Advances in Data Analysis and Classification
https://doi.org/10.1007/s11634-025-00661-6

REGULAR ARTICLE

An efficient incremental algorithm for clustering large datasets

Jenni Lampainen[1] · **Kaisa Joki**[1] · **Napsu Karmitsa**[2] · **Marko M. Mäkelä**[1]

Received: 13 June 2025 / Revised: 14 November 2025 / Accepted: 25 November 2025

Abstract
Clustering is a fundamental task in data mining and machine learning, particularly for analyzing large-scale data. In this paper, we introduce Clust-Splitter, an efficient algorithm based on novel incremental approach and nonsmooth formulation of the the minimum sum-of-squares clustering problem. Particularly, the clustering task is approached through a sequence of three nonsmooth optimization problems: two auxiliary problems used to generate suitable starting points, followed by a main clustering formulation. To solve these problems effectively in very large datasets, the limited memory bundle method (Haarala et al. in Optim Methods Softw 19(6):673–692, 2004) is applied as an underlying solver in Clust-Splitter. We test and evaluate Clust-Splitter on real-world datasets characterized by both a large number of attributes and a large number of data points and compare its performance with several state-of-the-art large-scale clustering algorithms. Experimental results demonstrate the efficiency of the proposed method for clustering very large datasets, as well as the high quality of its solutions, which are on par with those of the best existing methods.

Keywords Clustering · Incremental algorithm · Large-scale data · Limited memory bundle method · Nonsmooth optimization · Nonconvex optimization

Mathematics Subject Classification 90C90 · 90C26

✉ Jenni Lampainen
jmlamp@utu.fi

Kaisa Joki
kaisa.joki@utu.fi

Napsu Karmitsa
napsu@karmitsa.fi

Marko M. Mäkelä
makela@utu.fi

[1] Department of Mathematics and Statistics, University of Turku, 20014 Turku, Finland

[2] Department of Computing, University of Turku, 20014 Turku, Finland

Published online: 13 March 2026

1 Introduction

Clustering is a fundamental task in data mining and machine learning, aiming to group data points into clusters based on their similarity. It plays a vital role in numerous modern applications, including bioinformatics (Karim et al. 2021; Sanjak et al. 2024), cybersecurity (Riddle-Workman et al. 2021; Taheri et al. 2020), and image processing (Kim et al. 2020). Recent growth of data, along with advancements in computer hardware, now enables the storage and processing of massive datasets, including millions of data points and attributes. This capability makes large-scale clustering both possible and essential, but it also introduces major challenges: many existing algorithms either produce suboptimal outcomes, such as local minima, or require excessive computational resources. Therefore, there is a significant need for clustering methods that can generate accurate results within a reasonable time on very large datasets.

A commonly used clustering formulation is the *minimum sum-of-squares clustering (MSSC) problem*, which aims to partition data points into clusters by minimizing the sum of squared Euclidean distances of the points to the nearest cluster center. In this formulation, each data point is assigned to exactly one cluster, which corresponds to the *hard clustering problem*. The MSSC problem can be expressed as a global optimization problem, and various optimization techniques − such as nonsmooth optimization methods (Bagirov et al. 2023, 2025; Bagirov and Mohebi 2015; Bagirov and Ugon 2005; Bagirov and Yearwood 2006; Karmitsa et al. 2017, 2018), difference-of-convex (DC) methods (Bagirov et al. 2025, 2016; Karmitsa et al. 2017; Khalaf et al. 2017; Le Thi et al. 2007, 2014; Le Thi and Pham 2009), and metaheuristics (Seifollahi et al. 2019; Selim and Al-Sultan 1991; Al-Sultan 1995; Alotaibi 2022; Xu et al. 2014; Abdo et al. 2024; Cura 2012; Gribel and Vidal 2019; Mansueto and Schoen 2021; Rahman and Islam 2014) − have been applied to develop clustering algorithms for MSSC.

Among optimization-based approaches, nonsmooth optimization (NSO) (Bagirov et al. 2014) provides a flexible framework for clustering problems, as it can handle objective functions that are not continuously differentiable. More importantly, NSO-based clustering models are particularly effective in large-scale settings because, unlike traditional formulations where the number of variables grows with the number of data points, they use a fixed number of variables determined only by the number of clusters and features (Bagirov et al. 2025). This reduction in dimensionality leads to substantially lower computational complexity.

While global optimization methods for MSSC can provide high-quality solutions, they are often computationally too intensive for large datasets. Local optimization methods, on the other hand, are computationally efficient but highly sensitive to the choice of starting points, which may result in convergence to poor local minima far from the global minimum. Incremental algorithms offer a practical compromise, gradually building the solution by increasing the number of clusters one by one. This approach reduces computational complexity and improves the quality of solutions by systematically generating good initializations. A common way to implement this idea is to rely on the starting point selection procedure introduced in Ordin and Bagirov (2015). Variants of this approach have since been applied in several studies, includ-

ing Bagirov et al. (2023), Bagirov and Mohebi (2015), Bagirov et al. (2015, 2016), Karmitsa et al. (2017, 2018), Khalaf et al. (2017), Xavier and Xavier (2020).

In this paper, we introduce a novel incremental clustering method, Clust-Splitter, designed for large-scale MSSC problems. The proposed algorithm is based on the NSO approach and incorporates a new data splitting strategy to generate starting points: at each iteration, the number of clusters is increased by one, and then the cluster with the highest dissimilarity from the previous solution is split into two to generate effective starting points for the clustering task with one additional cluster. This novel strategy of splitting clusters based on their similarity clearly distinguishes Clust-Splitter from the other incremental clustering methods mentioned above. In particular, the proposed approach is computationally more efficient and the resulting optimization problems are significantly easier to solve. For the underlying NSO problems, we employ the limited memory bundle method (LMBM) (Haarala 2004; Haarala et al. 2004, 2007), which is considered an efficient method for solving large-scale NSO problems. Moreover, the LMBM has been successfully applied in various machine learning application [see, e.g., Halkola et al. (2023), Karmitsa et al. (2018, 2022, 2023), Paasivirta et al. (2024)].

The main contributions of this paper are as follows:

1. Development of a novel incremental model for solving MSSC problems;
2. Design of the new algorithm Clust-Splitter, which implements the proposed model and achieves high accuracy and efficiency in clustering datasets with hundreds of thousands of data points and/or hundreds of features;
3. Comprehensive numerical evaluation comparing Clust-Splitter with several state-of-the-art methods on very large datasets;
4. Open-source implementation of Clust-Splitter, available at https://github.com/jmlamp/Clust-Splitter.

The remainder of this paper is organized as follows. Section 2 introduces the notations and fundamental concepts from cluster analysis. A brief literature review of incremental clustering algorithms is provided in Section 3, which also presents the main idea of Clust-Splitter, emphasizing its differences from existing methods. In Section 4, the nonsmooth clustering problems required for Clust-Splitter are formulated. A detailed description of the proposed incremental clustering algorithm is presented in Section 5. The results of extensive numerical experiments are reported and analyzed in Section 6. Finally, Section 7 concludes the paper, while the Appendix provides the precise definitions of the validity indices as well as detailed numerical results.

2 Definitions in cluster analysis

This section presents the key definitions and concepts used in the paper. Throughout, bold symbols represent vectors. In cluster analysis, we consider a finite set A of points in the n-dimensional space $\mathbb{R}^n$. In other words,

$$A = \{\boldsymbol{a}^1, \ldots, \boldsymbol{a}^m\}, \quad \text{where } \boldsymbol{a}^i \in \mathbb{R}^n,\ i = 1, \ldots, m.$$

Each data point $\boldsymbol{a}^i$ has *n features*.

The *hard unconstrained clustering problem* entails partitioning the points in the set A into k disjoint subsets A^j, $j = 1, \ldots, k$, based on specific predefined criteria such that

1. $A^j \neq \emptyset$ for all $j = 1, \ldots, k$;
2. $A^j \bigcap A^l = \emptyset$ for all $j, l = 1, \ldots, k,\ j \neq l$; and
3. $A = \bigcup_{j=1}^{k} A^j$.

The subsets A^j, $j = 1, \ldots, k$, are referred to as *clusters*. Each cluster A^j can be characterized by its *center* $\boldsymbol{x}^j \in \mathbb{R}^n$, $j = 1, \ldots, k$. The task of determining these centers is known as the *k-clustering problem* (Bagirov et al. 2025). In what follows, we sometimes refer to the *k*-clustering problem as the main clustering problem.

The concept of a *similarity measure* is essential for formulating the clustering problem. Typically, a similarity measure is defined in terms of a distance metric. In this work, we define it using the squared Euclidean norm (the L_2-norm) as the distance

$$d_2(\boldsymbol{x}, \boldsymbol{a}) = \|\boldsymbol{x} - \boldsymbol{a}\|^2 = \sum_{i=1}^{n} (x_i - a_i)^2,$$

where $\boldsymbol{x}, \boldsymbol{a} \in \mathbb{R}^n$.

Each data point is assigned to the cluster with the nearest center. Furthermore, the *cluster function for the cluster* A^j, $j = 1, \ldots, k$, is defined as

$$f_{A^j}(\boldsymbol{x}) = \sum_{\boldsymbol{a}^i \in A^j} d_2(\boldsymbol{x}, \boldsymbol{a}^i), \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is treated as the center of the cluster A^j.

3 Related work

This section provides a brief literature review of incremental clustering algorithms, with the aim of highlighting the novelty of Clust-Splitter. As stated before, Clust-Splitter is an incremental clustering algorithm, meaning that the solution is built gradually by increasing the number of clusters one by one. In other words, to solve a clustering problem with k clusters, all intermediate problems with $1, 2, \ldots, k-1$ clusters are solved first sequentially. Consequently, an incremental algorithm generates not only the final solution but also all intermediate solutions, providing a complete sequence of clustering results. Many such incremental algorithms have been proposed for MSSC problems, the most notable being the global *k*-means algorithm

(Likas et al. 2003), and NSO-based algorithms using the starting point generation procedure introduced in Ordin and Bagirov (2015).

Global *k*-means is an incremental variant of the well-known *k*-means algorithm (McQueen 1967). The algorithm starts with a single cluster and adds one new cluster at a time. At each iteration, several candidate positions are considered for the new cluster center. Furthermore, each candidate center is combined with the existing clusters to form multiple starting points for *k*-means. Running *k*-means from these starting points produces multiple solutions, from which the one yielding the lowest clustering error is selected. This process is repeated until the desired number of clusters is reached. Many variants of global *k*-means have been proposed, see for example Bagirov et al. (2011), Bagirov (2008), Lai and Huang (2010), Xie et al. (2011). However, *k*-means-based incremental algorithms are not well suited for large-scale clustering.

NSO-based incremental clustering algorithms also address clustering problems gradually. The main differences between such methods lie in the choice of NSO solver for clustering problems, as well as in the strategy to select starting points. A well-established approach is the starting point generation procedure introduced in Ordin and Bagirov (2015), where the solution to the $(k-1)$-clustering problem (i.e., the main clustering problem with $(k-1)$ clusters, to be formally defined in Subsection 4.3) is used to produce starting points for the *k*-clustering problem. Thus, iteration *k* starts with considering the clusters A^j, $j = 1, \ldots, k-1$, based on the solution of the previous $(k-1)$-clustering problem and determining a suitable starting position for one additional cluster among them. For this purpose, an auxiliary clustering problem is employed to generate multiple candidate centers for the new cluster [see Algorithm 1 in Ordin and Bagirov (2015)]. Each starting point for the *k*-clustering problem is then constructed by combining the solution of the $(k-1)$-clustering problem with one of these candidate centers. The main *k*-clustering problem is solved from each of these starting points, and the best solution is carried forward to the next iteration. When $k = 1$, there is no previous solution, but the corresponding main clustering problem is convex and can therefore be solved straightforwardly, without sophisticated selection of starting points. Incremental clustering methods based on the described approach include, for example, Bagirov (2014), Bagirov et al. (2015, 2016), Cuong et al. (2019), Karmitsa et al. (2017, 2018, 2025).

As stated before, Clust-Splitter is also an incremental algorithm. Its approach resembles NSO-based incremental clustering methods, particularly those that employ the starting point generation procedure introduced in Ordin and Bagirov (2015), but with a key distinction. At iteration *k*, we also consider the clusters A^j, $j = 1, \ldots, k-1$, obtained during the iteration, but now the cluster with the highest cluster function (1) value is split into two by solving two auxiliary problems, while the other clusters are kept unchanged. The outcome of this procedure is a starting point for the *k*-clustering problem. Unlike Ordin and Bagirov (2015), which incorporates all data points into its auxiliary problem, our method restricts consideration in both auxiliary problems to the points within the cluster being split. Due to this, our auxiliary problems are substantially smaller and therefore computationally easier to solve. Moreover, the use of two auxiliary problems produces higher-quality starting points, making it sufficient to solve the *k*-clustering problem once, rather than from

multiple starting points as in Ordin and Bagirov (2015). The primary theoretical contribution of Clust-Splitter therefore lies in the construction of auxiliary problems, which balance efficiency with solution quality.

To the best of our knowledge, only a few incremental algorithms exist for large-scale clustering and they employ NSO formulations. The most notable examples are LMBM-Clust (Karmitsa et al. 2018) and its stochastic variant Big-Clust (Karmitsa et al. 2025), both of which rely on the starting point procedure described by Ordin and Bagirov (2015) or its slight variation. Accordingly, in the next section, our comparison of Clust-Splitter focuses primarily on Ordin and Bagirov (2015), as both methods, LMBM-Clust and Big-Clust, follow its principle and use the same auxiliary problem. This makes the comparison particularly meaningful.

4 Formulations of clustering problems

In this section, we present the NSO formulations of the clustering problems utilized by Clust-Splitter: the starting point auxiliary problem, the 2-clustering auxiliary problem, and the main k-clustering problem. For each problem, we highlight the differences from the formulations in Ordin and Bagirov (2015), thereby clarifying the contribution of our approach.

4.1 Starting point auxiliary problem

With the help of the *starting point auxiliary* (SPA) problem, the aim is to identify two appropriate starting cluster centers before the selected cluster can be split into two. While one of the centers is the center of the splitted cluster, the other is determined through the solution of the SPA problem. This ensures that the splitting process of the selected cluster begins with a well-chosen starting point being crucial for the success of the clustering algorithm.

Assume that c is the index of the cluster being split and the center of the cluster A^c is $\tilde{\boldsymbol{x}}^c \in \mathbb{R}^n$. The *SPA function* is defined as

$$\tilde{f}(\boldsymbol{z}) = \sum_{\boldsymbol{a}^i \in A^c} \min \left\{ d_2(\tilde{\boldsymbol{x}}^c, \boldsymbol{a}^i), d_2(\boldsymbol{z}, \boldsymbol{a}^i) \right\}, \tag{2}$$

where $\boldsymbol{z} \in \mathbb{R}^n$. Note that $\tilde{f}(\boldsymbol{z}) \leq \sum_{\boldsymbol{a}^i \in A^c} d_2(\tilde{\boldsymbol{x}}^c, \boldsymbol{a}^i) = f_{A^c}(\tilde{\boldsymbol{x}}^c)$ for all $\boldsymbol{z} \in \mathbb{R}^n$, and the inequality is strict if the distance from any data point to $\boldsymbol{z}$ is smaller than to $\tilde{\boldsymbol{x}}^c$. The function $\tilde{f}$ is nonsmooth, locally Lipschitz continuous (LLC) and typically nonconvex as a sum of minima of convex functions. The *SPA problem* is then formulated as

$$\begin{cases} \text{minimize} & \tilde{f}(\boldsymbol{z}) \\ \text{subject to} & \boldsymbol{z} \in \mathbb{R}^n. \end{cases} \tag{3}$$

In a traditional incremental method, individual clusters are not split. Instead, a new cluster is inserted among all existing clusters, and the auxiliary clustering problem

considers the entire dataset to find a suitable initial position for this additional cluster. In contrast, the SPA problem [conceptually similar to the auxiliary clustering problem in Ordin and Bagirov (2015)] in Clust-Splitter focuses only on the data points of the cluster being split. Starting points for the SPA problem are generated intuitively, for example, using the current center of the cluster or the center of randomly selected data points within the cluster (see Remark 3). For the auxiliary clustering problem in Ordin and Bagirov (2015), starting points are computed as centers of subsets of data points far from existing cluster centers. Both auxiliary problems are solved once for each starting point, but Clust-Splitter retains only the solution minimizing the SPA function (2), whereas Ordin and Bagirov (2015) stores all solutions satisfying specific criteria. Overall, the SPA problem in Clust-Splitter is simpler (fewer terms) and selects starting points more intuitively than the auxiliary problem in Ordin and Bagirov (2015).

4.2 2-clustering auxiliary problem

The 2-clustering auxiliary problem focuses on splitting the selected cluster into two separate clusters. After solving the 2-clustering auxiliary problem, we have finished the splitting of the considered cluster and produced the centers of the two newly formed clusters. Assume that c is the index of the cluster being split. The *2-clustering auxiliary function* is defined as

$$\hat{f}(\boldsymbol{y}) = \sum_{\boldsymbol{a}^i \in A^c} \min\left\{d_2(\boldsymbol{y}^1, \boldsymbol{a}^i), d_2(\boldsymbol{y}^2, \boldsymbol{a}^i)\right\},$$

where $\boldsymbol{y} = (\boldsymbol{y}^1, \boldsymbol{y}^2) \in \mathbb{R}^{2n}$. The function $\hat{f}$ is nonsmooth, LLC and typically nonconvex as a sum of minima of convex functions. The *2-clustering auxiliary problem* is then formulated as

$$\begin{cases} \text{minimize} & \hat{f}(\boldsymbol{y}) \\ \text{subject to} & \boldsymbol{y} = (\boldsymbol{y}^1, \boldsymbol{y}^2) \in \mathbb{R}^{2n}. \end{cases} \tag{4}$$

The 2-clustering auxiliary problem introduces an additional step compared to the starting point generation procedure in Ordin and Bagirov (2015). In Clust-Splitter, the 2-clustering auxiliary problem performs the splitting of the considered cluster into two separate clusters, and the resulting centers are then combined with the solution of the $(k-1)$-clustering problem to form one high-quality starting point for the k-clustering problem. In Ordin and Bagirov (2015), the auxiliary clustering problem (similar to the SPA problem in Clust-Splitter) is solved multiple times, and from these solutions, those satisfying specific criteria are combined with the solution of the $(k-1)$-clustering problem to generate several starting points for the k-clustering problem. Although the 2-clustering auxiliary problem represents an additional step in Clust-Splitter, it is solved only once and involves only the points within the cluster being split, making it computationally efficient. Overall, the 2-clustering auxiliary problem improves the quality of the starting point for the k-clustering problem without significantly increasing computation time.

4.3 *k*-clustering problem

The *k*-clustering problem is the main clustering task. As a result of solving the *k*-clustering problem, the process yields the centers of *k* clusters. The *k-clustering function*, or *kth cluster function*, is defined as

$$f_k(\boldsymbol{x}) = \sum_{i=1}^{m} \min_{j=1,\ldots,k} d_2(\boldsymbol{x}^j, \boldsymbol{a}^i), \tag{5}$$

where $\boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) \in \mathbb{R}^{nk}$. The function f_k is LLC for any *k*, convex for $k = 1$, and nearly always nonconvex and nonsmooth for $k > 1$. The *k-clustering problem* is formulated as

$$\begin{cases} \text{minimize} & f_k(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} = (\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k) \in \mathbb{R}^{nk}, \end{cases} \tag{6}$$

which corresponds to the MSSC problem.

As a result of the stronger initialization provided by the two auxiliary problems, Clust-Splitter needs to solve the *k*-clustering problem only once per iteration. In contrast, clustering algorithms based on the starting point generation procedure in Ordin and Bagirov (2015) solve the *k*-clustering problem multiple times, once for each good starting point generated with the auxiliary clustering problem. While the *k*-clustering formulations are identical in Clust-Splitter and in the methods based on the starting point generation procedure of Ordin and Bagirov (2015), solving the *k*-clustering problem only once makes Clust-Splitter computationally more efficient. In summary, the novelty of Clust-Splitter lies in its use of two auxiliary problems based on cluster splitting, which generate sufficiently strong starting points to avoid repeatedly solving the *k*-clustering problem and thus reduce excessive computational effort.

5 Clust-Splitter method

In this section, we introduce the new Clust-Splitter method, an incremental algorithm designed to solve MSSC problems. Suppose that we have already solved the $(k-1)$-clustering problem and are now moving on to solve the *k*-clustering problem. First, auxiliary problems are used to split a selected cluster from the previous solution into two smaller clusters. The resulting configuration, together with the previous solution of the $(k-1)$-clustering problem, provides a foundation (i.e., a starting point) for solving the *k*-clustering problem on the entire dataset. This process is repeated until the user-specified maximum number of clusters, k_{max}, is achieved. Therefore, Clust-Splitter solves not only the k_{max}-clustering problem but also all intermediate *k*-clustering problems for $k = 1, \ldots, k_{max} - 1$. Clust-Splitter uses the LMBM (Haarala 2004; Haarala et al. 2004, 2007) to solve all optimization problems.

A key aspect of CLUST-SPLITTER is the strategic use of cluster splitting, which facilitates the identification of high-quality starting points, an essential factor in achieving global or near-global solutions in nonconvex clustering problems. As already seen, this is also a notable advantage of the new approach compared to the traditional starting point generation procedure in incremental clustering algorithms [see, e.g., Bagirov et al. (2016), Karmitsa et al. (2018, 2025)]. Namely, the auxiliary problems in CLUST-SPLITTER are smaller and therefore significantly easier to solve. Furthermore, in the method, additional criteria for selecting the cluster to be split can be adjusted using parameters. For instance, splitting can be prohibited for clusters containing fewer than five data points (i.e., outlier clusters). Figure 1 illustrates the structure of the CLUST-SPLITTER method, and the exact algorithm is presented in Algorithm 1. Next, we take a closer look at the three main parts of the CLUST-SPLITTER method. In each part, one of the optimization problems (3), (4) or (6) is solved.

Suppose that we have just started the iteration k. First, we define the clusters A^j, $j = 1, \ldots, k-1$, based on the solution $\hat{\boldsymbol{x}} = (\hat{\boldsymbol{x}}^1, \ldots, \hat{\boldsymbol{x}}^{k-1})$ to the previous $(k-1)$-clustering problem. Second, we store this previous solution to $\tilde{\boldsymbol{x}} = (\tilde{\boldsymbol{x}}^1, \ldots, \tilde{\boldsymbol{x}}^{k-1})$. Third, we calculate the cluster function value $f_{A^j}(\tilde{\boldsymbol{x}}^j)$ using (1) for each cluster A^j, $j = 1, \ldots, k-1$, and select the cluster with the highest value for splitting. We denote the index of this cluster by c. Thus, in the splitting part of the method, the considered cluster is A^c whereas its center is $\tilde{\boldsymbol{x}}^c$. At this point, we are ready to proceed with the auxiliary problems.

The SPA problem (3) in Step 3 of Algorithm 1 aims to identify the best possible starting point for the 2-clustering auxiliary problem (4) in order to split the cluster A^c. The SPA problem is computed p times, where the number p is specified by the user. Each of these runs use a different starting point, which can be chosen, for example, as a center of randomly selected data points within the splitted cluster A^c.

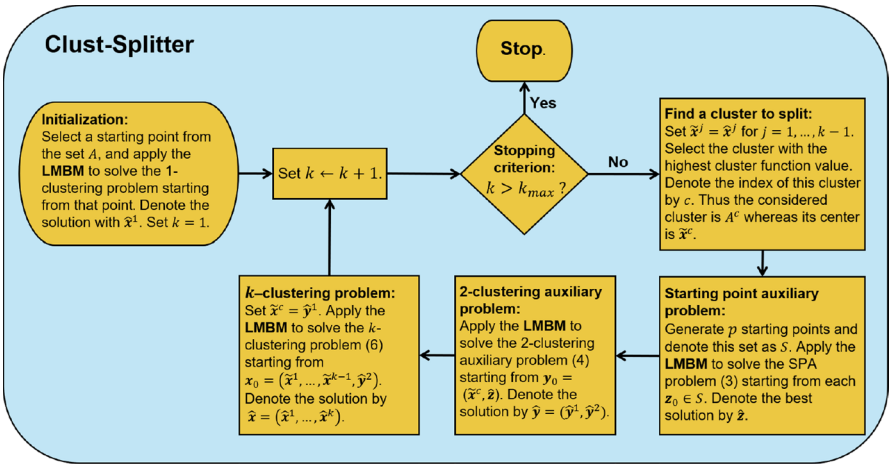


Fig. 1 The CLUST-SPLITTER method. Note that if $k = 2$, after solving the 2-clustering auxiliary problem, we set $\hat{\boldsymbol{x}} = (\hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2)$ and skip the step with the k-clustering problem.

For a more detailed description, see Remark 3. The solution that yields the smallest SPA function (2) value is denoted by $\hat{\boldsymbol{z}}$.

The goal of the 2-clustering auxiliary problem (4) in Step 4 of Algorithm 1 is to split the cluster A^c into two parts, resulting in the centers $\hat{\boldsymbol{y}}^1$ and $\hat{\boldsymbol{y}}^2$ of the newly formed clusters. The starting point for the problem (4) consists of the center $\tilde{\boldsymbol{x}}^c$ of the considered cluster A^c and the additional center $\hat{\boldsymbol{z}}$ determined by the SPA problem (3). In other words, the starting point is $\boldsymbol{y}_0 = (\tilde{\boldsymbol{x}}^c, \hat{\boldsymbol{z}})$. The solution of the 2-clustering auxiliary problem consists of the centers of the two new clusters, stored in the vector $\hat{\boldsymbol{y}} = (\hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2)$.

After solving the two auxiliary problems, we are finally ready to address the main problem: the k-clustering problem (6) in Step 5 of Algorithm 1. The input to the k-clustering problem is the solution $\hat{\boldsymbol{y}} = (\hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2)$ of the 2-clustering auxiliary problem splitting the cluster A^c. Using this input we construct the starting point for the k-clustering problem as

$$\boldsymbol{x}_0 = (\boldsymbol{x}_0^1, \ldots, \boldsymbol{x}_0^k) = (\tilde{\boldsymbol{x}}^1, \ldots, \tilde{\boldsymbol{x}}^{k-1}, \hat{\boldsymbol{y}}^2),$$

where $\tilde{\boldsymbol{x}}^1, \ldots, \tilde{\boldsymbol{x}}^{k-1}$ are otherwise the cluster centers obtained from the $(k-1)$-clustering problem (6) but $\tilde{\boldsymbol{x}}^c = \hat{\boldsymbol{y}}^1$. The solution to the k-clustering problem consists of the updated cluster centers and is denoted by $\hat{\boldsymbol{x}} = (\hat{\boldsymbol{x}}^1, \ldots, \hat{\boldsymbol{x}}^k)$. For example, if $k = 4$ then we know three cluster centers $\tilde{\boldsymbol{x}}^1$, $\tilde{\boldsymbol{x}}^2$ and $\tilde{\boldsymbol{x}}^3$ from the previous iteration of the method. Furthermore, if the second cluster is being split, then after obtaining $\hat{\boldsymbol{y}}^1$ and $\hat{\boldsymbol{y}}^2$, the starting point for the 4-clustering problem is set as $\boldsymbol{x}_0 = (\tilde{\boldsymbol{x}}^1, \hat{\boldsymbol{y}}^1, \tilde{\boldsymbol{x}}^3, \hat{\boldsymbol{y}}^2)$.

Algorithm 1 ***Clust-Splitter***

Input: The dataset A, the maximum number of clusters $k_{max} \geq 1$, and the number of starting points $p \geq 1$ for the auxiliary problem (3).

Output: The solution $\hat{\boldsymbol{x}}$ to the k_{max}-clustering problem and all the intermediate solutions to the k-clustering problems with $k = 1, \ldots, k_{max} - 1$.

Step 0. *Initialization:* Select a starting point from the set A, and apply the LMBM to solve the 1-clustering problem (6) starting from that point. Denote the solution with $\hat{\boldsymbol{x}}^1$. Set $k = 1$.

Step 1. *Stopping criterion:* Set $k \leftarrow k + 1$. If $k > k_{max}$, STOP: the k_{max}-clustering problem has been solved.

Step 2. *Find a cluster to split:* Define the clusters A^j, $j = 1, \ldots, k-1$, based on the solution to the previous $(k-1)$-clustering problem. Set $\tilde{\boldsymbol{x}}^j = \hat{\boldsymbol{x}}^j$ for $j = 1, \ldots, k-1$. Calculate the cluster function value $f_{A^j}(\tilde{\boldsymbol{x}}^j)$ using (1) for each cluster A^j, $j = 1, \ldots, k-1$, and select the cluster with the highest value. Denote the index of this cluster by c. Thus, the considered cluster is A^c whereas its center is $\tilde{\boldsymbol{x}}^c$.

Step 3. *Starting point auxiliary problem:* Generate p starting points from $\mathbb{R}^n$ and denote this set as S. Apply the LMBM to solve the SPA problem (3) starting from each $\boldsymbol{z}_0 \in S$. Choose the solution that gives the smallest value of the auxiliary function (2) and denote it by $\hat{\boldsymbol{z}}$.

Step 4. *2-clustering auxiliary problem:* Apply the LMBM to solve the 2-clustering auxiliary problem (4) starting from $\boldsymbol{y}_0 = (\tilde{\boldsymbol{x}}^c, \hat{\boldsymbol{z}})$. Denote the solution by $\hat{\boldsymbol{y}} = (\hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2)$. If $k = 2$, set $\hat{\boldsymbol{x}} = (\hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2)$ and go to Step 1.

Step 5. *k-clustering problem:* Set $\tilde{\boldsymbol{x}}^c = \hat{\boldsymbol{y}}^1$. Apply the LMBM to solve the k-clustering problem (6) starting from $\boldsymbol{x}_0 = (\tilde{\boldsymbol{x}}^1, \ldots, \tilde{\boldsymbol{x}}^{k-1}, \hat{\boldsymbol{y}}^2)$. Denote the solution by $\hat{\boldsymbol{x}} = (\hat{\boldsymbol{x}}^1, \ldots, \hat{\boldsymbol{x}}^k)$. Go to Step 1.

Remark 1 In Algorithm 1, the optimal number of clusters can also be determined when it is initially unknown. Specifically, the optimal k is reached when the Davies-Bouldin index (DBI) attains its minimum value and the Dunn index (DI) attains its maximum value (see the definitions of the validity indices in Appendix A). Thus, the computation can be stopped once the DBI begins to increase significantly while the DI decreases, which provides an effective criterion for estimating the optimal number k of clusters.

Remark 2 At Step 0 of Algorithm 1, the starting point can be selected as the center of $M_1 > 1$ randomly selected data points from A. Note that here only a single starting point is sufficient, as the 1-clustering problem is convex.

Remark 3 In Step 3 of Algorithm 1, the starting points for the SPA problem (3) can be generated, for example, as follows:

- One or more starting points are generated by computing the center of $M_1 > 1$ randomly selected data points within the cluster A^c being split.
- One or more starting points are computed as the center of $M_2 > 1$ randomly selected data points within the cluster A^c being split, with the additional requirement that each candidate center must be sufficiently distant from the current center of the cluster A^c. If this condition is not met, a new center is generated using another set of M_2 random data points.
- One starting point is selected as the current center of the cluster A^c being split.

As we have seen, the LMBM is used to solve all the optimization problems encountered in Clust-Splitter, but other methods suitable for NSO can also be employed. The LMBM is developed to handle general large-scale nonconvex and nonsmooth optimization problems, and it is one of the few approaches capable of solving such large-scale problems efficiently. Further details on the LMBM are provided in Haarala et al. (2004), and its convergence has been established in Haarala et al. (2007) under some mild assumptions. Since the SPA problem, the 2-clustering auxiliary problem and the k-clustering problem satisfy these assumptions directly, also Clust-Splitter is guaranteed to converge.

6 Numerical experiments

To demonstrate the performance of the proposed Clust-Splitter method, we compare it with five other clustering algorithms: LMBM-Clust (Karmitsa et al. 2018), DC-Clust (Bagirov et al. 2016), Big-Clust (Karmitsa et al. 2025), Big-Means (Mussabayev et al. 2023), and MiniBatchKMeans (Pedregosa et al. 2011) (abbreviated as MBKMeans in the rest of the paper). LMBM-Clust, DC-Clust, and Big-Clust are incremental clustering methods, with LMBM-Clust utilizing the LMBM, DC-Clust leveraging the nonsmooth DC representation of the MSSC problem, and Big-Clust employing stochastic version of the LMBM. Big-Means and MBK-Means are variants of the traditional k-means algorithm, with Big-Means tailored for large-scale data and MBKMeans using a mini-batch optimization approach. The tests are conducted on large-scale real-world data, as well as through external validation using small real-world and simulated datasets.

The solvers Clust-Splitter, LMBM-Clust, DC-Clust, and Big-Clust are implemented in Fortran 95, whereas the solvers Big-Means and MBKMeans are implemented in Python. All computational experiments are carried out on an Intel(R) Core(TM) i3-1215U CPU (1.20GHz, 4.40GHz) running under Windows 10. We use gfortran to compile the Fortran codes and Python 3.9.7 for Python codes with NumPy 1.26.4 and Numba 0.60.0 for Big-Means. Additionally, we follow the provided implementations and adhere to the recommended default parameter values for all methods, as specified in their respective references. An open-source implementation of the proposed Clust-Splitter algorithm is given at https://github.com/jmlamp/Clust-Splitter. Additionally, in Clust-Splitter, the number of starting points is set to $p = 3$ and we generate one starting point using each of the ways described in Remark 3, where the numbers of random data points are set to $M_1 = 10$ and $M_2 = 7$.

6.1 Experiments in large-scale real-world data

We conducted our experiments on a collection of 18 publicly available large-scale real-world datasets, which are the same as in Karmitsa et al. (2025). Summary information for the datasets used is presented in Table 1, while more detailed descriptions are accessible via the references provided. All datasets consist exclusively of numeric features, with no missing values. They also vary in size, with the number of features (n) ranging from 2 to 5000 and the number of data points (m) from 7797 to 581,012.

Clust-Splitter, LMBM-Clust, DC-Clust, and Big-Clust utilize an incremental approach to solve clustering problems. Using these methods, we incrementally compute up to 25 clusters in each dataset. In contrast, since Big-Means and MBK-Means do not produce intermediate results, we conduct multiple runs in each dataset with different numbers of clusters (denoted as k) for comparison purposes. Due to the stochastic characteristics, Big-Clust, Big-Means, and MBKMeans are executed ten times, with the results averaged over these runs. In other words, each dataset with k clusters is computed ten times, and the results are averaged. The incremental algorithm Big-Clust computes up to 25 clusters in a single run, and the algorithm is executed 10 times for each dataset. For Big-Means, each dataset with k clusters needs ten separate runs, whereas MBKMeans is able to produce these ten runs in a

Table 1 Brief description of the datasets used.

Dataset	m	n	$m \times n$	References
ISOLET	7797	617	4,810,749	Cole and Fanty (1991)
Gisette	13,500	5000	67,500,000	Guyon et al. (2004)
Gas Sensor Array Drift	13,910	128	1,780,480	Vergara (2012)
EEG Eye State	14,980	14	209,720	Roesler (2013)
D15112	15,112	2	30,224	Bixby and Reinel (1995)
Online News Popularity	39,644	58	2,299,352	Fernandes et al. (2015)
KEGG Metabolic	53,413	20	1,068,260	Naeem and Asghar (2011)
Shuttle Control	58,000	9	522,000	Markelle et al. (2025)
Sensorless Drive Diagnosis	58,509	48	2,808,432	Bator (2013)
MFCCs for Speech Emotion Recognition	85,134	58	4,937,772	De Dominicis (2020)
Pla85900	85,900	2	171,800	Bixby and Reinel (1995)
Music Analysis	106,574	518	55,205,332	Defferrard et al. (2017)
MiniBooNE Particle Identification	130,064	50	6,503,200	Roe (2005)
Protein Homology	145,751	74	10,785,574	KDD Cup 2004 (2004)
Range Queries Aggregates	200,000	7	1,400,000	Anagnostopoulos (2018)
Skin Segmentation	245,057	3	735,171	Bhatt and Dhall (2009)
3D Road Network	434,874	3	1,304,622	Kaul (2013)
Covertype	581,012	10	5,810,120	Blackard (1998)

single loop. The CPU time for all algorithms is limited to 20 h per dataset, including all calculations up to 25 clusters.

We have employed the following metrics to compare the performance of the algorithms: cluster function values, relative errors, computational time, and Davies-Bouldin (DBI) and Dunn (DI) cluster validity indices. The relative error is defined in Appendix B, whereas Appendix A presents the definitions of the cluster validity indices. In our study, the DBI and DI indices are calculated only for the incremental algorithms Clust-Splitter, LMBM-Clust, DC-Clust, and Big-Clust. This is because these indices are computed for each intermediate result, not only for the final solution, allowing progressive tracking of clustering quality. Non-incremental algorithms produce only the final clustering solution, so the indices are not calculated for them.

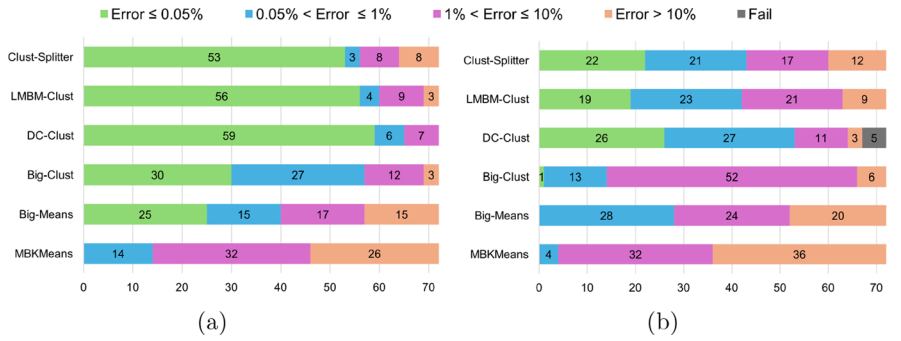


Fig. 2 Cases categorized by relative errors for (a) small k-clustering problems, where $k \leq 5$ (b) large k-clustering problems, where $k \geq 10$. In both categories, the total number of cases per algorithm is 72, comprising 18 datasets and four different numbers of clusters ($k = 2, 3, 4, 5$ for (a) and $k = 10, 15, 20, 25$ for (b)). Note that if an algorithm fails to find a solution within the 20-hour time limit, the case is assigned to the category 'Fail'.

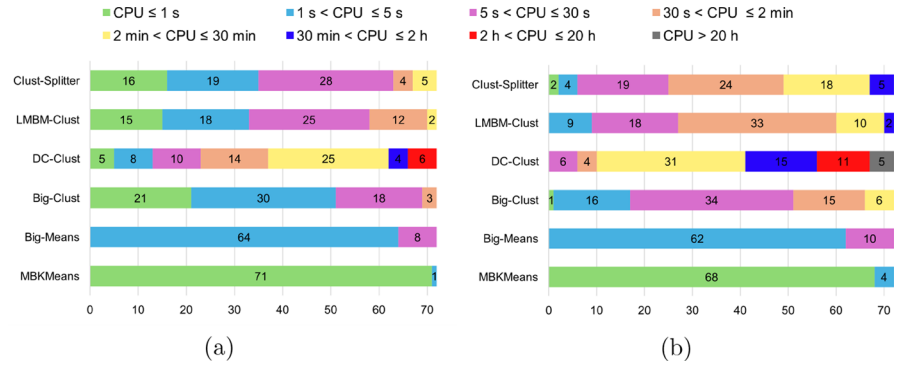


Fig. 3 Cases categorized by CPU times for (a) small k-clustering problems, where $k \leq 5$ (b) large k-clustering problems, where $k \geq 10$. In both categories, the total number of cases per algorithm is 72, comprising 18 datasets and four different numbers of clusters ($k = 2, 3, 4, 5$ for (a) and $k = 10, 15, 20, 25$ for (b)). Note that the category 'CPU > 20 h' denotes a failure to find a solution within the 20-hour time limit.

6.1.1 General overview of results

The results of the numerical experiments are summarized in Figs. 2 and 3, with more detailed results provided in Appendix B: Tables 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25 and Figs. 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21. In the following, a *case* refers to the problem of solving a dataset with a fixed number of clusters. Since we use different numbers of clusters, each dataset is thus divided into several cases. Each algorithm therefore has 144 cases in total (18 datasets and 8 cluster sizes). For analysis, we group the cluster sizes into small ($k = 2, 3, 4, 5$) and large ($k = 10, 15, 20, 25$), reducing the total number of cases per group to 72. The number of cases classified by relative errors and CPU times are illustrated in Figs. 2 and 3, respectively. In both figures, part (a) corresponds to small

k-clustering problems and part (b) to large *k*-clustering problems. In Appendix B, Tables 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25 present the detailed numerical results, including the best-known values of the *k*-clustering function, relative errors, and computational times for different algorithms. The validity indices for each dataset are summarized in Figs. 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, where part (a) represents the DBI values and part (b) corresponds to the DI values. Before proceeding with a detailed examination of the results, we first provide a general overview of the performance of each of the six methods.

MBKMeans is clearly the fastest method (see Fig. 3), but its accuracy is relatively poor: it never achieves a relative error below 0.05% and exceeds 10% more often than the other methods, both for small and large numbers of clusters (see Fig. 2). Big-Means is the second-fastest, but its accuracy is consistently lower than that of the four incremental algorithms, as it regularly produces high relative errors. Among the incremental algorithms, Big-Clust yields the highest relative errors. On the other hand, it is the fastest of the incremental methods, achieving the largest number of cases completed in under 5 s. DC-Clust produces very accurate results, but its runtimes are substantially longer than those of the other methods. For example, on the Covertype dataset its total runtime exceeded six hours and 30 min, whereas the other five methods required only 6.58−281.20 s. Moreover, within the 20-hour time limit, DC-Clust failed to find a solution in five cases, while all other methods succeeded. Overall, Clust-Splitter and LMBM-Clust demonstrate the best balance, frequently achieving relative errors below 0.05% while maintaining relatively fast computation times. Accordingly, we restrict our detailed analysis to Clust-Splitter and LMBM-Clust.

In the small *k*-clustering problems, LMBM-Clust is slightly more accurate than Clust-Splitter, achieving 56 cases with a relative error below 0.05%, compared to 53 cases with Clust-Splitter (see Fig. 2(a)). However, Clust-Splitter is slightly faster, with 35 cases having a CPU time below 5 s, compared to 33 cases for LMBM-Clust (see Fig. 3(a)). In the large *k*-clustering problems, Clust-Splitter performs better than LMBM-Clust in both accuracy and execution time: it achieves 22 cases with a relative error below 0.05%, compared to 19 for LMBM-Clust (see Fig. 2(b)), and has 2 cases with a CPU time below 1 s, whereas LMBM-Clust has none (see Fig. 3(b)). Nevertheless, the differences between Clust-Splitter and LMBM-Clust remain relatively small.

Next, we take a closer look at the results in Tables 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25. When examining the average relative error E_{aver}, Clust-Splitter performs better in 10 datasets, whereas LMBM-Clust achieves lower errors in 8 datasets. In terms of the total CPU time t_{total}, Clust-Splitter is faster in 3 datasets, whereas LMBM-Clust is faster in 15 datasets. However, when considering only the small *k*-clustering problems ($k = 2, 3, 4, 5$), Clust-Splitter has a lower computation time t_k in 37 cases, whereas LMBM-Clust is faster in 35 cases. These results suggest that Clust-Splitter achieves a slightly higher average accuracy and performs marginally faster for the small numbers of clusters. However, when computing up to 25 clusters, LMBM-Clust demonstrates better efficiency in terms of computational time. Consequently, if a case involves a large number of clusters, LMBM-Clust is the preferable choice. Conversely, for

problems with a smaller number of clusters, Clust-Splitter offers more accurate results with slightly faster computational times. The effectiveness of Clust-Splitter is further supported by its more intuitive selection of starting points compared to LMBM-Clust.

6.1.2 Results with validity indices

The DBI measures how well-separated clusters are, with lower values indicating better separation. In nearly all DBI visualizations, the four incremental algorithms produce closely related values (see, e.g., Figs. 7, 8, 9, 10), though differences become more apparent as k increases (see, e.g., Figs. 11 and 13). In 12 out of 18 datasets, Clust-Splitter produces DBI values identical to other methods for the small numbers of clusters ($k \leq 5$). The differences often arise from higher relative error values in Clust-Splitter (and in Big-Means and MBKMeans), suggesting that in some cases using partial data for clustering is not as effective as utilizing the full dataset. Furthermore, Clust-Splitter sometimes fails to find optimal solutions, resulting in higher DBI values (e.g., ISOLET at $k = 3$, relative error 0.54). However, in other cases, it still identifies the best solution despite increased DBI values (e.g., Gas Sensor Array Drift at $k = 4$). Importantly, even if Clust-Splitter produces a higher relative error for a specific k value, this does not mean it cannot find an optimal solution at other k values, including larger ones, for the same dataset. For instance, in the Protein Homology dataset, while the error is 1.82 at $k = 2$, Clust-Splitter finds the best known solution at $k = 3$ and $k = 4$. Overall, the DBI values of Clust-Splitter align closely with the other methods especially for small values of k, reinforcing its competitiveness in identifying well-separated clusters. The DBI also suggests that optimal number of clusters is generally small, as seen for example in EEG Eye State ($k = 4$) and MFCC Speech Emotion Recognition ($k = 3$), where the lowest DBI values occur.

The DI, which evaluates cluster compactness and separation (with higher values indicating better clustering quality), exhibits a similar trend to the DBI. For small values of k, the incremental algorithms produce nearly identical DI values, confirming that Clust-Splitter effectively identifies compact, well-separated clusters. However, its performance declines as k increases (see, e.g., Figs. 5 and 13). In addition, DI visualizations indicate that the optimal number of clusters is small. For instance, Online News Popularity and KEGG Metabolic datasets favor $k = 2$, while MFCC Speech Emotion Recognition dataset prefers $k = 3$. Notably, Clust-Splitter determines the same optimal number of clusters as other methods in 14 out of 18 datasets.

6.2 External validation and simulation study

In addition to the internal validation indices DBI and DI, we assess the performance of Clust-Splitter using two external validity measures: the proportion of objects that are correctly grouped together against the true clusters and the *adjusted Rand index* (ARI) (Hubert and Arabie 1985), a well-known index for comparing clustering algorithms. The exact definition of the ARI is provided in Appendix A. Furthermore,

we evaluate how accurately the method groups data points according to their true clusters. This approach follows the methodology outlined in Karmitsa et al. (2018).

6.2.1 Validation with real-world datasets

First, we apply Clust-Splitter, LMBM-Clust, DC-Clust, Big-Clust, Big-Means, and MBKMeans to three real-world datasets with known true cluster labels: Iris, Soybean, and Arcane (training set only). These datasets are publicly available from Markelle et al. (2025). The Iris dataset consists of 150 data points with four attributes and is classified into three clusters, each containing 50 data points. The Soybean dataset comprises 47 data points with 35 attributes and is divided into four clusters, with the first three clusters containing 10 data points each, whereas the fourth cluster has 17 data points. The Arcane dataset includes 100 data points with 10,000 attributes and is split into two clusters, with 44 data points in the first cluster and 56 in the second.

In our experiments, the number of clusters was set to match the known number of clusters in each original dataset. The algorithms were applied to the datasets without access to the known cluster labels. The performance of each method was evaluated based on accuracy, defined as the proportion of data points correctly grouped together against to the true clusters. In addition, we computed the ARI to further assess clustering quality. The results obtained for the different algorithms are presented in Tables 2, 3, 4.

The pairwise comparison of the clustering algorithms demonstrates that the Clust-Splitter method mainly achieves improved accuracy while it also yields in the most cases the highest ARI values. In the Iris dataset, Clust-Splitter attains the highest accuracy at 90.0%. For the Soybean dataset, it ranks third in accuracy, achieving 78.7%. In the Arcane dataset, Clust-Splitter achieves the highest accuracy at 52.0%. However, in the Arcane dataset, the negative ARI values indicate that clustering results are worse than random clustering across all the methods.

Overall, all the methods exhibit relatively similar performance across the datasets. The difference in accuracy within the Iris dataset is 1.3%, corresponding to the reassignment of three data points. In the Soybean dataset, the accuracy difference between the best and worst-performing methods is 17.1%, which translates to eight data points. In the Arcane dataset, the difference is 1.0%, meaning that one data point was assigned differently. In summary, Clust-Splitter has demonstrated good performance in these tests. Nevertheless, the differences between the algorithms remain relatively small.

6.2.2 Study with simulated data

The effectiveness of the Clust-Splitter method is further assessed using artificially generated datasets. These datasets were created in a two-dimensional space, incorporating varying proportions of outliers. Each dataset consists of three clusters (denoted as A, B, and C), with each cluster containing 120 data points. The coordinates of data points for the cluster A are sampled independently from the normal distributions $N(\mu_A^x, \sigma_A)$ and $N(\mu_A^y, \sigma_A)$, where μ_A^x and μ_A^y represent the mean values, and σ_A denotes the standard deviation. Similarly, the coordinates of data points for the

Table 2 Clustering results with Iris.

True cluster	Clust-Splitter			LMBM-Clust			DC-Clust			Big-Clust			Big-Means			MBKMeans		
	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
1	50	0	0	47	0	3	47	0	3	48	0	2	48	0	2	47	0	3
2	0	50	0	0	50	0	0	50	0	0	50	0	0	50	0	0	50	0
3	15	0	35	14	0	36	14	0	36	14	0	36	14	0	36	14	0	36
Accuracy (%)	90.0			88.7			88.7			89.3			89.3			88.7		
ARI	0.7455			0.7163			0.7163			0.7302			0.7302			0.7163		

Table 3 Clustering results with Soybean.

True cluster	Clust-Splitter				LMBM-Clust				DC-Clust				Big-Clust				Big-Means				MBKMeans			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	5	0	0	5	5	0	0	5	5	0	0	5	5	0	0	5	5	0	0	5	9	0	0	1
2	0	10	0	0	0	10	0	0	0	10	0	0	0	10	0	0	0	10	0	0	0	10	0	0
3	0	0	10	0	0	0	10	0	0	0	10	0	0	0	10	0	0	0	10	0	0	0	10	0
4	5	0	0	12	0	0	0	17	7	0	0	10	6	0	0	11	8	0	0	9	5	0	0	12
Accuracy (%)	78.7				89.4				74.5				76.6				72.3				87.2			
ARI	0.5814				0.7477				0.5513				0.5634				0.5452				0.6851			

Table 4 Clustering results with Arcane (training set only).

True cluster	CLUST-SPLITTER		LMBM-CLUST		DC-CLUST		BIG-CLUST		BIG-MEANS		MBKMEANS	
	1	2	1	2	1	2	1	2	1	2	1	2
1	17	27	17	27	17	27	17	27	17	27	17	27
2	21	35	22	34	21	35	22	34	22	34	21	35
Accuracy (%)	52.0		51.0		52.0		51.0		51.0		52.0	
ARI	−0.0087		−0.0099		−0.0087		−0.0099		−0.0099		−0.0087	

Table 5 Parameters for generating data points in artificial data.

	Cluster A	Cluster B	Cluster C
Mean μ^x	0	6	6
Mean μ^y	0	-1	2
Standard deviation σ	1.5	0.5	0.5
Outliers σ_C^O	–	–	2

Table 6 ARI values by different clustering algorithms in artificial data.

Outliers (%)	Clust-Splitter	LMBM-Clust	DC-Clust	Big-Clust	Big-Means	MBKMeans
0 %	0.9784	0.9776	0.9784	0.9705	0.9239	0.9238
10 %	0.9407	0.9415	0.9398	0.9407	0.8907	0.8890
20 %	0.9109	0.9141	0.9125	0.9108	0.8606	0.9125
30 %	0.8882	0.8890	0.8890	0.8874	0.8420	0.8416
40 %	0.8712	0.8706	0.8729	0.8750	0.8239	0.6510
50 %	0.8272	0.8250	0.8265	0.8279	0.7906	0.8248

clusters B and C are drawn from the corresponding normal distributions $N(\mu_B^x, \sigma_B)$, $N(\mu_B^y, \sigma_B)$, $N(\mu_C^x, \sigma_C)$, and $N(\mu_C^y, \sigma_C)$. In the cluster C, a designated proportion of data points, referred to as outliers, exhibit a greater standard deviation σ_C^O compared to the standard deviation σ_C of the remaining data points in the cluster. For instance, if 20% of the data points in the cluster C are outliers, then the coordinates of 24 data points are sampled from the normal distributions with σ_C^O, while the remaining 96 points follow the standard deviation σ_C. The parameters used to generate these simulated datasets are detailed in Table 5. Notably, the same parameter values as in Karmitsa et al. (2018) were used. To ensure robustness, datasets with varying proportions of outliers were generated, and for each proportion, ten different datasets were created. The results, presented in Table 6, represent the average performance across these ten datasets.

Table 6 shows that the ARI values for the different clustering algorithms are very close to each other. In particular, the four incremental algorithms − Clust-Splitter, LMBM-Clust, DC-Clust, and Big-Clust − exhibit highly similar ARI values. In contrast, the two other algorithms, Big-Means and MBKMeans, also yield comparable results to each other but with lower ARI values than obtained with the incremental algorithms, indicating weaker clustering performance. Overall, the differences in ARI values among the incremental algorithms were minimal, demonstrating that Clust-Splitter performed well in the tests, suggesting that it is capable of handling outliers effectively.

7 Conclusions

In this paper, we introduce a new clustering method, Clust-Splitter, for solving minimum sum-of-squares clustering problems, particularly in large-scale datasets. Clust-Splitter is an incremental algorithm, meaning that, in addition to solving the k_{max}-clustering problem for a given number of clusters k_{max}, it also solves all intermediate k-clustering problems for $k = 1, \ldots, k_{max} - 1$. The method consists of

three main components: a novel approach for selecting starting points based on cluster splitting with the help of two new auxiliary clustering problems, an incremental clustering algorithm and the limited memory bundle method, which is applied at each iteration to solve both the clustering and auxiliary clustering problems.

The proposed Clust-Splitter method was tested on 18 large-scale real-world datasets, with the number of data points ranging from tens of thousands to hundreds of thousands. It was compared against state-of-the-art clustering methods, including LMBM-Clust, DC-Clust, Big-Clust, Big-Means, and MBKMeans, using various evaluation metrics such as relative errors, two cluster validity indices, and CPU time. The results demonstrate that Clust-Splitter and LMBM-Clust are the most effective methods, as they achieve excellent accuracy while maintaining fast computation times. In contrast, MBKMeans is the fastest method but performs poorly in terms of accuracy. On the other hand, DC-Clust produces highly accurate solutions but requires significantly longer computation times. Big-Clust and Big-Means solve clustering problems relatively quickly and come close to the best known solutions, but they rarely achieve them. Since Clust-Splitter and LMBM-Clust frequently reach the best known solution within a reasonable time, they can be considered the best-performing methods in this study.

Additionally, while LMBM-Clust is the preferred choice for clustering more than ten clusters, Clust-Splitter is better suited for problems with small number of clusters. It efficiently identifies compact and well-separated clusters with low CPU time while often achieving or closely approximating the best known clustering function value. This advantage is further reinforced by the more intuitive selection of starting points in Clust-Splitter compared to LMBM-Clust, making it a practical choice for clustering tasks. Overall, these findings indicate that Clust-Splitter is a strong competitor alongside LMBM-Clust, particularly for real-time clustering in large-scale datasets.

Appendix A Definitions of validity indices

This appendix provides the definitions of three commonly used cluster validity indices: the Davies-Bouldin index (DBI) (Davies and Bouldin 1979), the Dunn index (DI) (Dunn 1974), and the adjusted Rand index (ARI) (Hubert and Arabie 1985). Throughout these definitions, the following notations are used. Let $A^1, \ldots, A^k$ represent a cluster distribution of the set A, where the number of clusters $k > 1$ and $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k$ are the cluster centers of these clusters. Let $d(\boldsymbol{x}^i, \boldsymbol{x}^j) = \sqrt{d_2(\boldsymbol{x}^i, \boldsymbol{x}^j)}$ denote the Euclidean distance between cluster centers $\boldsymbol{x}^i$ and $\boldsymbol{x}^j$.

The DBI is defined as

$$\mathrm{DBI} = \frac{1}{k} \sum_{i=1}^{k} \max_{j=1,\ldots,k,\ j \neq i} \frac{S_k(A^i) + S_k(A^j)}{d(\boldsymbol{x}^i, \boldsymbol{x}^j)}.$$

Table 7 Contingency table for comparing two partitions.

	V^1	V^2		V^s	Sums
U^1	n_{11}	n_{12}		n_{1s}	a_1
U^2	n_{21}	n_{22}		n_{2s}	a_2
			$\ddots$		
U^r	n_{r1}	n_{r2}		n_{rs}	a_r
Sums	b_1	b_2		b_s	

Here, $S_k(A^l)$ represents the average distance of all data points in the cluster A^l to its corresponding cluster center $\boldsymbol{x}^l$. The DBI ratio becomes smaller when the clusters are well-separated and compact. Therefore, the optimal number of clusters minimizes the DBI value.

The DI is given by

$$\mathrm{DI} = \frac{\min\limits_{i,j=1,\ldots,k,\ i\neq j} d(\boldsymbol{x}^i, \boldsymbol{x}^j)}{\max\limits_{l=1,\ldots,k}\left\{\max\limits_{\boldsymbol{a}\in A^l} \|\boldsymbol{x}^l - \boldsymbol{a}\|\right\}},$$

where the goal is to maximize inter cluster distances while minimizing intra cluster distances. Thus, the number of clusters that maximizes the DI can be regarded as the optimal number of clusters.

For the ARI, let $U = \{U^1, U^2, \ldots, U^r\}$ be the true partition of the data points, and $V = \{V^1, V^2, \ldots, V^s\}$ a clustering result. Then, the ARI for V is computed as

$$\mathrm{ARI} = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right]/\binom{n}{2}},$$

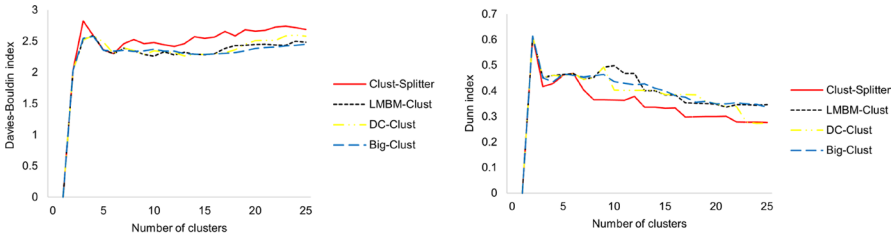
where the notation $\binom{n_{ij}}{2}$ denotes the binomial coefficient 'n_{ij} choose 2', n is the total number of data points in the dataset, and n_{ij}, a_i, and b_j are values derived from the contingency table given in Table 7. In this table, each entry n_{ij} represents the number of data points shared between clusters U_i and V_j. The ARI score ranges from -1 to 1, where 1 indicates a perfect clustering match, 0 corresponds to random clustering, and negative values suggest a clustering result worse than random assignment.

Appendix B Detailed numerical results

This appendix provides detailed numerical results in Tables 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25 and Figs. 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21. Before presenting the results, we briefly review the notations and measures used in the tables.

Table 8 Summary of the results with ISOLET ($\times 10^5$). Dimensions: $m = 7797$, $n = 617$.

k	f_{best}	Clust-Splitter		LMBM-Clust		DC-Clust		Big-Clust		Big-Means		MBKMeans	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	7.20988	0.00	6.02	0.00	4.11	0.00	66.63	0.26	4.60	0.13	4.74	0.27	0.13
3	6.77921	0.54	11.20	0.00	8.70	0.00	131.80	0.37	8.94	0.40	4.63	0.79	0.12
4	6.41487*	0.00	13.14	0.60	12.06	0.60	202.30	0.45	12.70	0.35	4.67	1.28	0.14
5	6.12976	0.00	16.53	0.00	17.03	1.01	271.64	0.49	18.29	0.50	4.59	1.68	0.16
10	5.28577	0.06	31.17	2.72	35.84	1.18	649.78	2.11	33.48	1.13	4.69	2.64	0.17
15	4.86785	0.89	44.34	0.10	58.47	0.00	1068.70	2.57	51.17	1.28	4.71	2.30	0.18
20	4.60260	0.46	63.48	0.02	81.55	0.00	1606.58	2.56	73.25	1.69	4.70	2.90	0.20
25	4.43890	0.30	88.52	0.58	106.28	0.27	2196.53	2.59	98.13	0.89	4.70	2.49	0.20
E_{aver}		0.28		0.50		0.38		1.42		0.80		1.80	
t_{init}			6.52		6.22		6.23		6.28		1.22		1.17
t_{total}			95.03		112.50		2202.77		104.40		38.65		2.47



(a) (b)

Fig. 4 ISOLET: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Cluster function values are also referred to as the sum of squared errors (SSE), a prototype-based cohesion measure evaluating the average variation within clusters. The tables include the best-known cluster function (5) value, f_{best} (scaled by the dataset-specific multiplier shown after the name of the dataset), and the relative errors for each algorithm. The *relative error* E_k for a dataset with k clusters is calculated as

$$E_k = \frac{f_k - f_{best}}{f_{best}} \times 100\%,$$

where f_k is the k-clustering function (5) value obtained by the algorithm. We adopt the f_{best} values reported in Karmitsa et al. (2025) unless better results are achieved in our experiments. In this study, these better values are marked with an asterisk. In addition, we give the average of relative errors E_{aver} for each method, and it is calculated as $E_{\text{aver}} = \frac{1}{8}\sum_{k\in K} E_k$, where $K = \{2, 3, 4, 5, 10, 15, 20, 25\}$.

The tables present the time measurements (in seconds) separately for different phases: the time required to read the data (t_{init}), the time taken to compute k clusters (t_k), and the total computation time (t_{total}) for all clusters. For the Clust-Splitter, LMBM-Clust, DC-Clust, and Big-Clust, the total time is calculated as: $t_{\text{total}} = t_{\text{init}} + t_{25}$. In contrast, for Big-Means and MBKMeans, the total time follows the formula $t_{\text{total}} = t_{\text{init}} + \sum_{k\in K} t_k$, where $K = \{2, 3, 4, 5, 10, 15, 20, 25\}$.

Table 9 Summary of the results with Gisette ($\times 10^{12}$). Dimensions: $m = 13{,}500$, $n = 5000$.

k	f_{best}	CLUST-SPLITTER		LMBM-CLUST		DC-CLUST		BIG-CLUST		BIG-MEANS		MBKMEANS	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	4.19944	0.00	23.58	0.00	48.66	0.00	4401.03	0.17	27.63	0.01	27.18	0.30	0.95
3	4.11596	0.00	182.97	0.00	103.64	0.00	9417.34	0.27	53.58	0.02	27.14	0.35	0.95
4	4.06539	0.04	310.44	0.00	164.19	0.00	15,448.19	0.38	82.75	0.08	27.19	0.37	0.98
5	4.02235*	0.00	404.20	0.02	236.16	0.02	21,447.47	0.49	110.61	0.09	26.65	0.38	1.21
10	3.87672	0.02	940.64	0.20	644.52	0.03	57,633.72	1.70	290.43	0.16	27.40	0.61	1.29
15	3.80098*	0.00	1603.30	0.14	1207.41	–	–	2.39	523.98	0.10	27.57	0.59	1.23
20	3.74592*	0.00	2554.77	0.43	1842.69	–	–	3.01	847.30	0.13	27.81	0.60	1.58
25	3.70152*	0.00	3634.23	1.51	2229.50	–	–	3.62	1285.57	0.20	27.68	0.66	1.48
E_{aver}		0.01		0.29		0.01		1.50		0.10		0.48	
t_{init}			64.30		61.89		63.55		61.86		21.12		17.90
t_{total}			3698.53		2291.39		57,697.27		1347.43		239.75		27.57

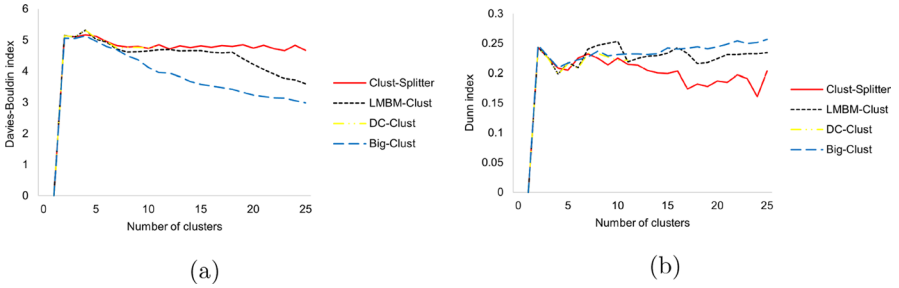


Fig. 5 Gisette: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 10 Summary of the results with Gas Sensor Array Drift ($\times 10^{13}$). Dimensions: $m = 13{,}910$, $n = 128$.

k	f_{best}	CLUST-SPLITTER		LMBM-CLUST		DC-CLUST		BIG-CLUST		BIG-MEANS		MBKMEANS	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	7.91186*	0.00	3.91	0.00	27.72	0.00	11.25	0.03	0.63	0.07	4.63	0.74	0.06
3	5.02412	0.00	7.70	0.00	32.14	0.00	27.73	0.10	1.61	0.13	4.64	2.97	0.05
4	3.97506*	0.00	11.20	4.61	32.94	4.63	50.75	4.75	2.22	1.49	4.64	3.64	0.05
5	3.22394	0.00	12.08	0.00	34.53	0.10	75.73	0.12	2.89	4.60	4.23	13.12	0.04
10	1.65230	0.18	23.23	0.18	48.50	0.18	245.08	5.85	5.87	3.52	4.65	26.89	0.06
15	1.13801	2.62	45.89	0.36	56.98	0.36	500.20	0.58	8.94	4.20	4.24	18.89	0.09
20	0.87916	1.99	68.27	2.79	63.45	0.61	771.25	3.07	12.89	5.37	4.54	17.02	0.08
25	0.72211	2.65	88.25	4.31	70.48	0.65	1051.48	3.46	17.33	4.61	4.66	23.73	0.06
E_{aver}		0.93		1.53		0.82		2.24		3.00		13.38	
t_{init}			2.53		2.42		2.44		2.50		0.47		0.45
t_{total}			90.78		72.91		1053.92		19.83		36.69		0.94

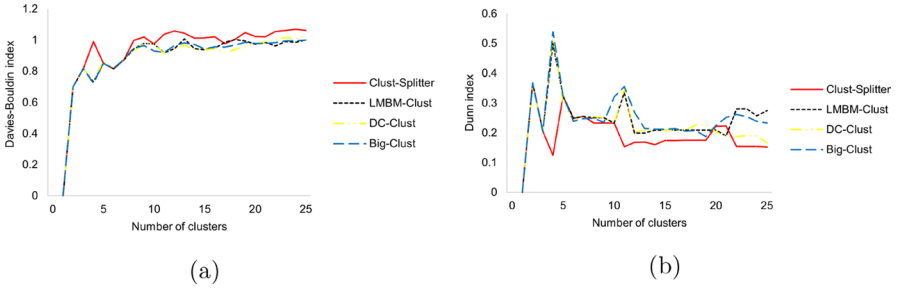


Fig. 6 Gas Sensor Array Drift: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 11 Summary of the results with EEG Eye State ($\times 10^8$). Dimensions: $m = 14{,}980$, $n = 14$.

k	f_{best}	Clust-Splitter		LMBM-Clust		DC-Clust		Big-Clust		Big-Means		MBKMeans	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	7845.09934	4.25	0.89	4.25	0.05	4.25	0.13	4.25	0.09	62.02	3.25	98.14	0.03
3	1833.88058	0.00	2.20	0.00	0.08	0.00	0.33	0.01	0.14	663.56	3.30	746.99	0.02
4	2.23605	0.00	3.00	0.00	0.11	0.00	0.63	0.01	0.18	510,690.30	3.28	68,9042.19	0.03
5	1.33858	0.00	3.19	0.00	0.17	0.00	1.33	0.01	0.25	882,321.25	3.39	1,154,863.91	0.03
10	0.45306*	0.00	4.09	0.80	2.25	0.81	9.05	1.06	0.80	2,583,946.58	3.33	3,417,200.94	0.03
15	0.34653	0.62	5.78	0.05	4.73	0.26	22.50	1.78	1.34	3,202,194.92	3.27	4,376,517.58	0.03
20	0.28986	0.75	9.47	0.00	7.58	1.34	44.09	2.41	1.98	3,000,995.30	3.35	5,212,690.01	0.03
25	0.25989	0.29	13.11	0.16	11.50	0.13	68.53	2.51	2.73	5,142,507.29	3.28	5,723,176.90	0.03
E_{aver}		0.74		0.66		0.85		1.51		1,915,422.65		2,571,792.08	
t_{init}			0.30		0.30		0.30		0.31		0.06		0.06
t_{total}			13.41		11.80		68.83		3.05		26.52		0.28

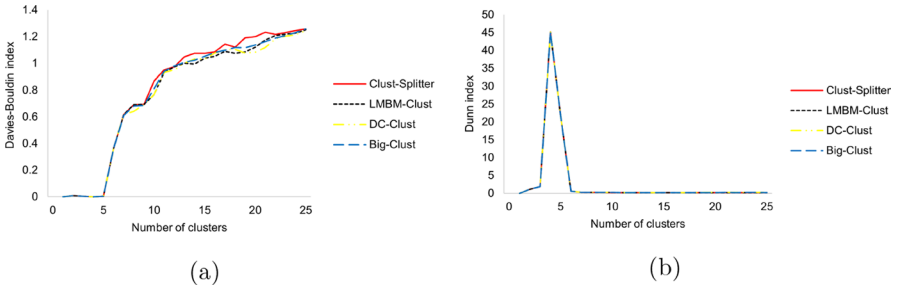


Fig. 7 EEG Eye State: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 12 Summary of the results with D15112 ($\times 10^{11}$). Dimensions: $m = 15{,}112$, $n = 2$. Online News Popularity.

k	f_{best}	Clust-Splitter		LMBM-Clust		DC-Clust		Big-Clust		Big-Means		MBKMeans	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	3.68403	0.00	0.02	0.00	1.08	0.00	0.75	0.05	0.19	0.01	3.35	0.30	0.06
3	2.53240	0.00	0.11	0.00	1.48	0.00	1.41	0.12	0.35	0.02	3.26	2.17	0.05
4	1.73600	0.00	0.17	0.00	1.81	0.00	1.81	0.16	0.50	0.02	3.40	8.27	0.05
5	1.32707*	0.00	0.23	0.00	2.11	0.00	2.23	0.17	0.64	0.03	3.26	3.46	0.05
10	0.64490*	0.00	0.55	1.41	2.98	0.62	5.02	0.75	1.39	1.18	3.31	8.61	0.05
15	0.43136	0.00	0.91	0.24	3.61	0.25	9.95	1.53	2.07	1.12	3.27	8.30	0.05
20	0.32177	0.92	1.44	0.24	4.30	0.24	15.39	2.14	3.11	1.17	3.26	6.00	0.05
25	0.25308	1.04	1.95	0.48	4.94	0.47	24.52	2.86	4.16	0.71	3.26	7.75	0.06
E_{aver}		0.25		0.30		0.20		0.97		0.53		5.61	
t_{init}			0.05		0.05		0.03		0.05		0.01		0.00
t_{total}			2.00		4.98		24.55		4.20		26.38		0.42

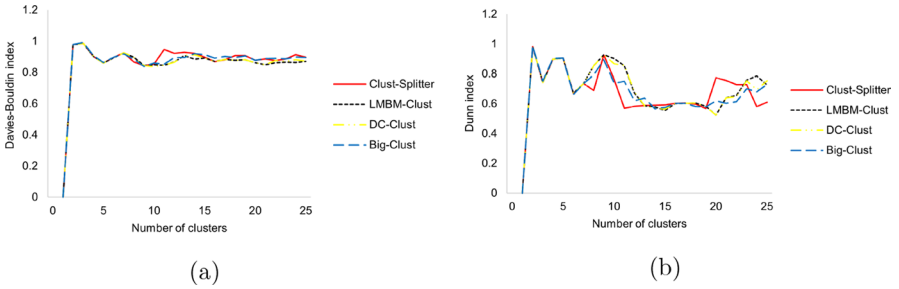


Fig. 8 D15112: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 13 Summary of the results with online news popularity ($\times 10^{14}$). Dimensions: $m = 39{,}644$, $n = 58$.

k	f_{best}	CLUST-SPLITTER		LMBM-CLUST		DC-CLUST		BIG-CLUST		BIG-MEANS		MBKMEANS	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	9.53913*	0.00	11.28	0.00	1.86	0.00	18.08	0.02	0.62	0.02	4.26	2.13	0.10
3	5.91077	0.00	16.89	0.00	26.56	0.00	45.73	0.04	1.07	0.09	4.25	3.32	0.05
4	4.30793	7.66	22.13	7.66	27.28	0.00	76.89	7.70	1.55	3.92	4.26	2.16	0.05
5	3.09885	0.00	27.19	0.00	39.12	0.00	103.64	0.07	2.07	1.88	4.26	14.62	0.05
10	1.17247	2.58	52.80	2.57	54.06	0.00	303.48	8.29	4.39	5.85	4.27	35.42	0.06
15	0.77637	0.00	97.92	14.77	65.48	0.00	577.20	21.36	7.40	6.55	4.50	38.53	0.07
20	0.59809	1.93	131.52	19.49	80.97	0.00	990.88	14.24	11.18	4.37	4.26	25.08	0.07
25	0.49616	1.55	164.88	7.03	93.77	0.00	1443.78	9.53	15.46	5.23	4.28	32.91	0.08
E_{aver}		1.72		6.44		0.00		7.66		3.49		19.27	
t_{init}			2.95		2.66		2.69		2.70		0.49		0.50
t_{total}			167.83		96.42		1446.47		18.16		34.83		1.04

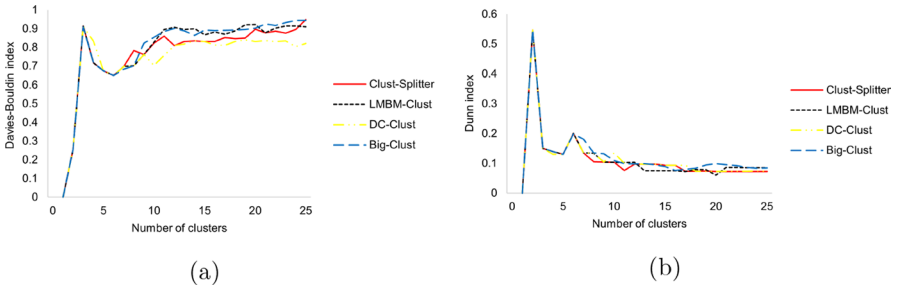


Fig. 9 Online News Popularity: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 14 Summary of the results with KEGG Metabolic ($\times 10^8$). Dimensions: $m = 53{,}413$, $n = 20$.

k	f_{best}	Clust-Splitter		LMBM-Clust		DC-Clust		Big-Clust		Big-Means		MBKMeans	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	11.38530	0.00	0.22	0.00	0.25	0.00	2.81	0.01	0.30	5.89	4.27	23.66	0.06
3	4.90060*	0.00	0.61	0.00	0.53	0.00	8.05	0.21	0.49	0.69	4.27	160.13	0.03
4	2.72950	0.01	1.00	0.00	0.91	0.00	17.42	2.04	0.77	1.10	4.27	316.62	0.04
5	1.88367	0.00	2.02	0.00	1.47	0.00	31.30	2.24	1.05	1.70	4.27	482.70	0.03
10	0.60513	5.17	7.05	4.96	4.05	5.00	139.17	9.65	2.34	38.11	4.27	1516.83	0.03
15	0.34940*	0.00	20.02	2.01	8.45	0.93	308.75	9.24	3.57	86.29	4.39	2542.99	0.05
20	0.25027	0.47	35.33	7.72	12.33	0.31	477.13	10.55	4.92	88.20	4.28	3679.29	0.04
25	0.19253*	0.00	67.94	2.83	17.69	2.11	692.36	9.23	6.36	120.35	4.28	4692.85	0.06
E_{aver}		0.71		2.19		1.04		5.40		42.79		1676.89	
t_{init}			1.41		1.28		1.30		1.36		0.24		0.27
t_{total}			69.34		18.97		693.66		7.72		34.54		0.61

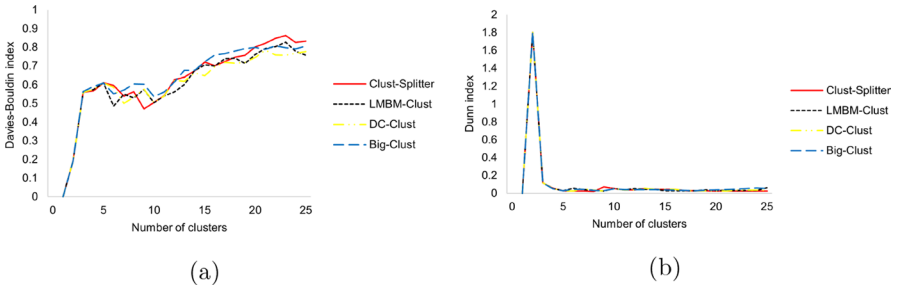
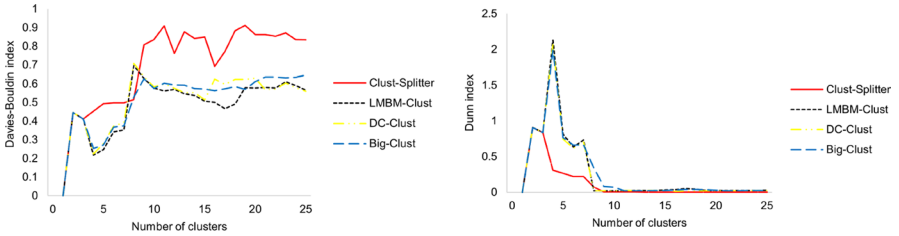


Fig. 10 KEGG Metabolic: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 15 Summary of the results with Shuttle Control ($\times 10^8$). Dimensions: $m = 58{,}000$, $n = 9$.

k	f_{best}	CLUST-SPLITTER		LMBM-CLUST		DC-CLUST		BIG-CLUST		BIG-MEANS		MBKMEANS	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	21.34329	0.00	0.13	0.00	0.23	0.00	0.47	0.00	0.36	9.80	3.26	52.07	0.03
3	10.85415	0.00	0.36	0.00	0.39	0.00	1.47	0.01	0.51	22.85	3.31	196.60	0.04
4	8.86910	5.28	0.52	0.00	0.55	0.39	3.66	3.33	0.74	17.39	3.25	260.32	0.03
5	7.24479	11.00	1.73	0.09	0.70	0.44	7.81	2.86	0.96	39.90	3.26	333.09	0.05
10	2.83216	61.76	4.55	0.55	1.95	0.24	45.66	4.31	2.34	64.84	3.26	990.79	0.03
15	1.53154	128.43	10.27	0.02	3.28	3.59	192.28	12.18	3.64	114.08	3.26	1860.54	0.05
20	1.05032	196.01	16.81	0.96	5.33	0.00	292.42	10.20	5.00	190.94	3.27	2677.05	0.04
25	0.77978	275.97	23.77	0.00	8.61	3.55	414.50	10.57	6.59	140.21	3.27	3556.60	0.03
E_{aver}		84.80		0.20		1.03		5.43		75.00		1240.88	
t_{init}			0.63		0.58		0.58		0.60		0.18		0.17
t_{total}			24.39		9.19		415.08		7.19		26.32		0.47



(a) (b)

Fig. 11 Shuttle Control: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 16 Summary of the results with Sensorless Drive Diagnosis ($\times 10^7$). Dimensions: $m = 58{,}509$, $n = 48$.

k	f_{best}	CLUST-SPLITTER		LMBM-CLUST		DC-CLUST		BIG-CLUST		BIG-MEANS		MBKMEANS	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	3.88116	1.51	0.42	1.51	0.44	1.51	7.50	1.81	0.66	57.81	4.59	101.67	0.04
3	2.91313	26.29	0.92	2.01	0.62	3.93	31.72	7.26	0.95	57.04	4.32	158.89	0.03
4	2.26160	33.94	3.66	2.59	1.33	4.94	62.64	11.97	1.32	83.97	4.27	221.94	0.03
5	1.93651	39.65	5.53	10.52	1.75	5.55	110.50	9.47	1.77	92.16	4.33	271.89	0.04
10	0.96090	118.11	29.19	10.50	5.50	7.87	428.95	8.20	3.51	111.71	4.28	603.18	0.04
15	0.62816	206.18	58.34	9.31	20.25	10.91	878.84	5.12	6.76	141.36	4.28	944.30	0.04
20	0.49884	270.39	90.19	14.28	30.81	11.79	1346.34	6.93	9.96	269.20	4.32	1201.56	0.04
25	0.42225	324.60	132.53	15.89	44.22	12.82	1922.28	8.20	13.56	286.79	4.43	1420.27	0.04
E_{aver}		127.58		8.33		7.41		7.37		137.50		615.46	
t_{init}			4.27		3.89		3.95		3.91		0.77		0.75
t_{total}			136.80		48.11		1926.23		17.47		35.58		1.05

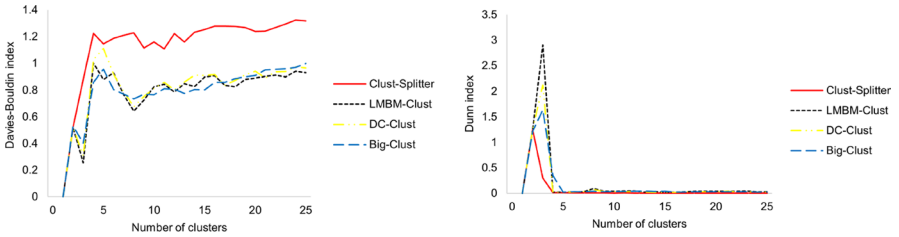


Fig. 12 Sensorless Drive Diagnosis: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 17 Summary of the results with MFCCs for Speech Emotion Recognition ($\times 10^8$). Dimensions: $m = 85{,}134$, $n = 58$.

k	f_{best}	Clust-Splitter		LMBM-Clust		DC-Clust		Big-Clust		Big-Means		MBKMeans	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	7.45130	0.00	1.81	0.00	6.45	0.00	125.27	0.84	1.60	0.01	4.29	3.25	0.12
3	5.02150	0.00	6.03	0.00	8.41	0.00	250.03	0.09	2.54	0.01	4.29	5.53	0.14
4	4.16900	1.67	10.44	0.00	11.03	0.00	413.13	0.15	4.21	0.51	4.41	1.59	0.10
5	3.45592	0.00	13.16	0.00	15.66	0.00	574.66	0.25	6.11	0.01	4.35	5.01	0.13
10	2.17618	2.47	33.11	0.00	29.59	0.00	1464.70	1.13	12.28	0.98	4.29	2.18	0.14
15	1.76044	0.00	71.27	1.19	47.28	0.00	2478.06	1.93	19.57	0.87	4.31	3.41	0.12
20	1.53799*	0.00	121.84	2.57	63.44	0.31	3627.80	3.65	26.23	1.01	4.33	3.69	0.17
25	1.41090	0.93	174.50	0.74	84.98	0.61	4983.70	2.61	34.82	1.01	4.31	3.59	0.14
E_{aver}		0.63		0.56		0.12		1.33		0.55		3.53	
t_{init}			7.23		6.94		6.94		6.95		2.08		2.05
t_{total}			181.73		91.92		4990.64		41.77		36.65		3.11

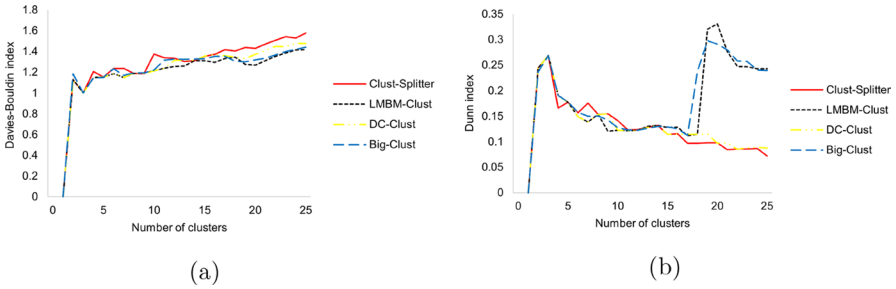


Fig. 13 MFCCs for Speech Emotion Recognition: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 18 Summary of the results with Pla85900 ($\times 10^{15}$). Dimensions: $m = 85{,}900$, $n = 2$.

k	f_{best}	CLUST-SPLITTER		LMBM-CLUST		DC-CLUST		BIG-CLUST		BIG-MEANS		MBKMEANS	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	3.74908	1.44	1.53	1.44	3.30	0.00	7.72	1.80	0.50	1.01	3.27	1.48	0.12
3	2.28057	0.00	2.67	0.00	4.69	0.00	15.09	0.02	0.84	0.01	3.26	2.43	0.12
4	1.59308*	0.00	3.19	0.00	5.88	0.00	22.13	0.03	1.17	0.01	3.33	3.99	0.16
5	1.33972	0.81	3.77	2.77	6.91	0.00	31.02	0.03	1.48	0.81	3.27	1.61	0.13
10	0.68294	0.55	7.06	0.40	16.36	0.00	84.81	1.49	3.03	0.42	3.27	3.98	0.12
15	0.46029	0.16	11.69	0.98	22.52	0.48	143.27	1.26	4.78	0.36	3.27	4.95	0.14
20	0.34988	1.60	23.23	0.94	28.34	0.52	203.89	0.76	6.82	0.52	3.27	5.65	0.15
25	0.28259	0.29	34.00	0.18	37.34	0.02	274.70	1.23	9.17	0.70	3.27	4.60	0.11
E_{aver}		0.61		0.84		0.13		0.83		0.48		3.59	
t_{init}			0.27		0.25		0.25		0.27		0.08		0.08
t_{total}			34.27		37.59		274.95		9.43		26.29		1.13

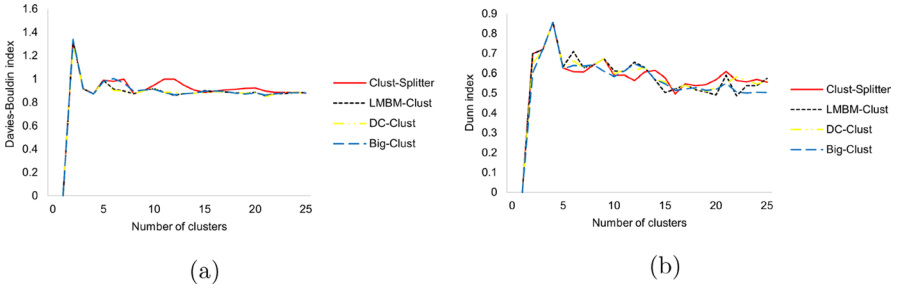
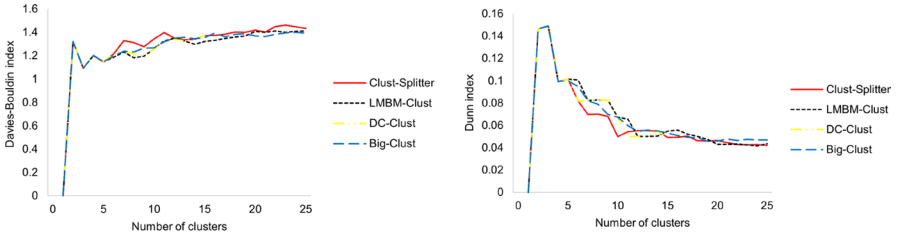


Fig. 14 Pla85900: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 19 Summary of the results with Music Analysis ($\times 10^{11}$). Dimensions: $m = 106{,}574$, $n = 518$.

k	f_{best}	Clust-Splitter		LMBM-Clust		DC-Clust		Big-Clust		Big-Means		MBKMeans	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	5.00470	0.00	11.77	0.00	16.48	0.00	4137.12	0.03	7.96	0.00	3.36	0.21	0.36
3	3.83746	0.00	50.09	0.00	31.08	0.00	7559.89	0.09	11.72	4.54	3.35	5.36	0.30
4	3.11830	0.00	111.72	0.00	63.42	0.00	12,042.67	0.13	17.83	3.95	3.38	7.98	0.53
5	2.74247	0.00	138.00	0.00	85.59	0.00	16,183.20	0.20	22.71	0.91	3.36	5.47	0.50
10	1.87257	1.70	448.25	0.00	195.33	0.00	41,252.87	0.79	56.11	1.00	3.60	4.75	0.34
15	1.54420	0.42	1049.08	0.52	392.30	0.42	68,087.94	1.48	107.47	0.88	4.37	3.43	0.41
20	1.35320	0.10	1824.34	0.08	628.92	–	–	2.45	161.17	0.59	5.98	3.88	0.41
25	1.22620	1.33	2701.72	1.64	908.87	–	–	3.12	224.96	0.73	6.44	3.94	0.44
E_{aver}		0.44		0.28		0.07		1.03		1.57		4.38	
t_{init}			78.52		74.39		74.41		74.19		14.14		14.14
t_{total}			2780.23		983.27		68,162.34		299.15		47.99		17.42



(a) (b)

Fig. 15 Music Analysis: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 20 Summary of the results with MiniBooNE Particle Identification ($\times 10^{10}$). Dimensions: $m = 130{,}064$, $n = 50$.

k	f_{best}	Clust-Splitter		LMBM-Clust		DC-Clust		Big-Clust		Big-Means		MBKMeans	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	8.92236	252,620.64	26.42	0.00	0.98	0.00	4.41	0.00	1.15	286,911.70	4.27	286,901.67	0.07
3	5.22601	404,522.85	55.98	21.68	1.70	0.00	102.80	21.69	1.97	391,916.32	4.28	489,605.83	0.07
4	2.70080	730,252.51	94.44	0.00	3.05	0.00	189.31	0.04	2.99	947,964.98	4.46	946,975.49	0.07
5	1.82252	1,006,752.24	131.94	0.00	6.84	0.00	441.72	0.12	5.71	1,264,307.71	4.42	1,404,755.54	0.08
10	0.90920	1,348,100.88	487.41	1.63	31.06	1.65	2025.89	2.37	15.03	2,815,972.07	4.31	2,813,472.85	0.07
15	0.63506	1,165,473.95	1059.80	0.00	59.73	0.02	3897.16	1.47	22.30	4,031,566.07	4.33	4,029,153.17	0.09
20	0.50863	739,691.13	1731.14	1.17	99.87	0.37	6035.58	2.30	31.07	4,530,328.25	4.41	5,006,683.99	0.09
25	0.44425	309,221.62	2535.77	1.12	141.33	0.03	8274.08	1.76	41.59	5,763,170.88	4.41	5,754,541.94	0.08
E_{aver}		744,579.48		3.20		0.26		3.72		2,504,017.25		2,591,511.31	
t_{init}			9.42		9.08		9.08		9.57		1.83		1.86
t_{total}			2545.19		150.41		8283.16		131.36		36.72		2.47

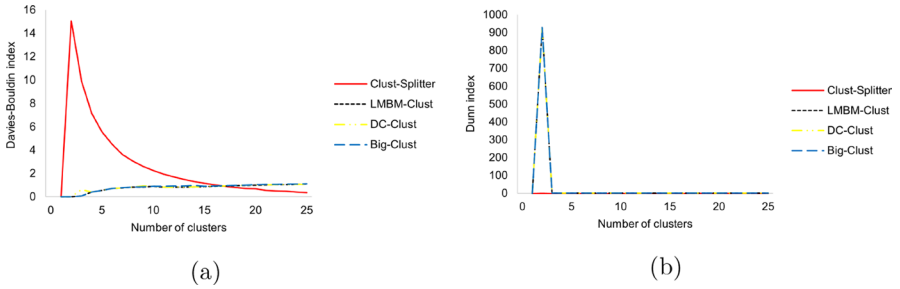


Fig. 16 MiniBooNE Particle Identification: both (a) Davies–Bouldin and (a) Dunn validity indices versus number of clusters.

Table 21 Summary of the results with Protein Homology ($\times 10^{11}$). Dimensions: $m = 145{,}751$, $n = 74$.

k	f_{best}	Clust-Splitter		LMBM-Clust		DC-Clust		Big-Clust		Big-Means		MBKMeans	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	15.20430	1.82	2.55	0.00	1.52	0.00	167.09	0.03	2.36	1.21	26.30	2.67	0.13
3	8.07129	0.00	7.56	0.00	5.13	0.00	457.25	0.15	3.14	1.04	26.29	63.92	0.08
4	6.09898*	0.00	16.44	19.54	7.14	0.00	958.56	13.87	4.15	1.11	26.43	104.58	0.06
5	5.30536*	0.00	21.77	0.41	10.84	0.50	1556.91	0.58	5.28	1.34	26.32	107.18	0.08
10	3.37658*	0.00	81.66	0.04	39.94	0.00	4920.78	4.30	13.40	2.48	26.41	170.44	0.08
15	2.86364*	0.00	264.69	2.49	92.27	1.58	8541.13	5.50	22.87	0.54	26.39	198.61	0.09
20	2.57320	0.74	416.70	3.11	167.95	0.77	12,588.48	4.19	35.20	0.75	26.45	163.11	0.12
25	2.38540	1.54	630.36	0.75	242.45	0.26	17,222.64	2.68	48.77	1.32	26.42	131.92	0.14
E_{aver}		0.51		3.29		0.39		3.91		1.22		117.80	
t_{init}			14.72		14.19		14.23		14.15		2.70		2.66
t_{total}			645.08		256.64		17,236.88		62.92		213.71		3.44

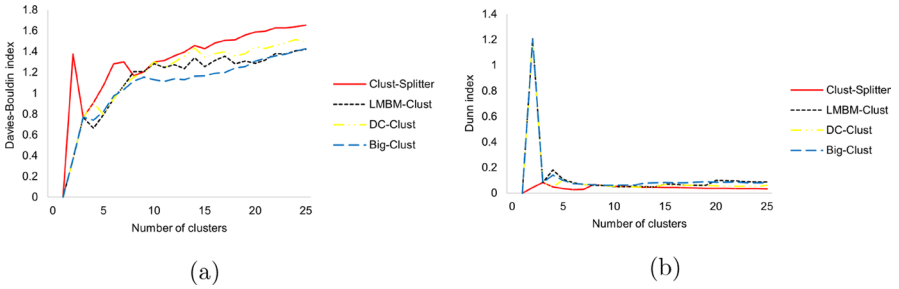


Fig. 17 Protein Homology: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 22 Summary of the results with Range Queries Aggregates ($\times 10^{14}$). Dimensions: $m = 200{,}000$, $n = 7$.

k	f_{best}	CLUST-SPLITTER		LMBM-CLUST		DC-CLUST		BIG-CLUST		BIG-MEANS		MBKMEANS	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	16.39968	0.00	5.55	0.00	2.11	0.00	90.66	0.01	1.21	0.01	4.26	0.20	0.15
3	8.22970	0.00	8.53	0.00	5.25	0.00	222.08	0.02	3.24	0.03	4.25	1.35	0.16
4	5.06319*	0.00	12.11	0.00	7.44	0.00	339.45	0.03	4.93	0.03	4.27	3.32	0.12
5	3.49938*	0.00	16.28	0.00	9.94	0.00	457.86	0.05	6.22	0.04	4.34	7.42	0.09
10	1.35277*	0.00	47.23	0.01	25.34	0.01	1048.14	0.20	11.04	1.59	4.43	3.77	0.17
15	0.93917	0.00	77.52	0.00	43.62	0.00	1616.27	1.17	15.18	0.76	4.39	3.36	0.21
20	0.74585	0.14	109.98	0.17	59.34	0.21	2234.92	0.00	20.20	0.21	4.27	3.56	0.17
25	0.62330*	0.00	158.28	1.40	73.62	0.46	2920.66	0.74	24.79	0.49	4.28	5.04	0.14
E_{aver}		0.02		0.20		0.09		0.28		0.40		3.50	
t_{init}			2.11		2.03		2.03		2.12		0.54		0.56
t_{total}			160.39		75.66		2922.69		26.91		35.02		1.77

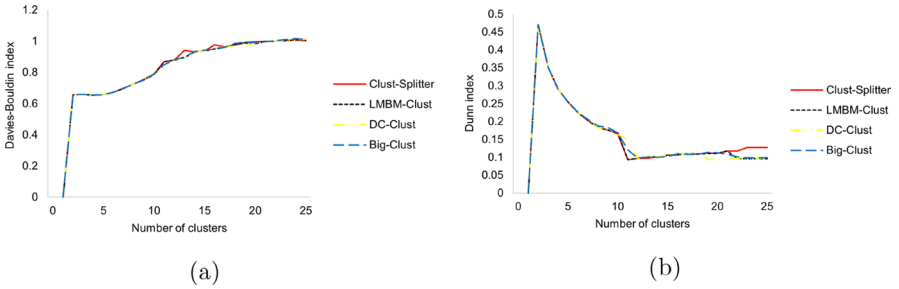


Fig. 18 Range Queries Aggregates: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 23 Summary of the results with Skin Segmentation ($\times 10^9$). Dimensions: $m = 245{,}057$, $n = 3$.

k	f_{best}	Clust-Splitter		LMBM-Clust		DC-Clust		Big-Clust		Big-Means		MBKMeans	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	1.32236	0.00	0.45	0.00	0.59	0.00	69.61	0.02	1.12	0.00	3.28	0.08	0.16
3	0.89362	0.00	0.88	0.00	1.34	0.00	130.41	0.03	1.70	0.01	3.36	5.20	0.13
4	0.63998	8.59	2.61	8.59	2.22	0.00	183.75	0.04	2.53	1.92	3.26	3.90	0.20
5	0.50203	0.00	3.14	0.00	3.08	0.00	243.22	1.36	3.16	1.60	3.31	10.48	0.11
10	0.25121	4.99	6.30	13.37	6.87	0.00	505.56	7.88	7.44	5.58	3.47	9.16	0.14
15	0.16688*	0.00	11.95	26.76	11.78	1.84	762.83	4.93	11.02	5.35	3.26	11.20	0.12
20	0.12615	0.25	19.25	11.56	17.34	0.37	1030.64	5.76	15.04	3.95	3.27	10.70	0.12
25	0.10228	3.64	28.03	14.43	23.31	0.00	1341.83	8.97	20.57	4.78	3.30	7.64	0.23
E_{aver}		2.18		9.34		0.28		3.62		2.90		7.30	
t_{init}			1.02		0.91		0.91		1.01		0.29		0.30
t_{total}			29.05		24.22		1342.73		21.57		26.78		1.51

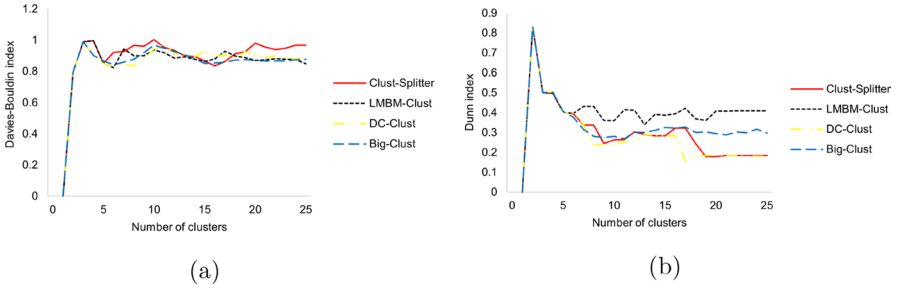


Fig. 19 Skin Segmentation: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 24 Summary of the results with 3D Road Network ($\times 10^6$). Dimensions: $m = 434{,}874$, $n = 3$.

k	f_{best}	CLUST-SPLITTER		LMBM-CLUST		DC-CLUST		BIG-CLUST		BIG-MEANS		MBKMEANS	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	49.13298	0.00	0.72	0.00	19.44	0.00	195.39	0.02	3.76	0.01	4.24	0.62	0.20
3	22.77818	0.00	1.52	0.00	21.47	0.00	480.98	0.03	6.77	0.03	4.42	1.19	0.30
4	13.52389	0.00	2.08	0.00	23.45	0.00	734.70	0.03	8.97	0.05	4.26	2.87	0.15
5	8.82574	0.00	2.73	0.00	25.59	0.00	1001.58	0.04	11.14	0.08	4.27	3.81	0.25
10	2.56661	0.00	10.00	0.00	35.64	0.02	2316.09	0.34	17.59	0.26	4.32	5.36	0.13
15	1.27069	0.00	24.36	0.00	47.30	0.00	3686.33	0.79	22.94	0.60	4.27	8.37	0.25
20	0.80865*	0.00	59.50	0.00	63.95	0.01	5077.50	0.68	28.97	0.71	4.28	11.04	0.23
25	0.59258	1.60	92.55	0.00	87.08	3.78	6568.56	1.40	36.30	1.10	4.36	10.91	0.06
E_{aver}		0.20		0.00		0.48		0.42		0.35		5.52	
t_{init}			2.36		2.13		2.14		2.24		0.66		0.66
t_{total}			94.91		89.20		6570.70		38.55		35.07		2.23

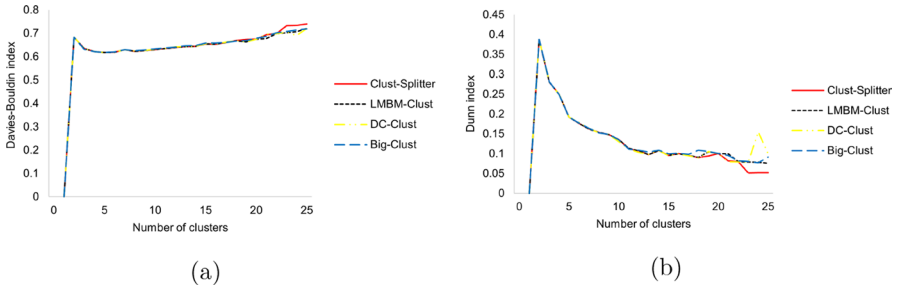


Fig. 20 3D Road Network: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Table 25 Summary of the results with Covertype ($\times 10^{11}$). Dimensions: $m = 581{,}012$, $n = 10$.

k	f_{best}	Clust-Splitter		LMBM-Clust		DC-Clust		Big-Clust		Big-Means		MBKMeans	
		E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k	E_k	t_k
2	13.41885	0.00	1.38	0.00	29.47	0.00	715.53	0.01	3.88	0.00	4.31	0.34	0.38
3	9.52870	0.00	5.31	0.00	31.84	0.00	1535.64	0.02	6.99	0.01	4.28	5.15	0.48
4	7.39484	0.07	13.22	0.07	34.78	0.07	2469.52	0.00	10.74	0.06	4.29	0.58	0.49
5	5.89769	0.00	17.44	0.00	38.63	0.00	3315.48	0.03	13.52	0.02	4.28	3.13	0.57
10	3.38780	2.13	55.33	0.00	60.42	0.00	8010.27	0.31	22.60	0.43	4.35	2.80	0.72
15	2.46689*	0.00	106.64	0.00	81.69	0.87	12,961.38	0.26	31.35	0.46	4.28	4.49	0.57
20	2.03496	0.39	178.25	0.05	114.14	0.00	18,125.08	0.29	42.45	0.60	4.30	2.17	0.70
25	1.73617	0.44	274.78	0.00	154.13	0.03	23,422.95	0.37	54.60	0.59	4.31	3.75	0.70
E_{aver}		0.38		0.01		0.12		0.16		0.27		2.80	
t_{init}			6.42		6.22		6.22		6.26		1.87		1.97
t_{total}			281.20		160.34		23,429.17		60.86		36.26		6.58

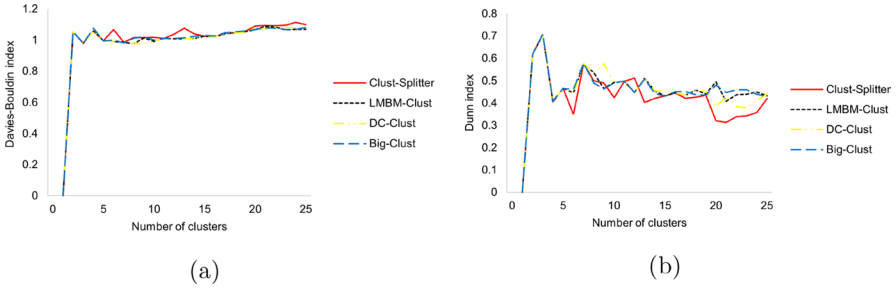


Fig. 21 Covertype: both (a) Davies–Bouldin and (b) Dunn validity indices versus number of clusters.

Acknowledgements The work was financially supported by the Research Council of Finland (projects no. #345804 and #345805 led by Prof. Tapio Pahikkala and Prof. Antti Airola, respectively), and Jenny and Antti Wihuri Foundation.

Funding Open Access funding provided by University of Turku (including Turku University Central Hospital).

Data Availability The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Code Availability The source code of the new method is available from https://github.com/jmlamp/Clust-Splitter.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animal performed by any of the authors.

Informed consent For this type of study formal consent is not required.

References

Abdo A, Abdelkader O, Abdel-Hamid L (2024) SA-PSO-GK++: a new hybrid clustering approach for analyzing medical data. IEEE Access 12:12501–12516

Al-Sultan KS (1995) A tabu search approach to the clustering problem. Pattern Recognit 28(9):1443–1451

Alotaibi Y (2022) A new meta-heuristics data clustering algorithm based on tabu search and adaptive search memory. Symmetry 14(3):623

Anagnostopoulos C (2018) Query analytics workloads dataset. UCI Machine Learning Repository. https://doi.org/10.24432/C50W4C
Bagirov AM (2008) Modified global k-means algorithm for sum-of-squares clustering problems. Pattern Recognit 41(10):3192–3199
Bagirov AM (2014) An incremental DC algorithm for the minimum sum-of-squares clustering. Iran J Oper Res 5(1):1–14
Bagirov AM, Mohebi E (2015) Nonsmooth optimization based algorithms in cluster analysis. In: Celebi E (ed) Partit Cluster Algorithms. Springer, Cham, pp 99–146
Bagirov AM, Ugon J (2005) An algorithm for minimizing clustering functions. Optimization 54(4–5):351–368
Bagirov AM, Yearwood J (2006) A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. Eur J Oper Res 170(2):578–596
Bagirov AM, Ugon J, Webb D (2011) Fast modified global k-means algorithm for sum-of-squares clustering problems. Pattern Recognit 44:866–876
Bagirov AM, Karmitsa N, Mäkelä MM (2014) Introduction to nonsmooth optimization: theory, practice and software. Springer, Cham
Bagirov AM, Ordin B, Ozturk G et al (2015) An incremental clustering algorithm based on hyperbolic smoothing. Comput Optim Appl 61(1):219–241
Bagirov AM, Taheri S, Ugon J (2016) Nonsmooth DC programming approach to the minimum sum-of-squares clustering problems. Pattern Recognit 53:12–24
Bagirov AM, Aliguliyev RM, Sultanova N (2023) Finding compact and well-separated clusters: clustering using silhouette coefficients. Pattern Recognit 135:109144
Bagirov AM, Karmitsa N, Taheri S (2025) Partitional clustering via nonsmooth optimization: clustering via optimization, 2nd edn. Springer, Cham
Bator M (2013) Dataset for sensorless drive diagnosis. UCI Mach Learn Reposit. https://doi.org/10.24432/C5VP5F
Bhatt R, Dhall A (2009) Skin segmentation. UCI Mach Learn Reposit. https://doi.org/10.24432/C5T30C
Bixby B, Reinel G (1995) TSPLIB—a library of travelling salesman and related problem instance. http://softlib.rice.edu/tsplib.html (September 27th, 2025)
Blackard J (1998) Covertype. UCI Mach Learn Reposit. https://doi.org/10.24432/C50K5N
Cole R, Fanty M (1991) ISOLET. UCI Mach Learn Reposit. https://doi.org/10.24432/C51G69
Cuong TH, Yao JC, Yen ND (2019) On some incremental algorithms for the minimum sum-of-squares clustering problem. Part 1: Ordin and Bagirov's incremental algorithm. arXiv preprint arXiv:1901.10151
Cura T (2012) A particle swarm optimization approach to clustering. Expert Syst Appl 39(1):1582–1588
Davies DL, Bouldin DW (1979) A cluster separation measure. IEEE Trans Pattern Anal Mach Intell PAMI-1(2):224–227
De Dominicis C (2020) MFCCs for speech emotion recognition. https://www.kaggle.com/datasets/cracc97/features (September 27th, 2025)
Defferrard M, Benzi K, Vandergheynst P, et al (2017) FMA: a dataset for music analysis. UCI Mach Learn Reposit. https://doi.org/10.24432/C5HW28
Dunn JC (1974) Well-separated clusters and optimal fuzzy partitions. J Cybern 4(1):95–104
Fernandes K, Vinagre P, Cortez P, et al (2015) Online news popularity. UCI Mach Learn Reposit. https://doi.org/10.24432/C5NS3V
Gribel D, Vidal T (2019) HG-means: a scalable hybrid genetic algorithm for minimum sum-of-squares clustering. Pattern Recognit 88:569–583
Guyon I, Gunn S, Ben-Hur A, et al (2004) Gisette. UCI Mach Learn Reposit. https://doi.org/10.24432/C5HP5B
Haarala M (2004) Large-scale nonsmooth optimization: variable metric bundle method with limited memory. Ph.D. thesis, University of Jyväskylä, Department of Mathematical Information Technology
Haarala M, Miettinen K, Mäkelä MM (2004) New limited memory bundle method for large-scale nonsmooth optimization. Optim Methods Softw 19(6):673–692
Haarala N, Miettinen K, Mäkelä MM (2007) Globally convergent limited memory bundle method for large-scale nonsmooth optimization. Math Program 109(1):181–205
Halkola AS, Joki K, Mirtti T et al (2023) OSCAR: optimal subset cardinality regression using the L0-pseudonorm with applications to prognostic modelling of prostate cancer. PLoS Comput Biol 19(3):1–31. https://doi.org/10.1371/journal.pcbi.1010333
Hubert L, Arabie P (1985) Comparing partitions. J Classif 2:193–218

Karim MR, Beyan O, Zappa A et al (2021) Deep learning-based clustering approaches for bioinformatics. Brief Bioinform 22(1):393–415
Karmitsa N, Bagirov AM, Taheri S (2017) New diagonal bundle method for clustering problems in large data sets. Eur J Oper Res 263(2):367–379
Karmitsa N, Bagirov AM, Taheri S (2018) Clustering in large data sets with the limited memory bundle method. Pattern Recognit 83:245–259
Karmitsa N, Taheri S, Bagirov AM et al (2022) Missing value imputation via clusterwise linear regression. IEEE Trans Knowl Data Eng 34(4):1889–1901
Karmitsa N, Taheri S, Joki K et al (2023) Nonsmooth optimization-based hyperparameter-free neural networks for large-scale regression. Algorithms 16(9):444
Karmitsa N, Eronen VP, Mäkelä MM et al (2025) Stochastic limited memory bundle algorithm for clustering in big data. Pattern Recognit 165:111654
Kaul M (2013) 3D Road Network (North Jutland, Denmark). UCI Mach Learn Reposit. https://doi.org/10.24432/C5GP51
KDD Cup 2004 (2004) Protein homology dataset. https://www.kdd.org/kdd-cup/view/kdd-cup-2004/Data (September 27th, 2025)
Khalaf W, Astorino A, D'Alessandro P et al (2017) A DC optimization-based clustering technique for edge detection. Optim Lett 11:627–640. https://doi.org/10.1007/s11590-016-1031-7
Kim W, Kanezaki A, Tanaka M (2020) Unsupervised learning of image segmentation based on differentiable feature clustering. IEEE Trans Image Process 29:8055–8068
Lai JZ, Huang TJ (2010) Fast global k-means clustering using cluster membership and inequality. Pattern Recognit 43(5):1954–1963
Le Thi HA, Pham TD (2009) Minimum sum-of-squares clustering by DC programming and DCA. Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence. Springer, Berlin, Heidelberg, pp 327–340
Le Thi HA, Belghiti MT, Pham TD (2007) A new efficient algorithm based on DC programming and DCA for clustering. J Glob Optim 37(4):593–608
Le Thi HA, Le HM, Pham TD (2014) New and efficient DCA based algorithms for minimum sum-of-squares clustering. Pattern Recognit 47(1):388–401
Likas A, Vlassis M, Verbeek J (2003) The global k-means clustering algorithm. Pattern Recognit 36(2):451–461
Mansueto P, Schoen F (2021) Memetic differential evolution methods for clustering problems. Pattern Recognit 114:107849
Markelle K, Longjohn R, Nottingham K (2025) The UCI machine learning repository. https://archive.ics.uci.edu (September 27th, 2025)
McQueen JB (1967) Some methods of classification and analysis of multivariate observations. In: Proceedings of 5th Berkeley symposium on mathematical statistics and probability. University of California Press, pp 281–297
Mussabayev R, Mladenovic N, Jarboui B et al (2023) How to use k-means for big data clustering? Pattern Recognit 137:109269
Naeem M, Asghar S (2011) KEGG metabolic relation network (directed). UCI Mach Learn Reposit. https://doi.org/10.24432/C5CK52
Ordin B, Bagirov AM (2015) A heuristic algorithm for solving the minimum sum-of-squares clustering problems. J Glob Optim 61(2):341–361
Paasivirta P, Numminen R, Airola A, et al (2024) Predicting pairwise interaction affinities with L0-penalized least squares—a nonsmooth bi-objective optimization based approach. Optim Methods Softw In Press, pp 1–28. https://doi.org/10.1080/10556788.2023.2280784
Pedregosa F, Varoquaux G, Gramfort A et al (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
Rahman MA, Islam MZ (2014) A hybrid clustering technique combining a novel genetic algorithm with k-means. Knowl-Based Syst 71:345–365
Riddle-Workman E, Evangelou M, Adams NM (2021) Multi-type relational clustering for enterprise cyber-security networks. Pattern Recognit Lett 149:172–178
Roe B (2005) MiniBooNE particle identification. UCI Mach Learn Reposit. https://doi.org/10.24432/C5QC87
Roesler O (2013) EEG eye state. UCI Mach Learn Reposit. https://doi.org/10.24432/C57G7J
Sanjak J, Binder J, Yadaw AS et al (2024) Clustering rare diseases within an ontology-enriched knowledge graph. J Am Med Inform Assoc 31(1):154–164

Seifollahi S, Bagirov AM, Borzeshi EZ et al (2019) A simulated annealing-based maximum-margin clustering algorithm. Comput Intell 35(1):23–41

Selim SZ, Al-Sultan KS (1991) A simulated annealing algorithm for the clustering. Pattern Recognit 24(10):1003–1008

Taheri S, Bagirov AM, Gondal I et al (2020) Cyberattack triage using incremental clustering for intrusion detection systems. Int J Inf Secur 19:597–607

Vergara A (2012) Gas sensor array drift dataset. UCI Mach Learn Reposit. https://doi.org/10.24432/C5RP6W

Xavier VL, Xavier AE (2020) Accelerated hyperbolic smoothing method for solving the multisource Fermat-Weber and k-Median problems. Knowl-Based Syst 191:105226

Xie J, Jiang S, Xie W et al (2011) An efficient global k-means clustering algorithm. J Comput 6(2):271–279

Xu KS, Kliger M, Hero AO III (2014) Adaptive evolutionary clustering. Data Min Knowl Discov 28:304–336