots, l-1.$$

It is clear that $\gamma_{ql} \geq 1, \ q = 1, \ldots, l-1$. Define the following subset of the cluster A^q:

$$\bar{A}^q = \left\{\boldsymbol{a} \in A^q : \ \gamma_{ql}\bar{d}_{av}^q \leq d_2(\boldsymbol{x}_q, \boldsymbol{a})\right\}.$$

In other words, the set $\bar{A}^q$ is obtained from the cluster A^q by removing all points for which $d_2(\boldsymbol{x}_q, \boldsymbol{a}) < \gamma_q \bar{d}_{av}^q$. Since in the incremental approach the clusters are becoming more stable as the number l increases it follows that the numbers γ_{ql} are increased as l increases. Define the set

$$\bar{A} = \bigcup_{q=1}^{l-1} \bar{A}^q,$$

and consider only data points $\boldsymbol{a} \in \bar{A}$ as the candidates to be starting points for minimizing the auxiliary cluster function $\bar{f}_l$.

Summarizing, Steps 2 and 3 of Algorithm 8.3 can be rewritten as follows:

2': for each $\boldsymbol{a} \in \tilde{S}_2^{u_t} \cap \bar{A}$ compute the set $\tilde{B}_3^{u_t}(\boldsymbol{a})$, its center $\boldsymbol{c}$, and the value $\bar{f}_{l,\boldsymbol{a}}^{u_t} = \bar{f}_l^{u_t}(\boldsymbol{c})$ of the function $\bar{f}_l^{u_t}$ at the point $\boldsymbol{c}$ over the set $\tilde{R}_w^u(\boldsymbol{a})$.

3': compute

$$\bar{f}_{l,\min}^{u_t} = \min_{\boldsymbol{a} \in \tilde{S}_2^{u_t} \cap \bar{A}} \bar{f}_{l,\boldsymbol{a}}^{u_t} \quad \text{and} \quad \bar{\boldsymbol{a}} = \operatorname*{argmin}_{\boldsymbol{a} \in \tilde{S}_2^{u_t} \cap \bar{A}} \bar{f}_{l,\boldsymbol{a}}^{u_t},$$

and the corresponding center $\bar{\boldsymbol{c}}$.

The use of these two schemes allows us to significantly reduce the computational complexity of Algorithm 8.4 and accelerate its convergence.

8.4 Limited Memory Bundle Method for Clustering

In this section, we present the *limited memory bundle method for clustering* (LMB-CLUST) [171]. The LMB-CLUST has been developed specifically to solve clustering problems in large data sets. The algorithm combines two different approaches to solve the clustering problem when the squared Euclidian distance d_2 is used as a similarity measure. The MSINC-CLUST is used to solve the clustering problem globally and the LMBM is applied at each iteration of the algorithm to solve both the clustering problem (7.2) and the auxiliary clustering problem (7.4). The flowchart of the LMB-CLUST is given in Fig. 8.2.

The LMBM, given in Fig. 3.5, is originally developed for solving general large-scale nonconvex NSO problems. Here, this method is slightly modified to be better suited for solving the clustering and the auxiliary clustering problems. In particular, a nonmonotone line search is used to find step sizes t_L^h and t_R^h. In addition, different stopping tolerances are utilized for different problems. That is, the tolerance ε is set to be relatively large for the auxiliary clustering problem (7.4)—since this problem need not to be solved very accurately—and smaller for the clustering problem (7.2).

Next, we give the modified version of the LMBM in its step by step form. We use $\boldsymbol{x}_1$ for the starting point; $\varepsilon_c > 0$ for the stopping tolerance; ε_L and ε_R for line search parameters; γ for the distance measure parameter; $\hat{m}_c$ for the maximum number of stored correction vectors used to form limited memory matrix updates; $t_{\max}$ is an upper bound for serious steps; C is a control parameter for the length of the direction vector. We also use i_{type} to show the type of the problem, that is:

- $i_{\text{type}} = 0$: the auxiliary clustering problem (7.4); and
- $i_{\text{type}} = 1$: the clustering problem (7.2).

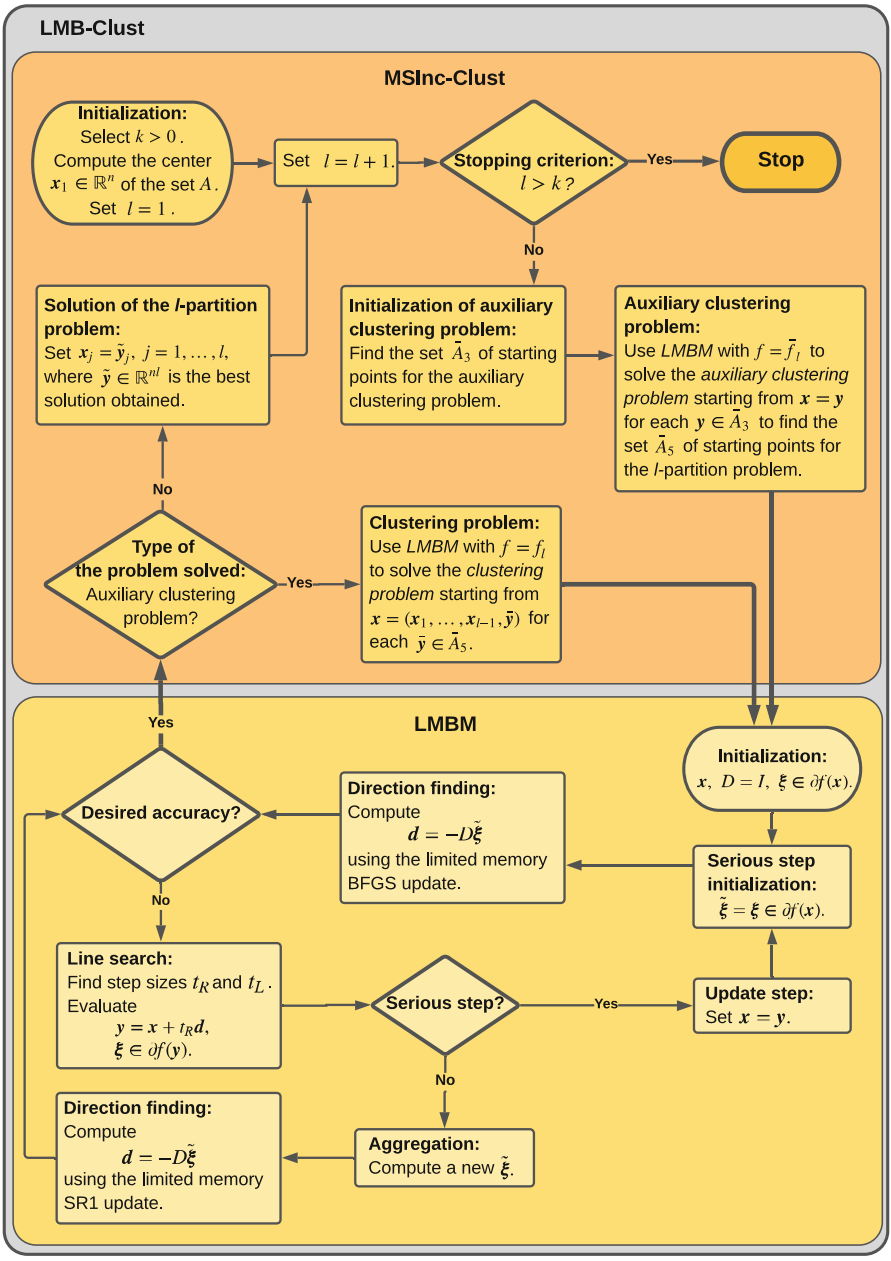

Fig. 8.2 Limited memory bundle method for clustering (LMB-CLUST)

In both cases, the objective function is denoted by f and the number of variables in the optimization problem is denoted by n. Hence $f = \bar{f}_l$ and $n = n$ for the auxiliary clustering problem and $f = f_l$ and $n = nl$ for the lth clustering problem.

Algorithm 8.5 Modified limited memory bundle algorithm

Input: $\boldsymbol{x}_1 \in \mathbb{R}^n$, $\varepsilon_c > 0$, $\varepsilon_L \in (0, 1/2)$, $\varepsilon_R \in (\varepsilon_L, 1/2)$, $t_{\max} > 1$, $\gamma > 0$, $C > 0$, $\hat{m}_c \geq 3$ and $i_{\text{type}} \in \{0, 1\}$.
Output: Clarke stationary point $\boldsymbol{x}_h$.

1: *(Initialization)* Set $\boldsymbol{y}_1 = \boldsymbol{x}_1$ and $\beta_1 = 0$. Compute $f_1 = f(\boldsymbol{x}_1)$ and $\boldsymbol{\xi}_1 \in \partial f(\boldsymbol{x}_1)$. If $i_{\text{type}} = 1$, then set $\varepsilon = \varepsilon_c$. Otherwise, set $\varepsilon = 10^3 \varepsilon_c$. Set $h = 1$.

2: *(Serious step initialization)* Set $\tilde{\boldsymbol{\xi}}_h = \boldsymbol{\xi}_h$, $\tilde{\beta}_h = 0$ and $m = h$.

3: *(Direction finding)* If $h = 1$, set $\boldsymbol{d}_1 = -\boldsymbol{\xi}_1$. Otherwise, compute

$$\boldsymbol{d}_h = -D_h \tilde{\boldsymbol{\xi}}_h$$

by the L-BFGS update if $m = h$ (use $\hat{m}_h = \min\{h - 1, \hat{m}_c\}$ correction vectors in U_h and S_h) and by the L-SR1 update, otherwise.

4: *(Stopping criterion)* Compute $w_h = -\tilde{\boldsymbol{\xi}}_h^T \boldsymbol{d}_h + 2\tilde{\beta}_h$. If $w_h < \varepsilon$, then **stop** with $\boldsymbol{x}_h$ as the final solution.

5: *(Line search)* Set the scaling parameter θ_h for the length of the direction vector as $\theta_h = \min\{1, C/\|\boldsymbol{d}_h\|\}$. Use a nonmonotone line search to determine the step sizes $t_R^h \in (0, t_{\max}]$ and $t_L^h \in [0, t_R^h]$ and set the corresponding values

$$\boldsymbol{x}_{h+1} = \boldsymbol{x}_h + t_L^h \theta_h \boldsymbol{d}_h, \qquad f_{h+1} = f(\boldsymbol{x}_{h+1}), \quad \text{and}$$
$$\boldsymbol{y}_{h+1} = \boldsymbol{x}_h + t_R^h \theta_h \boldsymbol{d}_h, \qquad \boldsymbol{\xi}_{h+1} \in \partial f(\boldsymbol{y}_{h+1}).$$

Set $\boldsymbol{u}_h = \boldsymbol{\xi}_{h+1} - \boldsymbol{\xi}_m$ and $\boldsymbol{s}_h = \boldsymbol{y}_{h+1} - \boldsymbol{x}_h = t_R^h \theta_h \boldsymbol{d}_h$ and append these values to U_h and S_h. If the modified serious step condition

$$t_R^h = t_L^h > 0 \quad \text{and} \quad f(\boldsymbol{y}_{h+1}) \leq \max_{i \in M} f(\boldsymbol{x}_i) - \varepsilon_L t_R^h w_h,$$

where $M \subseteq \{l : \boldsymbol{x}_{l+1} = \boldsymbol{x}_l + t_R^l \theta_l \boldsymbol{d}_l\}$ such that M contains at most the ten greatest indices l, is satisfied, then set $\beta_{h+1} = 0$, $h = h + 1$ and go to Step 2. Otherwise, calculate the locality measure β_{h+1} by

$$\beta_{h+1} = \max\left\{|f(\boldsymbol{x}_h) - f(\boldsymbol{y}_{h+1}) + \boldsymbol{\xi}_{h+1}^T (\boldsymbol{y}_{h+1} - \boldsymbol{x}_h)|, \gamma \|\boldsymbol{y}_{h+1} - \boldsymbol{x}_h\|^2\right\}.$$

6: *(Aggregation)* Determine multipliers $\lambda_i^h \geq 0$ for all $i \in \{1, 2, 3\}$, $\sum_{i=1}^3 \lambda_i^h = 1$ that minimize the function $\varphi(\lambda_1, \lambda_2, \lambda_3)$ given in (3.9), where D_h is calculated by the same updating formula as in Step 3. Compute $\tilde{\boldsymbol{\xi}}_{h+1}$ and $\tilde{\beta}_{h+1}$ as

$$\tilde{\boldsymbol{\xi}}_{h+1} = \lambda_1^h \boldsymbol{\xi}_m + \lambda_2^h \boldsymbol{\xi}_{h+1} + \lambda_3^h \tilde{\boldsymbol{\xi}}_h \quad \text{and} \quad \tilde{\beta}_{h+1} = \lambda_2^h \beta_{h+1} + \lambda_3^h \tilde{\beta}_h.$$

Set $h = h + 1$ and go to Step 3.

The convergence properties of the LMBM are given in Sect. 3.4. Here, we recall the most important results in light of the clustering problem. Note that

Assumptions 3.1–3.3, needed to prove the global convergence of the LMBM, are trivially satisfied for both the clustering and the auxiliary clustering problems.

Proposition 8.2 *Assume that $\varepsilon_c = 0$. If the* LMBM *terminates after a finite number of iterations, say at the iteration h, then the point $\boldsymbol{x}_h$ is a Clarke stationary point of the (auxiliary) clustering problem.*

Proposition 8.3 *Assume that $\varepsilon_c = 0$. Every accumulation point $\bar{\boldsymbol{x}}$ generated by the* LMBM *is a Clarke stationary point of the (auxiliary) clustering problem.*

Remark 8.2 The LMBM terminates in a finite number of steps if we choose $\varepsilon_c > 0$.

Next, we describe the LMB-CLUST and give its step by step algorithm. Since the problem (7.2) is nonconvex it is important to select favorable starting points before applying a local search method like the LMBM to solve it. The LMB-CLUST uses the MSINC-CLUST for solving the clustering problem globally and the LMBM is applied at each iteration of the MSINC-CLUST to solve both the problems (7.2) and (7.4).

Algorithm 8.6 Limited memory bundle method for clustering (LMB-CLUST)

Input: Data set A and the number of clusters k to be computed.
Output: The l-partition of the set A with $l = 1, \ldots, k$.

1: *(Initialization)* Compute the center $\boldsymbol{x}_1 \in \mathbb{R}^n$ of the set A. Set $l = 1$.

2: *(Stopping criterion)* Set $l = l + 1$. If $l > k$, then **stop**—the k-partition problem has been solved.

3: *(Computation of a set of starting points for the auxiliary clustering problem)* Apply Algorithm 7.2 to find the set $\bar{A}_3 \subset \mathbb{R}^n$ of starting points for the auxiliary clustering problem (7.4).

4: *(Computation of a set of starting points for the clustering problem)* For each $\boldsymbol{y} \in \bar{A}_3$ apply Algorithm 8.5 with $i_{\text{type}} = 0$ to solve the auxiliary clustering problem (7.4) and find $\bar{A}_5$, a set of starting points for the lth partition problem (7.2).

5: *(Computation of a set of cluster centers)* For each $\bar{\boldsymbol{y}} \in \bar{A}_5$ apply Algorithm 8.5 with $i_{\text{type}} = 1$ to solve the clustering problem (7.2) starting from the point $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}, \bar{\boldsymbol{y}})$ and find a solution $(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_l)$. Denote by $\bar{A}_6$ a set of all such solutions.

6: *(Computation of the best solution)* Compute

$$f_l^{\min} = \min \left\{ f_l(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_l) \ : (\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_l) \in \bar{A}_6 \right\},$$

and the collection of cluster centers $(\tilde{\boldsymbol{y}}_1, \ldots, \tilde{\boldsymbol{y}}_l)$ such that

$$f_l(\tilde{\boldsymbol{y}}_1, \ldots, \tilde{\boldsymbol{y}}_l) = f_l^{\min}.$$

7: *(Solution to the lth partition problem)* Set $\boldsymbol{x}_j = \tilde{\boldsymbol{y}}_j, \ j = 1, \ldots, l$ as a solution to the lth partition problem and go to Step 2.

8.5 Discrete Gradient Clustering Algorithm

In this section, we describe the *discrete gradient clustering algorithm* (DG-CLUST) to solve the clustering problem (7.2). As mentioned in Sect. 3.8, the underlying optimization solver DGM is a semi-derivative free method for solving nonconvex NSO problems. The DGM does not use subgradients or their approximations but only at the end of the solution process and thus, it can be used to solve problems which are not subdifferentially regular. Therefore, the clustering algorithm based on the DGM can be used to solve clustering problems with the similarity measures d_1 and d_∞, in addition to d_2 based clustering problems.

The flowchart of the DG-CLUST is given in Fig. 8.3. Similar to other optimization based clustering algorithms, the DG-CLUST uses the MSINC-CLUST for solving the clustering problem globally and the DGM is applied at each iteration of the MSINC-CLUST to solve both the problems (7.2) and (7.4).

The flowchart of the DGM with a more detailed description is given in Sect. 3.8. Here, we give this method in its step by step form. Note that we use $\boldsymbol{x}_1$ for the starting point; $\varepsilon > 0$ for the stopping tolerance; and ε_L and ε_R for line search parameters.

As before, we use the following notations: the objective function is denoted by f and n stands for the size of the optimization problem. That is, $f = \bar{f}_l$ and $n = n$ for the auxiliary clustering problem and $f = f_l$ and $n = nl$ for the l-partition problem.

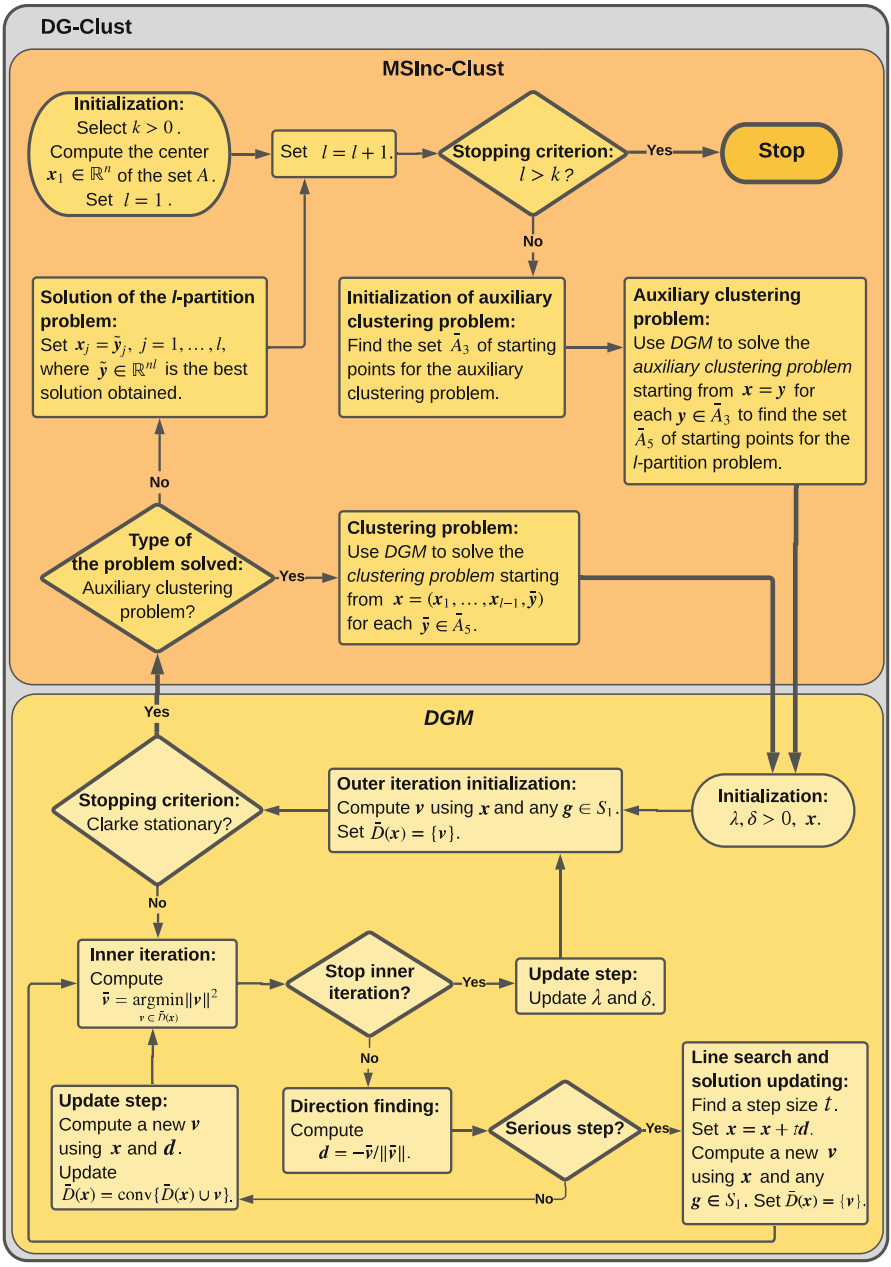

Fig. 8.3 Discrete gradient clustering algorithm (DG-CLUST)

Algorithm 8.7 Discrete gradient method

Input: $\boldsymbol{x}_1 \in \mathbb{R}^n$, $\varepsilon > 0$, $\lambda_1 > 0$, $\delta_1 > 0$, $\alpha \in (0, 1]$, $\varepsilon_L \in (0, 1]$ and $\varepsilon_R \in (0, \varepsilon_L]$.
Output: Final solution $\boldsymbol{x}_h$.

1: *(Outer iteration initialization)* Set $h = 1$.
2: *(Inner iteration initialization)* Set $s = 1$ and $\boldsymbol{x}_{h_s} = \boldsymbol{x}_h$. Choose any $\boldsymbol{g} \in S_s$, $\boldsymbol{w} \in G$. Compute a discrete gradient $\boldsymbol{v}_{h_s} = \boldsymbol{\Gamma}^i(\boldsymbol{x}_{h_s}, \boldsymbol{g}, \boldsymbol{w}, \lambda_h, \alpha)$. Set $\bar{D}(\boldsymbol{x}_{h_s}) = \{\boldsymbol{v}_{h_s}\}$.
3: *(Stopping criterion)* If $\lambda_h < \varepsilon$ and $\delta_h < \varepsilon$, then **stop** with $\boldsymbol{x}_h$ as a final solution.
4: (Minimum norm). Compute the vector

$$\bar{\boldsymbol{v}}_{h_s} = \operatorname*{argmin}_{\boldsymbol{v} \in \bar{D}(\boldsymbol{x}_{h_s})} \|\boldsymbol{v}\|^2.$$

5: *(Inner iteration termination)* If $\|\bar{\boldsymbol{v}}_{h_s}\| \leq \delta_h$, then update λ_{h+1} and δ_{h+1}. Set $\boldsymbol{x}_{h+1} = \boldsymbol{x}_{h_s}$, $h = h + 1$ and go to Step 2.
6: *(Search direction)* Compute the search direction

$$\boldsymbol{d}_{h_s} = -\frac{\bar{\boldsymbol{v}}_{h_s}}{\|\bar{\boldsymbol{v}}_{h_s}\|}.$$

7: If $f(\boldsymbol{x}_{h_s} + \lambda_h \boldsymbol{d}_{h_s}) - f(\boldsymbol{x}_{h_s}) > -\varepsilon_L \lambda_h \|\bar{\boldsymbol{v}}_{h_s}\|$, then go to Step 9.
8: *(Serious step)* Construct $\boldsymbol{x}_{h_{s+1}} = \boldsymbol{x}_{h_s} + t_{h_s} \boldsymbol{d}_{h_s}$, where the step size t_{h_s} is computed as

$$t_{h_s} = \operatorname{argmax} \left\{ t \geq 0 : f(\boldsymbol{x}_{h_s} + t\boldsymbol{d}_{h_s}) - f(\boldsymbol{x}_{h_s}) \leq -\varepsilon_R t \|\bar{\boldsymbol{v}}_{h_s}\| \right\}.$$

Compute a new discrete gradient $\boldsymbol{v}_{h_{s+1}}$ using $\boldsymbol{x}_{h_{s+1}}$ and any $\boldsymbol{g} \in S_1$:

$$\boldsymbol{v}_{h_{s+1}} = \boldsymbol{\Gamma}^i(\boldsymbol{x}_{h_{s+1}}, \boldsymbol{g}, \boldsymbol{w}, \lambda_h, \alpha).$$

Set $\bar{D}(\boldsymbol{x}_{h_{s+1}}) = \{\boldsymbol{v}_{h_{s+1}}\}$, $s = s + 1$ and go to Step 4.
9: *(Null step)* Compute a new discrete gradient $\boldsymbol{v}_{h_{s+1}}$ using $\boldsymbol{x}_{h_s}$ and $\boldsymbol{d}_{h_s}$:

$$\boldsymbol{v}_{h_{s+1}} = \boldsymbol{\Gamma}^i(\boldsymbol{x}_{h_s}, \boldsymbol{d}_{h_s}, \boldsymbol{w}, \lambda_h, \alpha).$$

Update the set

$$\bar{D}(\boldsymbol{x}_{h_{s+1}}) = \operatorname{conv} \left\{ \bar{D}(\boldsymbol{x}_{h_s}) \cup \{\boldsymbol{v}_{h_{s+1}}\} \right\}.$$

Set $\boldsymbol{x}_{h_{s+1}} = \boldsymbol{x}_{h_s}$, $s = s + 1$ and go to Step 4.

The global convergence of Algorithm 8.7 has been studied in Sect. 3.8. Note that assumptions needed to get its convergence are satisfied for both the cluster and the auxiliary cluster functions. Next, we present the step by step description of the DG-CLUST.

Algorithm 8.8 Discrete gradient clustering algorithm (DG-CLUST)

Input: Data set A and the number of clusters k to be computed.
Output: The l-partition of the set A with $l = 1, \ldots, k$.

1: *(Initialization)* Compute the center $\boldsymbol{x}_1 \in \mathbb{R}^n$ of the set A. Set $l = 1$.

2: *(Stopping criterion)* Set $l = l + 1$. If $l > k$, then **stop**—the k-partition problem has been solved.

3: *(Computation of a set of starting points for the auxiliary clustering problem)* Apply Algorithm 7.2 to find the set $\bar{A}_3 \subset \mathbb{R}^n$ of starting points for solving the auxiliary clustering problem (7.4).

4: *(Computation of a set of starting points for the lth cluster center)* Apply Algorithm 8.7 to solve the auxiliary clustering problem (7.4) starting from each point $\boldsymbol{y} \in \bar{A}_3$. This algorithm generates a set $\bar{A}_5$ of starting points for the lth cluster center.

5: *(Computation of a set of cluster centers)* For each $\bar{\boldsymbol{y}} \in \bar{A}_5$ apply Algorithm 8.7 to solve the clustering problem (7.2) starting from the point $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}, \bar{\boldsymbol{y}})$ and find a solution $(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_l)$. Denote by $\bar{A}_6$ a set of all such solutions.

6: *(Computation of the best solution)* Compute

$$f_l^{\min} = \min \left\{ f_l(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_l) : \ (\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_l) \in \bar{A}_6 \right\},$$

and the collection of cluster centers $(\tilde{\boldsymbol{y}}_1, \ldots, \tilde{\boldsymbol{y}}_l)$ such that

$$f_l(\tilde{\boldsymbol{y}}_1, \ldots, \tilde{\boldsymbol{y}}_l) = f_l^{\min}.$$

7: *(Solution to the lth partition problem)* Set $\boldsymbol{x}_j = \tilde{\boldsymbol{y}}_j, \ j = 1, \ldots, l$ as a solution to the l-partition problem and go to Step 2.

Note that the DGM uses only discrete gradients to find an approximate solution to both the clustering and the auxiliary clustering problems. The calculation of discrete gradients can be simplified using a special structure of a problem such as the piecewise separability or piecewise partial separability of the objective functions (see Sect. 2.6.3).

It is proved in Propositions 4.5 and 4.12 that both the cluster function (7.3) and the auxiliary cluster function (7.5) are piecewise separable with the similarity measures d_1, d_2, and d_∞. Therefore, we can simplify the calculations of discrete gradients for both the cluster and the auxiliary cluster functions.

First, we consider the computation of discrete gradients of the cluster function f_k. This function is the special case of the function f, defined in (2.21) as

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} \max_{h \in \mathcal{H}_i} \min_{j \in \mathcal{J}_h} f_{ihj}(\boldsymbol{x}).$$

The cluster function f_k does not depend on the index h and the sets $\mathcal{H}_i$, $i = 1, \ldots, m$ are all singletons. Therefore, for all $h \in \mathcal{H}_i$ we have $\mathcal{J}_h = \{1, \ldots, k\}$ and

$$f_k(\boldsymbol{x}) = \frac{1}{m} \sum_{i=1}^{m} \min_{l=1,\ldots,k} f_{il}(\boldsymbol{x}_l),$$

where $f_{il}(\boldsymbol{x}_l) = d_p(\boldsymbol{x}_l, \boldsymbol{a}_i),\ l = 1, \ldots, k$.

Then the term functions are

$$(x_{lt} - a_{it})^2 \quad \text{for} \quad p = 2, \quad \text{and}$$
$$|x_{lt} - a_{it}| \quad \text{for} \quad p = 1, \infty.$$

Here, $t = 1, \ldots, n,\ l = 1, \ldots, k,\ i = 1, \ldots, m$, and therefore, the total number of such term functions is mnk. Since the function f_k has nk number of variables one needs $nk + 1$ evaluations of this function to compute its one discrete gradient. Then the total number of evaluations of term functions to compute one discrete gradient of f_k is $N_t = mnk(nk + 1)$.

According to the definition of the discrete gradients for a given $i \in \{1, \ldots, nk\}$ we compute values of the function f_k at the following $nk + 1$ points:

$$\boldsymbol{x}, \boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^{i-1}, \boldsymbol{x}^{i+1}, \ldots, \boldsymbol{x}^{nk}.$$

We need the full evaluation of the function f_k only at two points: at $\boldsymbol{x}$ and $\boldsymbol{x}^0$ which requires $2mnk$ calculations of the term functions. Other points from this sequence are obtained from the previous point by changing only one coordinate which is the coordinate of only one cluster center. This means that we need to update only m term functions at points $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{i-1}, \boldsymbol{x}^{i+1}, \ldots, \boldsymbol{x}^{nk}$ and the number of evaluations of the term functions at these points is $m(nk - 1)$. Therefore, the total number of evaluations of term functions for computation of one discrete gradient is $\bar{N}_t = m(3nk - 1)$.

Thus, in order to calculate one discrete gradient of the function f_k at the point $\boldsymbol{x}$ the following simplified scheme can be used. We compute the values of the function f_k at the points $\boldsymbol{x}$ and $\boldsymbol{x}^0$. Then we store values of all term functions calculated at $\boldsymbol{x}^0$. In order to calculate the value of f_k at $\boldsymbol{x}^1$ we update only those term functions which contain the first coordinate and keep all other term functions as they are. We repeat this scheme for all other coordinates. Note that we compute the function f_k at the point $\boldsymbol{x}$ when we compute the first discrete gradient at this point. The use of this scheme allows us to reduce the number of term functions evaluations for computation of the first discrete gradient

$$\frac{N_t}{\bar{N}_t} = \frac{mnk(nk + 1)}{m(3nk - 1)} \approx \frac{nk + 1}{3}$$

times and approximately $(nk + 1)/2$ times for the computation of all other discrete gradients at $\boldsymbol{x}$. This reduction becomes very significant as the number of clusters k increases.

The similar scheme can be designed to compute discrete gradients of the auxiliary cluster function $\bar{f}_k$. Here, the total number of term functions is mn. The function $\bar{f}_k$ has n variables and therefore, one needs $n+1$ evaluations of this function to compute its one discrete gradient. This means that the total number of evaluations of term functions to compute one discrete gradient of $\bar{f}_k$ is $N_t = mn(n + 1)$.

For a given $i \in \{1, \ldots, n\}$, we compute values of the function $\bar{f}_k$ at the following $n + 1$ points:

$$\boldsymbol{x}, \boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^{i-1}, \boldsymbol{x}^{i+1}, \ldots, \boldsymbol{x}^n.$$

The full evaluation of the function $\bar{f}_k$ at points $\boldsymbol{x}$ and $\boldsymbol{x}^0$ requires $2mn$ calculations of the term functions. Other points from this sequence are obtained from the previous point by changing only one coordinate. This means that we need to update only m term functions at points $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{i-1}, \boldsymbol{x}^{i+1}, \ldots, \boldsymbol{x}^n$ and therefore, the total number of evaluations of the term functions for calculating of $\bar{f}_k$ at these points is $m(n-1)$. The total number of evaluations of term functions for computation of one discrete gradient is $\bar{N}_t = m(3n-1)$.

Therefore, we can apply the following simplified scheme to compute one discrete gradient of the $\bar{f}_k$ at the point $\boldsymbol{x}$. We compute the function $\bar{f}_k$ at the points $\boldsymbol{x}$ and $\boldsymbol{x}^0$ and store the values of all term functions calculated at $\boldsymbol{x}^0$. In order to calculate the value of $\bar{f}_k$ at $\boldsymbol{x}^1$ for each data point we update only the first term function and keep all other term functions as they are. This scheme is repeated for all other coordinates. Applying this scheme leads to the reduction of the number of term functions evaluations to compute the first discrete gradient

$$\frac{N_t}{\bar{N}_t} = \frac{mn(n+1)}{m(3n-1)} \approx \frac{n+1}{3}$$

times and approximately $(n+1)/2$ times for the computation of all other discrete gradients at $\boldsymbol{x}$.

8.6 Smooth Incremental Clustering Algorithm

In this section, we describe the *smooth incremental clustering algorithm* (IS-CLUST) where the objective functions in both the clustering and the auxiliary clustering problems are approximated by smooth functions [33]. To approximate objective functions, we apply the HSM, described in Sect. 3.9. The hyperbolic smoothings of the cluster function f_k and the auxiliary cluster function $\bar{f}_k$ are given in Sects. 4.7.2 and 4.7.3, respectively. For convenience, we recall these smooth functions for any $l = 2, \ldots, k$:

$$\Phi_{l,\tau}(\boldsymbol{x}, \boldsymbol{t}) = -\frac{1}{m} \sum_{i=1}^{m} \left(t_i + \sum_{j=1}^{l} \frac{-d_p(\boldsymbol{x}_j, \boldsymbol{a}_i) - t_i + \sqrt{(d_p(\boldsymbol{x}_j, \boldsymbol{a}_i) + t_i)^2 + \tau^2}}{2} \right)$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(-t_i + \sum_{j=1}^{l} \frac{t_i + d_p(\boldsymbol{x}_j, \boldsymbol{a}_i) - \sqrt{(d_p(\boldsymbol{x}_j, \boldsymbol{a}_i) + t_i)^2 + \tau^2}}{2} \right),$$

and

$$\begin{aligned}\bar{\Phi}_{l,\tau}(\boldsymbol{y}) &= \frac{1}{m}\sum_{i=1}^{m} r_{l-1}^{i} \\ &\quad - \frac{1}{m}\sum_{i=1}^{m} \frac{r_{l-1}^{i} - d_p(\boldsymbol{y},\boldsymbol{a}_i) + \sqrt{(r_{l-1}^{i} - d_p(\boldsymbol{y},\boldsymbol{a}_i))^2 + \tau^2}}{2} \\ &= \frac{1}{m}\sum_{i=1}^{m} \frac{r_{l-1}^{i} + d_p(\boldsymbol{y},\boldsymbol{a}_i) - \sqrt{(r_{l-1}^{i} - d_p(\boldsymbol{y},\boldsymbol{a}_i))^2 + \tau^2}}{2},\end{aligned}$$

where $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l) \in \mathbb{R}^{nl}$, $\boldsymbol{y} \in \mathbb{R}^n$ and $\boldsymbol{t} = (t_1, \ldots, t_m)$, such that

$$t_i = -\min_{j=1,\ldots,l} d_p(\boldsymbol{x}_j, \boldsymbol{a}_i), \quad i = 1, \ldots, m.$$

As mentioned before, if the function d_p is defined using the squared Euclidean norm, then the functions $\Phi_{l,\tau}$ and $\bar{\Phi}_{l,\tau}$ are both smooth since d_2 is differentiable. However, the other two functions d_1 and d_∞ are nonsmooth, and we need to reapply the hyperbolic smoothing technique to these functions to approximate them with the smooth functions. These results are presented in Sect. 4.7.

Take any sequence $\{\tau_h\}$ such that $\tau_h \downarrow 0$ as $h \to \infty$, then the clustering and the auxiliary clustering problems (7.2) and (7.4) can be replaced by the sequence of the following smooth problems, respectively:

$$\begin{cases} \text{minimize} & \Phi_{l,\tau_h}(\boldsymbol{x}, \boldsymbol{t}) \\ \text{subject to} & \boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l) \in \mathbb{R}^{nl}, \end{cases} \tag{8.2}$$

and

$$\begin{cases} \text{minimize} & \bar{\Phi}_{l,\tau_h}(\boldsymbol{y}) \\ \text{subject to} & \boldsymbol{y} \in \mathbb{R}^n. \end{cases} \tag{8.3}$$

The IS-CLUST solves the clustering problem by combining the MSINC-CLUST and an optimization method. The IS-CLUST applies the MSINC-CLUST to solve the clustering problem globally. Since the clustering and the auxiliary clustering problems (8.2) and (8.3) are smooth problems the IS-CLUST can utilize any smooth optimization method to solve them. The flowchart of the IS-CLUST is presented in Fig. 8.4.

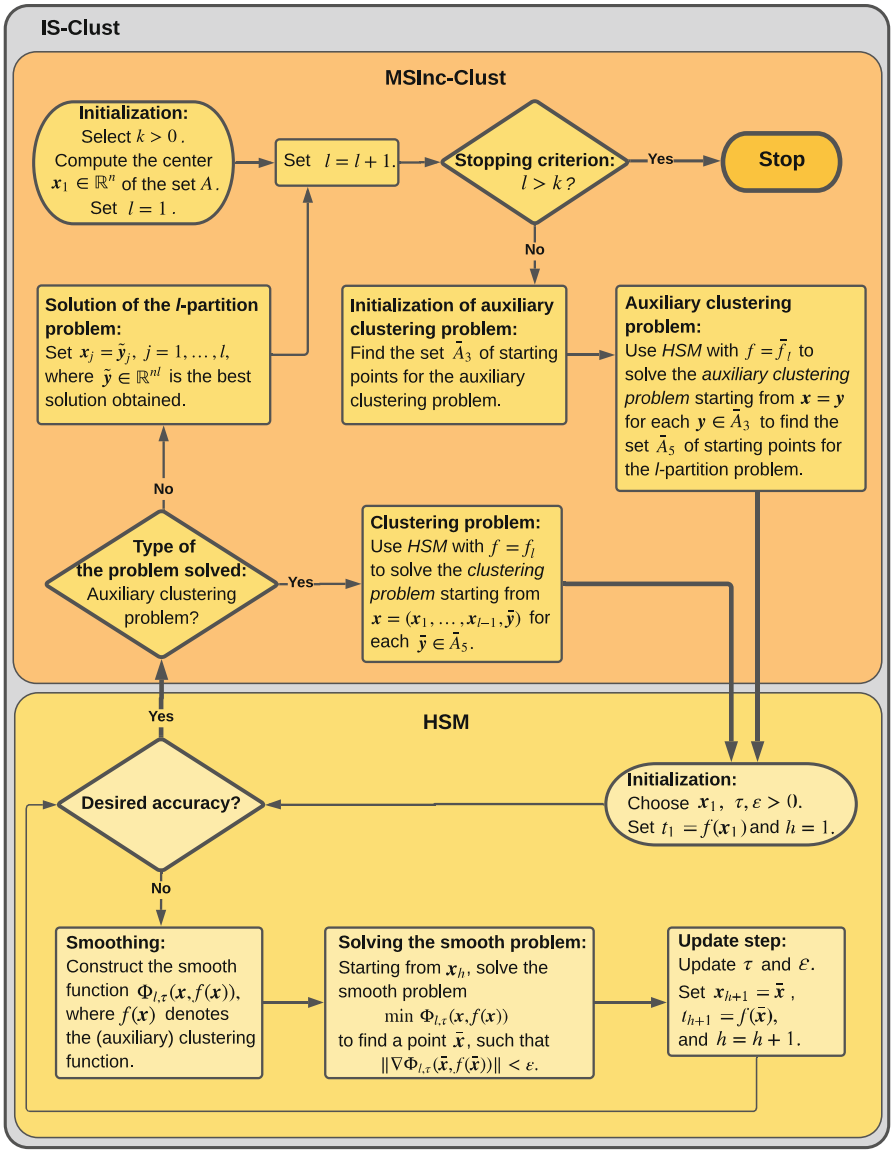

Fig. 8.4 Smooth incremental clustering algorithm (IS-CLUST)

The IS-CLUST is given in its step by step description as follows.

Algorithm 8.9 Smooth incremental clustering algorithm (IS-CLUST)

Input: Data set A and the number of clusters k to be computed.
Output: The l-partition of the set A with $l = 1, \ldots, k$.

1: *(Initialization)* Compute the center $\boldsymbol{x}_1 \in \mathbb{R}^n$ of the set A. Set $l = 1$.
2: *(Stopping criterion)* Set $l = l + 1$. If $l > k$, then **stop**—the k-partition problem has been solved.
3: *(Computation of a set of starting points for the next cluster center)* Apply Algorithm 7.2 and by solving the smooth auxiliary clustering problem (8.3), compute the set $\bar{A}_5$.
4: *(Computation of a set of cluster centers)* For each $\bar{\boldsymbol{y}} \in \bar{A}_5$, take $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{l-1}, \bar{\boldsymbol{y}})$ as a starting point and solve the smooth clustering problem (8.2) and find a solution $(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_l)$. Denote by $\bar{A}_6$ a set of all such solutions.
5: *(Computation of the best solution)* Compute

$$f_l^{\min} = \min \left\{ f_l(\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_l) : \ (\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_l) \in \bar{A}_6 \right\},$$

and the collection of cluster centers $(\tilde{\boldsymbol{y}}_1, \ldots, \tilde{\boldsymbol{y}}_l)$ such that

$$f_l(\tilde{\boldsymbol{y}}_1, \ldots, \tilde{\boldsymbol{y}}_l) = f_l^{\min}.$$

6: *(Solution to the lth partition problem)* Set $\boldsymbol{x}_j = \tilde{\boldsymbol{y}}_j, \ j = 1, \ldots, l$ as a solution to the lth partition problem and go to Step 2.

Note that in Step 3 of this algorithm when we apply Algorithm 7.2 to compute the set of starting points $\bar{A}_5$, the auxiliary cluster function $\bar{f}_l$ is approximated by the smooth function $\bar{\Phi}_{l,\tau}$.