The VLDB Journal (2022) 31:711–732
https://doi.org/10.1007/s00778-021-00716-y

REGULAR PAPER

Efficient exploratory clustering analyses in large-scale exploration processes

Manuel Fritz[1] · Michael Behringer[1] · Dennis Tschechlov[1] · Holger Schwarz[1]

Received: 17 March 2021 / Revised: 31 August 2021 / Accepted: 15 October 2021 / Published online: 29 November 2021

Abstract
Clustering is a fundamental primitive in manifold applications. In order to achieve valuable results in exploratory clustering analyses, parameters of the clustering algorithm have to be set appropriately, which is a tremendous pitfall. We observe multiple challenges for large-scale exploration processes. On the one hand, they require specific methods to efficiently explore large parameter search spaces. On the other hand, they often exhibit large runtimes, in particular when large datasets are analyzed using clustering algorithms with super-polynomial runtimes, which repeatedly need to be executed within exploratory clustering analyses. We address these challenges as follows: First, we present LOG-Means and show that it provides estimates for the number of clusters in sublinear time regarding the defined search space, i.e., provably requiring less executions of a clustering algorithm than existing methods. Second, we demonstrate how to exploit fundamental characteristics of exploratory clustering analyses in order to significantly accelerate the (repetitive) execution of clustering algorithms on large datasets. Third, we show how these challenges can be tackled at the same time. To the best of our knowledge, this is the first work which simultaneously addresses the above-mentioned challenges. In our comprehensive evaluation, we unveil that our proposed methods significantly outperform state-of-the-art methods, thus especially supporting novice analysts for exploratory clustering analyses in large-scale exploration processes.

Keywords Exploratory clustering analysis · Exploration · Clustering · Centroid-based clustering

1 Introduction

Clustering is a fundamental primitive for exploratory tasks. Manifold application domains rely on clustering techniques: In computer vision, image segmentation tasks can be formulated as a clustering problem [21,48]. Documents may be clustered to support faster information access and retrieval [10,36]. For business purposes, clustering may be used for grouping customers, for workforce management or for planning tasks [32,45]. In biology, clustering is applied to study genome data amongst others [9].

✉ Manuel Fritz
manuel.fritz@ipvs.uni-stuttgart.de

Michael Behringer
michael.behringer@ipvs.uni-stuttgart.de

Dennis Tschechlov
dennis.tschechlov@ipvs.uni-stuttgart.de

Holger Schwarz
holger.schwarz@ipvs.uni-stuttgart.de

1 University of Stuttgart, Stuttgart, Germany

Jain identified three main general purposes of clustering throughout these and many more application domains, which emphasize the exploratory power of clustering analyses [37]: (i) Assessing the structure of the data. Here, the goal is to exploit clustering to gain a better understanding of data, to generate hypotheses, or to detect anomalies. (ii) Grouping entities. Clustering aims to group similar entities into the same cluster. Thus, previously unseen entities can be assigned to a specific cluster. (iii) Compressing data, i.e., to use the clusters and their information as summary of the data for further steps.

Due to their apparently linear runtime behavior, centroid-based clustering algorithms [13], such as k-Means [40,41], k-Medians [11,38], or k-Mode [35] are commonly used [53]. However, the expected number of clusters k has to be provided prior to the execution of these algorithms. Especially for arbitrary, previously unknown datasets, estimating this number is a tremendous pitfall and requires particular caution. Wrong values for k lead to questionable results regarding the above-mentioned purposes, i.e., wrong struc-

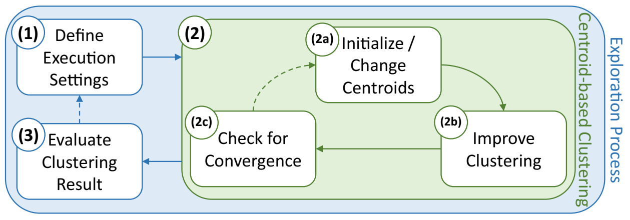

(1) Define Execution Settings
(2)
(2a) Initialize / Change Centroids
(2b) Improve Clustering
(2c) Check for Convergence
(3) Evaluate Clustering Result
Centroid-based Clustering
Exploration Process

Fig. 1 Exploratory clustering analysis for valuable clustering results. Within an exploration iteration, the centroid-based clustering algorithm is executed

turings, groupings or compressions are performed, making the clustering results unusable in the worst case.

1.1 Problem statement

Several methods have been proposed to estimate the number of clusters in arbitrary datasets [12,15–17,30,44,46,49,51,52]. These methods rely on a search space $\mathcal{R}$, which is typically defined by analysts and is thus highly influenced by their domain knowledge. Hence, we have to distinguish between experienced and novice analysts: Experienced analysts may use their strong domain knowledge to reduce $\mathcal{R}$ to a manageable size. In contrast, novice analysts typically lack in-depth domain knowledge and therefore often define larger search spaces, because of the underlying uncertainty.

Figure 1 shows the typical procedure of exploratory clustering analyses as they are performed by commonly used estimation methods. Exploratory clustering analyses comprises an exploration process (depicted in blue), which performs several exploration iterations. Each iteration includes steps to (1) define execution settings, i.e., choosing the parameter value k within the given search space $\mathcal{R}$, (2) execute the centroid-based clustering algorithm (depicted in green), and (3) evaluate the clustering result. Depending on the estimation method, several execution settings may be defined, i.e., the centroid-based clustering algorithm is typically executed multiple times with varying parameter values $k \in \mathcal{R}$. The centroid-based clustering algorithms used in step (2) perform several clustering iterations, where each of these iterations consists of three steps: (2a) k centroids are initialized. These centroids are centers of gravity for a certain cluster, such as the mean value of the cluster for k-Means. (2b) In the “improve clustering” step, entities from the dataset are assigned to the closest centroid. (2c) Check for convergence, i.e., check if an entity changed its membership to another cluster. If not, the algorithm converges. However, if an entity changed its membership, another clustering iteration is performed. Hence, in step (2a), the centroids are moved to the center of each corresponding cluster.

Existing estimation methods commonly use two strategies to explore the search space: (i) An exhaustive search in $\mathcal{R}$ is often conducted. A well-known estimation method following this search strategy is the elbow method [50]. (ii) $\mathcal{R}$ can be explored in a non-exhaustive manner, i.e., stopping the search as soon as the clustering results of subsequent values for $k \in \mathcal{R}$ differ only marginally. A common method following this approach is X-Means [44].

We observe three main challenges regarding exploratory clustering analyses:

Challenge 1 (C1) The runtime complexity of the exploration process (depicted in blue in Fig. 1) lies in $\mathcal{O}(|\mathcal{R}|)$ for a search space $\mathcal{R}$. That is, in the worst case, a clustering algorithm needs to be executed for each value of $k \in \mathcal{R}$. Note, that this worst-case scenario also holds true for non-exhaustive estimation methods. Hence, existing estimation methods do in particular not address an efficient exploration in large search spaces.

Challenge 2 (C2) Regarding the repetitive execution of the centroid-based clustering algorithm (depicted in green in Fig. 1), we face the following problem: Typically, k-Means is used as an instantiation of a centroid-based clustering algorithm due to its simplicity and apparently appealing runtime behavior in contrast to other clustering algorithms, which have higher runtime complexities per se. Yet, as an in-depth analysis of the k-Means algorithm unveils, it has a super-polynomial runtime in the input size in the worst case [4]. Hence, each single execution of a clustering algorithm may become very time-consuming on large datasets.

Challenge 3 (C3) Related work proposes novel methods to address either C1 or C2, yet they ignore their interdependencies. However, these interdependencies are of high relevance for exploratory clustering analyses, since novice analysts typically define large parameter search spaces while using large datasets. This results in large-scale exploration processes. Hence, there is currently no feasible support for novice analysts in order to achieve valuable clustering results in large-scale exploration processes in a reasonable time.

1.2 Our contributions

In this work, we address the aforementioned challenges. Regarding C1, we present LOG-Means, which is able to overcome the pitfalls of existing estimation methods. Similarly to existing methods, LOG-Means draws on individual clustering results from $\mathcal{R}$, yet aims for significantly fewer executions of a clustering algorithm until an estimation can be made. Thus, it is of particular interest for large search spaces, as they might be defined by novice analysts. To address C2, we exploit specific characteristics of the procedure of exploratory clustering analyses (cf. Fig. 1) in order to accelerate the repetitive executions of a centroid-based clustering algorithm. Finally, we address C3 by unveiling how existing estimation methods, as well as LOG-Means, can be significantly accelerated, thus supporting novice analysts in large-scale exploration processes. To the best of our knowledge, there is no prior work, which (i) exploits characteristics of exploration processes in order to accelerate estimation methods, (ii) systematically analyzes these interdependencies with several commonly used estimation methods, and (iii) thus unveils strong benefits for large-scale exploration processes.

The contributions of this paper are partially based on our previous works to address C1 [23] and C2 [24,25] independent from each other. We significantly extend our prior work by presenting additional contributions related to C3, i.e., exploratory clustering analyses in large-scale exploration processes. Overall, the contributions are as follows:

– We describe our estimation method LOG-Means and discuss it in comparison to existing estimation methods.
– We analyze LOG-Means and demonstrate that it provides better estimates and scales sublinearly with the search space, thus being a strong fit for large search spaces.
– We present Delta Initialization, which uses results of a previous execution of a centroid-based clustering algorithm to enhance subsequent executions.
– We summarize our Generic Qualitative Approximation to reduce the clustering iterations, while still achieving high qualitative clustering results.
– We show how Delta Initialization and Generic Qualitative Approximation can seamlessly be integrated into the generic procedure of exploratory clustering analyses, e.g., into existing estimation methods.
– In our evaluation on an Apache Spark cluster, we unveil that several estimation methods highly benefit from Delta Initialization and Generic Qualitative Approximation, i.e., speedups of 6$\times$ were achieved, while also achieving more accurate estimates.
– Finally, we show that LOG-Means and Generic Qualitative Approximation significantly outperform state-of-the-art approaches for exploratory clustering analyses, as speedups of more than 34$\times$ (more than 7 hours) are achieved, while simultaneously achieving up to 50$\times$ more accurate estimates. Hence, especially novice analysts highly benefit when exploring large-scale exploration processes. To the best of our knowledge, this is the first systematic evaluation addressing C3 in detail.

1.3 Structure of this work

The remainder of this paper is structured as follows: In Sect. 2, we present related work, i.e., existing estimation methods and methods to accelerate centroid-based clustering algorithms. We reveal the procedure of LOG-Means in Sect. 3 and analyze it in detail. In Sect. 4, we address the repetitive execution of a centroid-based clustering algorithm and show how Delta Initialization and Generic Qualitative Approximation can seamlessly be integrated into the generic procedure of exploratory clustering analyses to accelerate these executions. In Sect. 5, we discuss the results of our experimental evaluation unveiling the benefits of LOG-Means and our prior work regarding efficiently performing exploratory clustering analyses for valuable clustering results. Finally, we conclude this work in Sect. 6.

2 Related work

As shown in Fig. 1, exploratory clustering analysis relies on a repetitive execution of a clustering algorithm with varying execution settings. Since there is currently no work addressing C3, we divide related work into two categories, according to challenges C1 and C2: First, we investigate fundamental characteristics of estimation methods for the number of clusters, since they perform an exploratory clustering analysis in a (semi-)automated manner. Second, we examine how clustering algorithms can be accelerated, since they are at the core of exploratory clustering analyses.

2.1 Estimation methods

Estimation methods require a prior definition of the search space $\mathcal{R}$ for the expected number of clusters. $\mathcal{R}$ is a discrete range of values for $k \in \mathbb{N}$ where the actual number of clusters is expected to be in. Since clustering groups similar entities together, this search space is in the worst case $\mathcal{R} = [2; n-1]$, where n is the number of entities in the dataset. An experienced analyst may be able to significantly reduce the search space based on prior domain knowledge. However, especially for novice analysts, advanced estimation methods are of paramount interest to efficiently estimate the number of clusters. Such estimation methods follow a common procedure of three steps, which we dub exploration iteration (cf. Fig. 1, depicted in blue): (1) Define execution settings, i.e., which parameter in $\mathcal{R}$ to execute on which subset of the data, (2) execute the clustering algorithm with the defined execution settings, and subsequently (3) evaluate the clustering result.

In general, estimation methods can be divided into different categories: They are either exhaustive, meaning they perform an exhaustive search and execute the clustering algorithm for each $k \in \mathcal{R}$, or they are non-exhaustive in the sense that they do not perform an exhaustive search. Moreover, these methods either work in an automated or in a semi-automatic manner, i.e., with user interaction. In the following, we briefly present related work for these categories. For a more detailed discussion of estimation methods, we refer to our previous paper [23].

Exhaustive estimation methods These methods execute a clustering algorithm for each $k \in \mathcal{R}$ and subsequently evaluate each result, e.g., according to a clustering validity measure

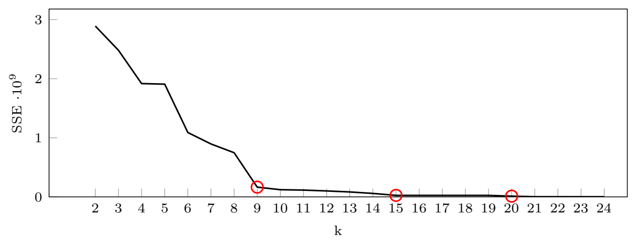


Fig. 2 Elbow graph. Red circles depict possible bends that may be selected by analysts [23]

[12,15–17,46,49]. Finally, the best result is selected as an estimation for k. Estimation methods in this category mainly differ in the validity measures they use to evaluate the quality of a single clustering result, e.g., they use clustering validity measures or approaches from information theory [22].

Non-exhaustive estimation methods Non-exhaustive methods perform an ascending search in $\mathcal{R}$ and stop as soon as subsequent clustering results differ only marginally according to a certain evaluation criterion. Typical criteria used by non-exhaustive estimation methods are clustering validity measures [51], criteria from information theory [44], or criteria addressing the data distribution within the resulting clusters [30].

Semi-automatic methods While all the estimation methods presented above are purely automated, there is also the group of semi-automated methods. Undoubtedly, the elbow method [50] is the most commonly used method of this group. The elbow method was first discussed by Thorndike [50]. It comprises four steps: (i) Execute a centroid-based clustering algorithm, e.g., k-Means, for each $k \in \mathcal{R}$, (ii) calculate the variance of each clustering result, e.g., sum of squared errors (SSE) for k-Means, (iii) plot the results in a graph, and (iv) select the bend in the graph. As the first two steps are straightforward and do not differ from existing exhaustive estimation methods, we focus on the latter two steps. Figure 2 shows an example of an elbow graph as it is created in the third step.

Here, $k \in \mathcal{R}$ is depicted on the x-axis and the corresponding SSE values are shown on the y-axis. The intuition of this graph is to visually show after which $k \in \mathcal{R}$ the reduction of the SSE becomes negligible with an increasing value for k. This point can be seen as a "bend" in the graph, similarly to the bend of the human's elbow. That's why practitioners coined the name elbow method for this estimation method. It is the task of an analyst to select this bend in the fourth step, which is however often very ambiguous (possible bends depicted in red in Fig. 2). By selecting a bend, the analyst prevents an overfitting of the clustering to the data in terms of a too high value for k, where no crucial additional benefits are achieved. In conclusion, the elbow method consists of automated parts (steps i - iii) and parts that require human interaction (step iv) in order to estimate the number of clusters in a dataset.

As this overview shows, existing estimation methods rely on the excessive execution of a clustering algorithm. Exhaustive methods execute such an algorithm for each element in $\mathcal{R}$. Even non-exhaustive methods typically draw on an excessive execution of a clustering algorithm until they stop. Furthermore, there is no theoretical guarantee that they stop at all, i.e., in the worst case, they perform an exhaustive search. Hence, these approaches are not feasible for large parameter search spaces, since they are costly to perform. In Sect. 3, we present our novel estimation method LOG-Means, which addresses this problem of exploring large search spaces by drawing on a more elaborated search strategy.

2.2 Accelerating clustering algorithms

Since centroid-based clustering algorithms are at the core of the above-mentioned estimation methods, we focus on their procedure in the following. These algorithms proceed in an iterative manner, which means that the same sequence of steps is repeated until a given convergence criterion is met. As shown in step (2) in Fig. 1 (depicted in green), each clustering iteration comprises the following three steps: (2a) initialize or change the position of the centroids, which are centers of gravity for a specific cluster, (2b) improve the clustering by (re-)assigning entities to the closest centroid, and (2c) check for convergence. Many centroid-based clustering algorithms terminate when no entities change their membership anymore.

Typically, k-Means is used as instantiation of a centroid-based clustering algorithm across researchers and practitioners due to its runtime behavior and its simplicity [53]. The runtime complexity of k-Means is $\mathcal{O}(nd\omega k)$ [29], where n denotes the number of entities, d is the number of dimensions, ω is the number of clustering iterations performed, and k denotes the expected number of clusters. Next, we will discuss these four influencing factors for exploratory clustering analyses.

In order to reduce the number of entities n in a dataset, sampling or coresets [6,7] can be used to ultimately reduce the runtime of a clustering algorithm. Similar observations apply to (i) dimensionality reduction techniques, e.g., PCA or SVD, (ii) embeddings, or (iii) sketches, which all together aim to reduce the number of dimensions d of a dataset [1].

While n and d are dependent on dataset characteristics, a prior work from Arthur and Vassilvitskii unveiled, that k-Means requires in the worst case a super-polynomial number of clustering iterations until convergence, i.e., $\omega = 2^{\Omega(\sqrt{n})}$ [4]. Hence, addressing the number of clustering iterations ω is crucial in order to accelerate clustering algorithms, especially on large datasets.

There are three aspects of related work, which address the internals of a clustering iteration: (a) It has been shown that the initialization of centroid-based clustering algorithms is crucial to reduce the number of clustering iterations [5,8]. The initial centroids of state-of-the-art initialization techniques are close to their optimum position, therefore requiring less clustering iterations until convergence than seeding these initial centroids at random. However, these approaches typically require more time for the initialization step as rather simple approaches. (b) Several works address how a single clustering iteration can be accelerated, e.g., by making distance calculations faster [18,39] or by caching previously calculated distances [28]. This mostly addresses the "improve clustering" step. (c) Undoubtedly, reducing ω is crucial since it subsumes approaches from (a) and (b). Therefore, we argue that the "check for convergence" step is of paramount interest when aiming to reduce ω to a reasonable size and thereby reducing the runtime of the clustering algorithm. Note however, that a combination with the other improvements is possible to achieve additional speedups.

As each clustering iteration comprises many distance calculations, it is not feasible to perform a clustering algorithm on large datasets until convergence due to the excessive runtime. Given the importance of the number of clustering iterations, the question arises when to terminate the algorithm earlier than convergence. An easy approach to reduce the runtime of the clustering algorithm is to allow a fixed number of clustering iterations. However, it is challenging to choose a promising value for this threshold: Too few iterations lead to an imprecise result, whereas too many iterations lead to a long runtime. A generic threshold for all datasets is not feasible, because of too many influencing factors, such as the feature space or data distribution.

Mexicano et al. provide a more generic approach by focusing on the displacement of the centroids after each iteration [43]. They assume that the maximal centroid displacement happens in the first iteration. Hence, they propose to stop the clustering algorithm once the centroid displacement is less than 5 % of the maximum centroid displacement. Thereby, they neglect to explicitly set a maximum number of iterations. However, they did not address why the coefficient is set to 5 % and how their approach correlates to the final clustering quality.

While all of these works consider the acceleration of a single execution of a centroid-based clustering algorithm, there is only little work regarding the acceleration of several executions with different parameter values, such as it is common in exploratory clustering analyses. These few works propose an improvement regarding two aspects [27,54]: Firstly, they address the "initialize centroids" step by reusing centroids and adding randomly selected centroids, such that k centroids are achieved in total for the next exploration iteration. While this seems like a straightforward approach, several works have proven - theoretically and empirically - that a completely random-based initialization leads to worse results than more sophisticated initialization techniques [5,8,14,20]. Secondly, they exploit the triangle inequality in the "improve clustering" step to reduce the number of distance calculations. While this is a promising approach as several previous works have already shown [18,28,31], we argue that this improvement can be combined with our methods, which address the remaining steps outlined in Fig. 1. Hence, even faster exploratory clustering analyses can be conducted.

Summarizing existing approaches to accelerate clustering algorithms, they mostly address the downsizing of the dataset, as well as making clustering iterations faster. For exploratory clustering analyses, there are only very few works addressing specific steps and unveil only moderate accelerations between 5× to 9× [27,54]. In Sect. 4, we present our approaches, which accelerate clustering algorithms tremendously by exploiting fundamental characteristics of exploration processes.

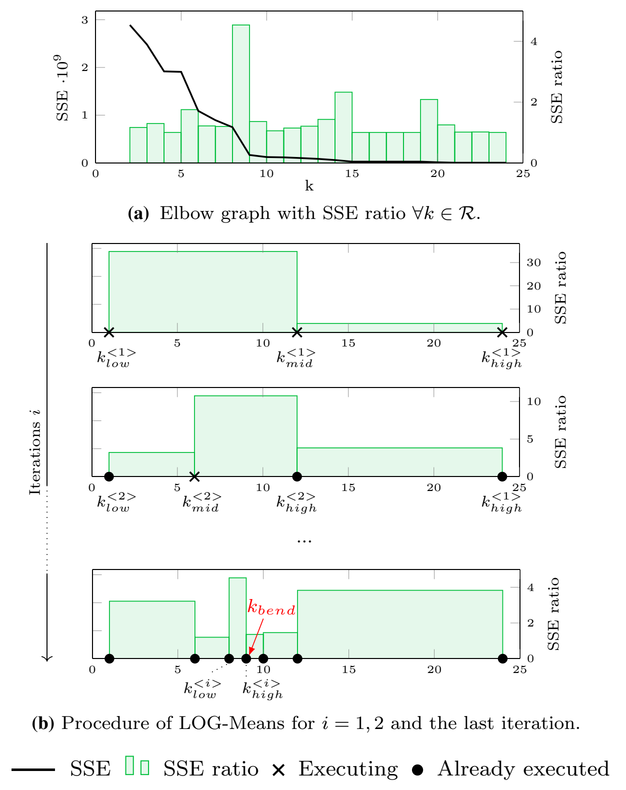
3 LOG-Means

In this section, we present LOG-Means, our novel estimation method for the number of clusters, which particularly addresses challenge C1. In contrast to existing estimation methods (cf. Sect. 2.1), LOG-Means proceeds in a greedy manner, thereby efficiently reducing the number of exploration iterations and thus the number of executions of centroid-based clustering algorithms. To this end, it exploits fundamental properties of the elbow method, since the elbow method is commonly used by researchers and practitioners from several domains. However, as the elbow method has severe drawbacks, such as the manual selection of the bend, and suffers thereby from the ambiguity of the term "bend" (cf. Fig. 2), LOG-Means aims to overcome these problems.

Before detailing on LOG-Means, we briefly summarize the basics of centroid-based clustering algorithms. Let $\mathcal{X}$ be a dataset with n entities and d dimensions, i.e., $\mathcal{X} \subset \mathbb{R}^d$. The goal of centroid-based clustering algorithms is to group $\mathcal{X}$ into k disjoint clusters, such that each entity is assigned to the closest centroid $c \in \mathcal{C}$. As this problem is NP-hard [3,26], several heuristics exist which aim to approximate the solution. One of these heuristics is the k-Means algorithm [40,41]. The goal of k-Means is to find the set $\mathcal{C}$ of k centroids which minimizes the objective function in Equation 1.

$$\phi_{\mathcal{X}}(\mathcal{C}) = \sum_{x\in\mathcal{X}} \min_{c\in\mathcal{C}} \|x - c\|^2 \tag{1}$$

Here, the Euclidean distance from an entity $x \in \mathcal{X}$ to the closest centroid $c \in \mathcal{C}$ is calculated. $\phi_{\mathcal{X}}(\mathcal{C})$ denotes the sum of these distances over all entities in $\mathcal{X}$. This sum is also called the sum of squared errors (SSE). Algorithms like k-Means move these k centroids to a better position in each clustering iteration, until their position converges, i.e., until



(a) Elbow graph with SSE ratio $\forall k \in \mathcal{R}$.

(b) Procedure of LOG-Means for $i = 1, 2$ and the last iteration.

Fig. 3 Relation between SSE and SSE ratios is shown. LOG-Means iteratively halves areas with the highest ratios [23]

no more changes occur. In order to measure the quality of a clustering result, the SSE can be used. This validity measure denotes the variance of the resulting clusters, i.e., the smaller the SSE, the more compact the clusters.

3.1 Intuition

The elbow graph provides valuable properties, which can be exploited by specialized search strategies. The intuition of LOG-Means relies in particular on two specific properties of this graph. Since these properties are valid independent of datasets and the size of search spaces, LOG-Means preserves generality by exploiting these properties.

Property 1 In general, the sum of squared errors (SSE) follows a decreasing trend with an increasing value for k. This can be shown based on the objective function in Equation 1, assuming that a global optimum can be found by the clustering algorithm. We proof this property by induction, where the base case is $|\mathcal{C}| = |\mathcal{X}|$, i.e., we cluster with as many clusters as entities in $\mathcal{X}$. The goal of centroid-based clustering algorithms is to assign entities $x \in \mathcal{X}$ to centroids $c \in \mathcal{C}$, which are closer to c_i than to any other c_j with $c_i \neq c_j \in \mathcal{C}$. Hence, for the base case of the induction, each entity x is its own centroid c, thus having no errors regarding the objective function, i.e., $\phi_{\mathcal{X}}(\mathcal{C}) = 0$. Proving the induction step is trivial based on Equation 1: We argue that if we remove one centroid from an arbitrary $\mathcal{C}'$ with $1 < |\mathcal{C}'| \leq |\mathcal{X}|$, the SSE increases, since the distance between entities and their closest centroids increases. Turning these observations the other way around, we conclude that the SSE decreases with an increasing value for k. A more detailed discussion, which also addresses possible local optima of the clustering algorithm, is deferred to Sect. 3.3.

Property 2 The bend describes a significant change in the elbow graph. Thorndike describes the bend as the sudden drop of the SSE between two adjacent values for k [50], yet left a clear definition for the terms "sudden" and "drop" open. Following his statement on the visual representation of the bend, we formalize the decrease of the SSE as $SSEratio_k = SSE_{k-1}/SSE_k$. This ratio can be exploited to investigate the SSE throughout $\mathcal{R}$. The *most significant* bend is denoted by $max(SSEratio_k)$. Figure 3a shows the SSE ratio for all $k \in \mathcal{R}$ for an exemplary search space $\mathcal{R}$. Here, the highest SSE ratio is between $k = 8$ and $k = 9$, i.e., the most significant bend is at $k = 9$.

Putting both properties together, we aim to avoid an exhaustive search by calculating the SSE ratio (cf. property 2) for areas of non-directly adjacent values of $k \in \mathcal{R}$. Due to the decreasing character of the elbow graph (cf. property 1), these areas provide a meaningful insight of the SSE ratio. Hence, the SSE ratio can be used to iteratively shrink these areas in a greedy manner and subsequently find the bend efficiently without an exhaustive search in $\mathcal{R}$.

The general idea of LOG-Means is depicted in Fig. 3b. For each iteration i, crosses show for which values $k \in \mathcal{R}$ a clustering algorithm is executed, whereas dots indicate where the SSE is available from previous iterations. After each execution of the clustering algorithm, the corresponding SSE value is calculated. Since the SSE ratio is a relative measure between two clustering results, it is updated whenever new adjacent SSE values for $k \in \mathcal{R}$ become available.

For $i = 1$, $k_{low}^{<1>} = min(\mathcal{R}) - 1$ and $k_{high}^{<1>} = max(\mathcal{R})$ and the clustering algorithm is executed for these two values. Note, that $min(\mathcal{R})$ must be reduced by one in order to keep the possibility to predict $min(\mathcal{R})$ as the value for k due to the definition of the SSE ratio. Next, the middle element $k_{mid}^{<1>}$ halves the area between $k_{low}^{<1>}$ and $k_{high}^{<1>}$. Subsequently, k-Means with $k = k_{mid}^{<1>}$ is executed and the SSE ratios of adjacent values of $k_{mid}^{<1>}$ are calculated, i.e., the ratios for $(k_{low}^{<1>}, k_{mid}^{<1>})$ and $(k_{mid}^{<1>}, k_{high}^{<1>})$. This allows to identify the next area with the highest SSE ratio. In Fig. 3b, the highest ratio is found in the area of $(k_{low}^{<1>}, k_{mid}^{<1>})$. Hence, for $i = 2$, the ratios $(k_{low}^{<2>}, k_{mid}^{<2>})$, $(k_{mid}^{<2>}, k_{high}^{<2>})$ and $(k_{high}^{<2>}, k_{high}^{<1>})$ are calculated, where the latter is equal to $(k_{mid}^{<1>}, k_{high}^{<1>})$ (note the different y-axes). Note, that previously calculated ratios are kept, if they are not adjacent to $k_{mid}^{<i>}$. Subsequently, the same procedure is iteratively applied to the area with the highest ratio.

The search stops as soon as the low and high elements are directly adjacent. In this area, the SSE ratio in the elbow graph is expected to be the highest. We denote the value for k with the highest SSE ratio of k_{low} and k_{high} of the last iteration as k_{bend}, since we expect the *bend* here.

It can be seen that no exhaustive search in the search space is conducted. However, the idea is to approach the area of the bend from the left- and the right-hand side of the elbow graph by following principles from logarithmic search. To this end, the search space is efficiently narrowed down around the highest SSE ratio and the clustering algorithm is executed with only few selected values from $\mathcal{R}$.

As our evaluation in the previous paper unveiled, an optional additional step may further increase the accuracy [23]. The reason for this is that k_{bend} may be only a local optimum. Yet, the global optimum, i.e., $max(SSERatio)$, is typically within a small ε environment. To this end, the ε environment around k_{bend} can be optionally evaluated in an exhaustive manner.

The general procedure of LOG-Means can also be applied to other centroid-based clustering algorithms, since they also minimize their specific notion of variance. Yet, we use k-Means, since it is the most commonly used algorithm of this family [53].

3.2 Algorithm

Algorithm 1 outlines the pseudo code for LOG-Means. As LOG-Means draws on executions of a centroid-based clustering algorithm, we assume k-Means as an instantiation thereof. Furthermore, we do not make any assumptions about its execution, i.e., improvements of k-Means (cf. Sect. 2.2) can be used in the corresponding steps. The algorithm of LOG-Means is separated into 5 parts:

In the first part, key-value data structures are defined (lines 2 and 3). These data structures keep track of already evaluated values within LOG-Means. $\mathcal{K}$ stores tuples of executed values for k and the corresponding SSE value. $\mathcal{M}$ stores tuples of k and the corresponding SSE ratio between k and the left adjacent value.

In the second part, i.e., from lines 4 to 7, k-Means is executed for k_{low} and k_{high}. The clustering results are evaluated according to the SSE and stored in $\mathcal{K}$.

The third part ranges from lines 8 to 20 and narrows down the search space around the estimated bend in the elbow graph. Here, the middle element is defined, k-Means is executed, the SSE is calculated and stored in $\mathcal{K}$ (lines 9-11). Subsequently, the SSE ratios are calculated in lines 12 and 13. These calculated SSE ratios are stored in $\mathcal{M}$ (lines 14 and 15). Since the area with the highest SSE ratio is halved in each iteration, the corresponding values in $\mathcal{M}$ are either

Algorithm 1: LOG-Means [23]

```
Input: $\mathcal{X}$ - dataset, $k_{low}$ - minimum number of desired clusters, $k_{high}$ -
       maximum number of desired clusters, $\varepsilon$ - number of neighbors to
       evaluate
Output: $k_{est}$ - estimated number of clusters for $\mathcal{X}$
1  $k_{low} \leftarrow k_{low} - 1$;
2  $\mathcal{K} \leftarrow \emptyset$;
3  $\mathcal{M} \leftarrow \emptyset$;
4  $SSE_{low} \leftarrow$ SSE from $k$-Means with $k_{low}$;
5  $\mathcal{K} \leftarrow \mathcal{K} \cup \{(k_{low}, SSE_{low})\}$;
6  $SSE_{high} \leftarrow$ SSE from $k$-Means with $k_{high}$;
7  $\mathcal{K} \leftarrow \mathcal{K} \cup \{(k_{high}, SSE_{high})\}$;
8  while ( $k_{low}$ and $k_{high}$ are not directly adjacent ) {
9      $k_{mid} \leftarrow \lfloor (k_{high} + k_{low})/2 \rfloor$;
10     $SSE_{mid} \leftarrow$ SSE from $k$-Means with $k_{mid}$;
11     $\mathcal{K} \leftarrow \mathcal{K} \cup \{(k_{mid}, SSE_{mid})\}$;
12     $ratio_{left} \leftarrow SSE_{low}/SSE_{mid}$;
13     $ratio_{right} \leftarrow SSE_{mid}/SSE_{high}$;
14     $\mathcal{M} \leftarrow$ store or update $\{(k_{mid}, ratio_{left})\}$;
15     $\mathcal{M} \leftarrow$ store or update $\{(k_{high}, ratio_{right})\}$;
16     $k_{high} \leftarrow k$ with highest ratio from $\mathcal{M}$;
17     $k_{low} \leftarrow$ left adjacent value of $k_{high}$ from $\mathcal{K}$;
18     $SSE_{high} \leftarrow$ SSE for $k_{high}$ from $\mathcal{K}$;
19     $SSE_{low} \leftarrow$ SSE for $k_{low}$ from $\mathcal{K}$;
20 }
21 if $\varepsilon > 0$ then
22     $k_{bend} \leftarrow k \in [k_{low}, k_{high}]$ with highest ratio in $\mathcal{M}$;
23     $k_{low} \leftarrow k_{bend} - \lfloor \varepsilon/2 \rfloor$;
24     $k_{high} \leftarrow k_{bend} + \lfloor \varepsilon/2 \rfloor$;
25     for ( $\forall k \in [k_{low}; k_{high}]$ ) {
26         $SSE_{k_{prev}} \leftarrow$ SSE of $k_{prev}$ from $\mathcal{K}$;
27         if $k \in \mathcal{K}$ then
28             $SSE_k \leftarrow$ SSE for $k$ from $\mathcal{K}$;
29         else
30             $SSE_k \leftarrow$ SSE from $k$-Means with $k$;
31             $\mathcal{K} \leftarrow \mathcal{K} \cup \{(k, SSE_k)\}$;
32         end
33         $ratio_k \leftarrow SSE_{k_{prev}}/SSE_k$;
34         $\mathcal{M} \leftarrow$ store or update $\{(k, ratio_k)\}$;
35     }
36 $k_{est} \leftarrow k \in [k_{low}, k_{high}]$ with highest ratio in $\mathcal{M}$;
37 return $k_{est}$;
```

stored or updated, if calculated previously. At the end of each iteration, new values for k_{low} and k_{high} are set in such a way that the area between these values localize the highest SSE ratio (lines 16 and 17). Subsequently, the SSE values for k_{low} and k_{high} are retrieved from $\mathcal{K}$ for the calculation of the SSE ratios in the next iteration (lines 18 and 19). The loop stops as soon as k_{low} and k_{high} are directly adjacent.

The fourth part ranges from lines 21 to 35 and is optional, if an $\varepsilon > 0$ environment is given. To this end, the highest SSE ratio between k_{low} and k_{high} is retrieved from $\mathcal{M}$ in line 22. As the bend is expected to be here, this point is called k_{bend}. Subsequently, the ε environment around k_{bend} is defined (lines 23 and 24). Within this ε environment, the SSE values are determined for each value of k. If available in $\mathcal{K}$, the corresponding SSE is retrieved, otherwise it will be calculated and stored in $\mathcal{K}$ (lines 27-32). Subsequently, the SSE ratio is calculated and stored in $\mathcal{M}$ (lines 33-34).

Finally, the algorithm provides an estimate in line 36 by selecting the value for k with the highest SSE ratio in $\mathcal{M}$. Note, that with $\varepsilon = 0$, this would be the same result as when ignoring the optional fourth step from lines 21 to 35.

3.3 Analysis

Property 1 states that the elbow graph follows a decreasing trend for increasing values of $k \in \mathcal{R}$. However, as k-Means is solely a heuristic to the NP-hard centroid-based clustering problem, the objective function in Equation 1 may comprise local optima, i.e., the centroids are not at the globally best position. Hence, the SSE does not necessarily decrease monotonously. Yet, it has been proven that enhanced initialization algorithms, such as k-Means++ [5] or k-Means$\|$ [8], provide an $\mathcal{O}(log\ k)$-approximation to the optimal clustering result w.r.t. the error (SSE) from the objective function in Equation 1 independent of dataset characteristics, thus supporting this property in most cases.

Despite these local optima, the SSE ratio (property 2) still provides crucial insights into how the elbow graph changes between two values for $k \in \mathcal{R}$. The $\mathcal{O}(log\ k)$-approximation becomes more noticeable the closer the values for $k \in \mathcal{R}$ are evaluated: For far-distant values for k, the approximation error has only slight impact on the SSE ratio, thus predicting large areas with high SSE ratios mostly correct. Hence, LOG-Means can efficiently narrow down large search spaces around areas with high SSE ratios.

For closer distant values for k, especially for direct neighbors of $k \in \mathcal{R}$, the SSE and therefore the SSE ratio can be more strongly influenced by local optima of the clustering algorithm. This can be seen for example in Fig. 3a at $k \approx 5$, where the SSE does not tend to decrease between two subsequent values for k, before it decreases rather strong at $k = 6$. Since LOG-Means keeps all areas and compares them in each iteration regarding their SSE ratio, we argue that using these "false" elbows for an estimation is rather unlikely.

Yet, since the areas become smaller in each iteration, LOG-Means becomes most sensitive towards the $\mathcal{O}(log\ k)$-approximation in the last iteration, resulting solely in a minor deviation from the optimum number of clusters. We argue and show in the evaluation in Sect. 5 that with the $\mathcal{O}(log\ k)$-approximation of state-of-the-art approaches of k-Means, LOG-Means is able to provide reasonable accurate estimates, yet no perfect estimates in every scenario. In our previous paper, we showed that by analyzing an optional ε environment around the expected bend, the effect of the local optima of the clustering algorithm can be further reduced [23].

Complexity analysis

As shown in Fig. 1, estimation methods proceed in three steps: (1) Defining execution settings, e.g., identifying which parameter to consider next, (2) executing k-Means with the determined parameter, and subsequently (3) evaluating the result. As discussed in Sect. 2, existing estimation methods typically perform an ascending or even worse an exhaustive search in the search space $\mathcal{R}$. Hence, the complexity of existing estimation methods lies in $\mathcal{O}(|\mathcal{R}|)$. Furthermore, the evaluation step can be costly due to a complex metric, which may lead to an even worse complexity class. Hence, these strategies require a huge overall runtime until an estimation can be made. On the contrary, LOG-Means promises a better runtime behavior regarding $\mathcal{R}$. To analyze this complexity, we focus on LOG-Means with $\varepsilon = 0$ and address the above mentioned three steps.

(1) Identifying which parameter to execute next can be done in $\mathcal{O}(1)$ when exploiting matching data structures. In Algorithm 1, these observations apply to lines 16 and 17, where the highest SSE ratio is identified. The middle element in this area is calculated in line 9, which can also be done in $\mathcal{O}(1)$, since it is only an arithmetical division.

(2) Executing k-Means. Due to the principle of logarithmic search, only $\mathcal{O}(log|\mathcal{R}|)$ executions of the clustering algorithm are required. However, as we do not eliminate areas with lower SSE ratios, LOG-Means could proceed with logarithmic search in eliminated and non-eliminated areas within each iteration. That is, $\mathcal{O}(log|\mathcal{R}|+log|\mathcal{R}|) = \mathcal{O}(2*log|\mathcal{R}|)$ executions of k-Means are performed at highest, which can be reduced to $\mathcal{O}(log|\mathcal{R}|)$ again.

(3) Evaluating a single clustering result via the SSE metric can be done in linear time for a single clustering result with $k = |\mathcal{C}|$ clusters, because the SSE depends linearly on the number of clusters in the dataset (cf. Equation 1), i.e., the complexity lies in $\mathcal{O}(k)$. However, since solely $\mathcal{O}(log|\mathcal{R}|)$ clustering results are evaluated, the complexity of evaluating the results also lies in $\mathcal{O}(log|\mathcal{R}|)$.

Concluding, the overall complexity of LOG-Means is $\mathcal{O}(1+log|\mathcal{R}|+log|\mathcal{R}|)$, which can be reduced to $\mathcal{O}(log|\mathcal{R}|)$. Therefore, LOG-Means outperforms existing estimation methods, which have a worst-case complexity of $\mathcal{O}(|\mathcal{R}|)$.

4 Accelerating centroid-based clustering algorithms in exploratory clustering analyses

As shown in the previous section, LOG-Means proceeds very efficiently in order to estimate the number of clusters in datasets. However, the repetitive execution of centroid-based clustering algorithms with varying values for k (cf. Fig. 1) is still at the core of LOG-Means. As already mentioned, k-Means as commonly used centroid-based clustering algorithm requires in the worst-case a super-polynomial runtime until convergence (cf. Sect. 2.2), thus making the overall exploration process on large datasets infeasible in a reasonable time frame. Therefore, the question remains whether certain characteristics of exploration processes, which address the repetitive execution of centroid-based clustering algorithms, can be exploited in order to address challenge C2.

As discussed in Sect. 2.2, especially the initialization step as well as the convergence step of centroid-based clustering algorithms are crucial in order to reduce the number of clustering iterations ω, thus avoiding a super-polynomial runtime in the worst-case. Note, that several improvements have been proposed to accelerate each clustering iteration by exploiting the triangle inequality for example [18,27,28,31,54]. Yet, still a super-polynomial number of clustering iterations is required in the worst case.

In the following, we briefly present two prior works from us in this area, which aim to reduce the number of clustering iterations. In Sect. 4.1, we present Delta Initialization, which focuses on the initialization step of centroid-based clustering algorithms and aims to re-use clustering results from previous exploration iterations for an enhanced initialization. In Sect. 4.2, we detail on the Generic Qualitative Approximation, which aims to reduce the number of clustering iterations for an execution of a centroid-based clustering algorithm. This approach provides a trade-off between the achieved clustering quality and the required runtime. Finally, we show in Sect. 4.3 that these two works can be seamlessly integrated into exploration processes, such as they are typically performed by estimation methods, in order to perform these exploration processes efficiently on large datasets.

4.1 Delta Initialization (DELTA)

Existing initialization strategies for centroid-based clustering algorithms aim to select initial centroids, which are close to the entities in $\mathcal{X}$. Therefore, several strategies exist with different characteristics.

On the one hand, there are fast initialization strategies, which require only a single pass over the dataset $\mathcal{X}$, yet solely achieve a very limited quality. A common initialization strategy following this approach is a random initialization [19]. Here, randomly chosen entities from $\mathcal{X}$ are selected as initial centroids. However, there is no guarantee regarding the quality of the chosen centroids, i.e., they can be very close to each other, resulting in (i) long runtimes of the clustering algorithm until convergence, and (ii) questionable clustering results.

On the other hand, there are slower initialization strategies, which require several passes over $\mathcal{X}$, yet with a guaranteed quality. Common initialization strategies following this approach are k-Means++ [5] and k-Means|| [8]. Here, only the first initial centroid is chosen at random, whereas the remaining $k-1$ centroids are chosen in a greedy manner by addressing the underlying data distribution. Note, that k-Means++ is inherently sequential and thus requires k rounds over $\mathcal{X}$. In contrast, the authors of k-Means|| showed that their approach can be performed in parallel and requires solely $\mathcal{O}(log\ \psi)$ rounds over $\mathcal{X}$, where ψ denotes the SSE after the first randomly chosen centroid. Therefore, less than k rounds are typically necessary. Furthermore, both strategies provably achieve an $\mathcal{O}(log\ k)$-approximation to the optimal clustering result, resulting in (i) short runtimes of the actual clustering algorithm until convergence, and (ii) more valuable clustering results compared to random initialization.

While k-Means++ and k-Means|| achieve more valuable clustering results in contrast to random-based approaches, they require a higher runtime for the initialization step. Especially for exploration processes on large datasets, where the centroid-based clustering algorithm is repeatedly executed, the runtime for the initialization step is crucial.

Our approach Delta Initialization efficiently initializes centroid-based clustering algorithms in exploration processes [24]. The underlying idea is straightforward: Instead of performing a time-consuming initialization via k-Means++ or k-Means|| for each execution of a centroid-based clustering algorithm throughout an exploration process, Delta Initialization re-uses previous clustering results. To this end, Delta Initialization assumes that $\mathcal{R}$ is explored in an ascending manner, i.e., the centroid-based clustering algorithm is executed with k_{prev} before k_{cur}, where $k_{prev} < k_{cur}$ and $k_{prev}, k_{cur} \in \mathcal{R}$. The required results are (i) the set of centroids of a previous execution of a clustering algorithm $\mathcal{C}_{prev}$, where $|\mathcal{C}_{prev}| = k_{prev}$, as well as (ii) the SSE of this previous run $\phi_{\mathcal{X}}(\mathcal{C}_{prev})$. Note, that the centroids $\mathcal{C}_{prev}$ are already at a (local) optimum position, since the clustering algorithm was already executed with k_{prev}. This enables Delta Initialization to solely add $\Delta k = k_{cur} - k_{prev}$ centroids to a previous clustering result. The concrete value for Δk depends on the underlying search strategy of the used estimation method: For exhaustive estimation methods, where the clustering algorithm is executed with every value for $k \in \mathcal{R}$, $\Delta k = 1$. For LOG-Means, Δk is typically larger due to the logarithmic search alike procedure and becomes smaller since the areas with high SSE ratios are iteratively halved (cf. Sect. 3.1).

Yet, the question remains how these Δk centroids are selected. In our previous work, we showed that these centroids can be selected according to k-Means++ and k-Means||, yet without the repeated overhead of selecting the set of $\mathcal{C}_{prev}$ centroids within each execution of the clustering algorithm. Therefore, Delta Initialization tremendously reduces the runtime for the repetitive initialization in exploration processes on large datasets. Furthermore, we showed in our previous work that Delta Initialization achieves even more valuable clustering results than initializing via k-Means|| [24], since it exploits previous clustering results, where the position of the centroids are (local) optima, which only emerge after the execution of the clustering algorithm.

4.2 Generic qualitative approximation (GQA)

As discussed in Sect. 2.2, reducing the number of clustering iterations ω is crucial to avoid long runtimes of a single execution of a centroid-based clustering algorithm, i.e., to avoid the super-polynomial runtime of k-Means in the worst-case. Implementations of centroid-based clustering algorithms allow to set a fixed threshold for ω, yet it is typically unclear how to set this threshold: Too few clustering iterations lead to results with poor quality, whereas too many clustering iterations lead to long runtimes. Hence, these thresholds are not tangible for novice analysts.

In contrast to that, GQA terminates centroid-based clustering algorithms early based on an arbitrary definable qualitative demand $q \in [0; 1]$, thus providing a tangible notion when to stop the clustering algorithm [25]. The intuition behind this approach is that analysts as well as estimation methods for the number of clusters can work with clustering results, which exhibit a high quality, while avoiding a time-consuming execution of the clustering algorithm until convergence.

Therefore, GQA relies on the following property: Regarding k-Means as instantiation of a centroid-based clustering algorithm, the sum of squared errors (SSE) is monotonically decreasing throughout each clustering iteration. For other centroid-based clustering algorithms, a similar observation can be made, since they minimize their specific notion of variance according to a different criterion. Manning et al. discuss the monotonic decrease of the SSE for k-Means in more detail [42], which can be transferred to other centroid-based clustering algorithms very similarly. We remain generic, i.e., we denote the variance of a centroid-based clustering algorithm as defined by the corresponding objective function as $\phi(\mathcal{C})$ (cf. Equation 1 for the objective function of k-Means).

Since the variance is monotonically decreasing, we derive that the quality of the clustering is becoming better in each iteration, according to the objective function. Hence, we can formulate the gain in quality as changes of the variance between two subsequent iterations. To this end, we focus on the quotient σ_i of the variance between two subsequent clustering iterations $i-1$ and i, i.e., $\sigma_i = (\phi_{i-1}/\phi_i)$. Finally, centroid-based clustering algorithms converge as soon as $\phi_{i-1} = \phi_i$, hence $\sigma_i = 1$, i.e., the variance cannot be reduced any further. As σ_i typically becomes smaller per iteration and since we do not make any further assumptions on the dataset $\mathcal{X}$ and its variance in order to preserve generality, we conclude that $\sigma_i \in [1; \infty]$.

In order to apply GQA, σ_i (and therefore the respective notion of variance) has to be calculated for each clustering iteration. Subsequently, the "check for convergence" step should be adjusted (cf. step 2c in Fig. 1), i.e., terminate the clustering algorithm as soon as Inequality 2 is satisfied.

$$1 - q \geq \sigma_i - 1 \tag{2}$$

Inequality 2 denotes that further clustering iterations would typically reduce $\sigma_i - 1$ less than $1 - q$.

In our prior work, we showed that considerable runtime savings of several orders of magnitude are possible, while regularly meeting several qualitative demands [25]. Thus, novice analysts can easily express a specific qualitative demand q and benefit from high runtime savings, which are especially of interest for exploratory clustering analyses.

Note, that GQA preserves generality regarding dataset characteristics or centroid-based clustering algorithms, since it solely addresses the variance, which can be easily derived from the objective function throughout each clustering iteration.

4.3 Integration into exploratory clustering analyses

As shown in the previous sections, DELTA and GQA provide fundamental improvements for efficiently performing centroid-based clustering algorithms in exploratory clustering analyses. Yet, the question remains how to combine these approaches and how to integrate them into exploration processes, such that existing estimation methods can seamlessly benefit thereof.

In order to preserve generality, we cling to the fundamental procedure of exploratory clustering analyses as shown in Fig. 1 and demonstrate that DELTA and GQA can be seamlessly integrated into it. To this end, we detail on the three steps of exploratory clustering analyses and show how both approaches can be integrated appropriately.

(1) Define execution settings Here, inputs for the execution of the clustering algorithm are set. In order to use DELTA, our adjusted k-Means algorithm requires two additional inputs: (i) $\mathcal{C}_{prev}$ as a previous clustering result from the exploration process, as well as (ii) $\phi(\mathcal{C}_{prev})$ as its corresponding SSE. In order to use GQA, the qualitative demand q is a required additional input. Note, that the inputs for DELTA are set implicitly by exploiting the results of previous executions of a clustering algorithm in the exploration process, whereas the input for GQA is set explicitly by analysts. This input (q - qualitative demand) is subject to the analysts' demands regarding quality and runtime. Furthermore, it can be regarded as a continuum between these dimensions: If q is set to a high value, a high quality is demanded, which requires more clustering iterations, and thus a longer runtime. Vice versa, a lower quality can be achieved in a shorter runtime. Thus, q should be set according to the analysts' demands regarding the underlying exploratory process. Note, that existing approaches in this area provide no support for novice analysts at all. Thus, we regard this parameter as a crucial step towards a better support for novice analysts.

(2) Execute clustering algorithm Here, the centroid-based clustering algorithm is executed with the predefined execution settings. Algorithm 2 outlines the adjusted procedure of k-Means. Changes in contrast to the regular k-Means algo-

Algorithm 2: Adjusted k-Means algorithm including options to use DELTA and GQA

```
Input: $\boldsymbol{\mathcal{X}}$ - dataset, $\boldsymbol{k}$ - number of clusters, $\boldsymbol{\mathcal{C}_{prev}}$ -
       set of centroids from a previous clustering
       execution (DELTA), $\boldsymbol{\phi(\mathcal{C}_{prev})}$ - SSE from
       a previous clustering execution (DELTA), $\boldsymbol{q}$ -
       qualitative demand (GQA)
Output: $\boldsymbol{\mathcal{P}}$ - a combination ($\circ$) between $\mathcal{X}$ and the assigned
        centroid $c \in \mathcal{C}_{cur}$ for each entity, $\boldsymbol{\mathcal{C}_{cur}}$ - set of final
        position of centroids, $\boldsymbol{\phi_i(\mathcal{C}_{cur})}$ - SSE
/* initialize centroids */
1  if $\mathcal{C}_{prev} == \emptyset$ then
2      $\mathcal{C}_{cur} \leftarrow$ initialize a set of $k$ centroids via k-Means++ or
       k-Means||;
3  else
4      $\Delta k \leftarrow k - |\mathcal{C}_{prev}|$;
5      $\mathcal{C}_{cur} \leftarrow$ initialize a set of $\Delta k$ centroids via DELTA
       with $\mathcal{C}_{prev}, \phi_{\mathcal{X}}(\mathcal{C}_{prev})$;
6  end
7  $i \leftarrow 0$;
8  repeat
       /* improve clustering */
9      for ( $\forall x \in \mathcal{X}$ ) {
10         $\mathcal{P} \leftarrow \{x \circ c\}$, where $c$ denotes the closest centroid to the
           entity $x$;
11     }
12     $\phi_i(\mathcal{C}_{cur}) \leftarrow 0$;
       /* change centroids */
13     for ( $\forall c \in \mathcal{C}_{cur}$ ) {
14         $c, \phi(c) \leftarrow$ new $c$ and its corresponding SSE $\phi(c)$
           according to Equation 1, where $\{x \circ c\} \in \mathcal{P}$;
15         $\phi_i(\mathcal{C}_{cur}) \leftarrow \phi_i(\mathcal{C}_{cur}) + \phi(c)$;
16     }
17     if $i > 0$ then
18         $\sigma_i \leftarrow \phi_{i-1}(\mathcal{C}_{cur})/\phi_i(\mathcal{C}_{cur})$;
19     $i \leftarrow i + 1$;
       /* check for convergence */
20 until convergence or $i > 1$ and $1 - q \geq \sigma - 1$;
21 return $\mathcal{P}, \mathcal{C}_{cur}, \phi_i(\mathcal{C}_{cur})$;
```

rithm are depicted underlined. Similar as depicted in green in Fig. 1, the algorithm proceeds in a sequence of three steps:

(2a) Initialize centroids (lines 1-6). If no previous clustering result $\mathcal{C}_{prev}$ is available, i.e., when starting the exploration process, the set of initial centroids are initialized with a state-of-the-art initialization strategy, such as k-Means++ or k-Means‖. However, if a previous clustering result is available, then Δk is calculated (line 4), and Delta Initialization is used (line 5). Subsequently, several clustering iterations i are performed, which comprise steps (2b) and (2a').

(2b) Improve clustering (lines 9-11). Here, all entities $x \in \mathcal{X}$ are assigned to the closest centroid c.

(2a') Change centroids (lines 13-18). The centroids, which are centers of gravity for centroid-based clustering algorithms, are moved to the center of the corresponding cluster according to the underlying objective function. For k-Means, the objective function is denoted in Equation 1. Note, that the overall SSE $\phi_i(\mathcal{C}_{cur})$ of a clustering iteration is the sum of the single variances for each single cluster (line 15). In order to apply GQA, σ_i is calculated in line 18. This calculation of σ_i requires at least two subsequent clustering iterations, hence the check for $i > 0$ in line 17.

(2c) Check for Convergence (line 20). Typically, centroid-based clustering algorithms converge when no changes between two subsequent clustering iterations occur, i.e., entities do not change their cluster membership. We exploit GQA to set an termination criterion according to Inequation 2 (line 20). Again, at least two subsequent clustering iterations need to be performed in order to use σ_i.

Finally, the adjusted k-Means algorithm returns the same triple as the regular algorithm: (i) a set $\mathcal{P}$, which is a combination between $\mathcal{X}$ and the assigned centroid $c \in \mathcal{C}_{cur}$ for each entity, (ii) a set $\mathcal{C}_{cur}$, which is a set of the final centroids, and (iii) $\phi_i(\mathcal{C}_{cur})$ as the corresponding SSE value. Note, that specific implementations of k-Means may return only a subset of this triple, such as solely $\mathcal{P}$.

(3) Evaluate clustering result Here, the clustering result $\mathcal{P}$ is evaluated according to the concrete estimation method (cf. Sect. 2.1).

It should be emphasized, that the results $\mathcal{C}_{cur}$ and $\phi_i(\mathcal{C}_{cur})$ of the current iteration of the exploration process are used as input, i.e., they become $\mathcal{C}_{prev}$ and $\phi(\mathcal{C}_{prev})$ in the next iteration. Finally, note, that Algorithm 2 preserves generality, i.e., it can be used by (a) several estimation methods following the exploration process outlined in Fig. 1, and (b) several centroid-based clustering algorithms, since solely the objective function in line 14 has to be adjusted accordingly.

5 Evaluation

As we have shown in the previous section, DELTA and GQA can seamlessly be integrated into exploratory clustering analyses. Estimation methods, such as LOG-Means can thus benefit thereof. Yet, it is unclear to what extent these methods benefit from DELTA and GQA. Hence, the two main questions for our evaluation are: (i) To what extent do DELTA and GQA accelerate estimation methods? (ii) What is the quality of the estimates provided by estimation methods when using DELTA and GQA?

To provide a clear overview of the evaluation, we summarize the key messages in Table 1. Furthermore, this table shows improvements over the corresponding state-of-the-art and unveils which of the aforementioned challenges are addressed, respectively.

The remainder of this section is structured as follows: Firstly, we discuss the experimental setup. Secondly, we investigate the runtime benefits, when using DELTA and GQA within various commonly used estimation methods. Thirdly, we unveil the effects on the accuracy of estimation

Table 1 Overview of important key messages of the evaluation and corresponding challenges. Improvements over state-of-the-art show average values across all synthetic datasets, whereas values for the largest synthetic dataset S-XXVII are shown underlined

Abbr.	Key message	Improvements over State-of-the-art	Challenge
KM1	LOG-Means outperforms existing estimation methods.	– Speedups of almost 21× ($\underline{27\times}$) are achievable, which corresponds to > 8 minutes ($\underline{> 7\text{ h}}$) – Up to 54× ($\underline{100\times}$) more accurate estimates are achievable	C1
KM2	Existing estimation methods benefit from k-Means methods, and achieve the best results mostly with DELTA+GQA	– Speedups of almost 6× ($\underline{6\times}$) are achievable, which corresponds to > 7 minutes ($\underline{> 6\text{ hours}}$) – Very accurate estimates with a relative error of 0.1 % ($\underline{0.2\ \%}$) and even less are achievable	C2
KM3	LOG-Means and GQA is best suited for exploratory clustering analysis	– Speedups of more than 34× ($\underline{34\times}$) are achievable, which corresponds to almost 9 minutes ($\underline{> 7\text{ hours}}$) – Up to 45× ($\underline{50\times}$) more accurate estimates are achievable	C1, C2, C3
KM4	LOG-Means and GQA is a strong fit for large-scale exploration processes	– Fast estimates in large-scale exploration processes – Accurate estimates with a very small relative error are achievable	C3

methods, when using DELTA and GQA. Fourthly, we present results on real-world datasets, since the previous results were made on synthetic datasets. Finally, we unveil how DELTA and GQA influence LOG-Means in particular large-scale exploration processes.

5.1 Experimental setup

In this section, we present the hardware and software setup for our experiments. Furthermore, we detail the characteristics of the used datasets, before we discuss the implementation and details of the experiments.

5.1.1 Hardware and software

We conducted all of our experiments on a distributed Apache Spark cluster. It consists of one master node and six worker nodes. The master node has a 12-core CPU with 2.10 GHz each and 192 GB RAM. Each worker has a 12-core CPU with 2.10 GHz each and 160 GB RAM. Each node in this cluster operates on Ubuntu 18.04. We installed OpenJDK 8u191, Scala 2.11.12 and used Apache Hadoop 3.2.0 as well as Apache Spark 2.4.0.

5.1.2 Datasets

Existing works on methods to estimate the number of clusters in datasets focus on rather small datasets [12,15–17,44,46,49,51]. They rely on synthetic datasets with different numbers of entities (up to 36,000), dimensions (up to 10), and clusters in the dataset (up to 150, however for small datasets). Furthermore, the distribution of the used datasets varies: some use a Gaussian distribution for each cluster, others create a 2-dimensional dataset and create the clusters manually by placing the entities close to each other.

For our evaluation, we conducted a comprehensive comparison across many existing estimation methods that considers more voluminous datasets and is more systematic with respect to varying dataset characteristics. To this end, we focus on 27 synthetic datasets with controlled dataset characteristics as well as on 5 real-world datasets, resulting in 32 datasets in total.

We implemented a synthetic dataset generator. It generates datasets based on the following input parameters: n as the number of entities, d as the number of dimensions, and c as the number of clusters, where each cluster contains n/c entities. Our tool generates datasets with values that lie within the range $[-10; 10]$ for each dimension. Each cluster has a Gaussian distribution with the mean at the center and a standard deviation of 0.5. The c centers are chosen randomly and the clusters are non-overlapping. Table 2 depicts the characteristics of the 27 synthetic datasets used for the evaluation.

In addition, we used 5 real-world classification datasets mostly from the UCI machine learning library,[1] which are regularly used to benchmark new algorithms. In order to use these datasets for clustering, we removed any non-numeric and symbolic values, IDs, timestamps, class labels and empty values. Table 3 summarizes the datasets' characteristics.

[1] https://archive.ics.uci.edu/ml/datasets.php

Table 2 Characteristics of the used 27 synthetic datasets

Abbr.	n	d	c
S-I - S-III	10,000	10	$\{10; 50; 100\}$
S-IV - S-VI	10,000	50	$\{10; 50; 100\}$
S-VII - S-IX	10,000	100	$\{10; 50; 100\}$
S-X - S-XII	100,000	10	$\{10; 50; 100\}$
S-XIII - S-XV	100,000	50	$\{10; 50; 100\}$
S-XVI - S-XVIII	100,000	100	$\{10; 50; 100\}$
S-XIX - S-XXI	1,000,000	10	$\{10; 50; 100\}$
S-XXII - S-XXIV	1,000,000	50	$\{10; 50; 100\}$
S-XXV - S-XXVII	1,000,000	100	$\{10; 50; 100\}$

Table 3 Characteristics of the used 5 real-world datasets, where c denotes #classes

Abbr.	Dataset	n	d	c
R-I	Avila	10,430	10	12
R-II	Dataset for Sensorless Drive Diagnosis	58,509	48	11
R-III	MNIST	60,000	784	10
R-IV	KDD Cup 1999 Data	4,898,431	34	23
R-V	KITSUNE Network Attack Dataset	21,017,597	115	10

Note, that these datasets exhibit similar or even larger characteristics as the synthetic datasets regarding n and d.

5.1.3 Implementation

Analogous to the generic exploration process outlined in Fig. 1, we investigated several estimation methods, which perform the exploration process in an automated manner, as well as several k-Means methods as implementation of a centroid-based clustering algorithm. In the following, we detail both aspects and define two experiments for the parameter search space $\mathcal{R}$.

Estimation methods

Besides LOG-Means, we implemented several estimation methods on Apache Spark. This includes very commonly used exhaustive and non-exhaustive estimation methods as described in Sect. 2. Table 4 summarizes the 13 methods that we used throughout the evaluation as well as their abbreviations that we use for referring to them. Since some estimation methods draw on parameters, we used recommendations provided by the authors of the respective estimation method, where available. The names of the parameters cling to the definition of the corresponding authors and can be found in Table 4. For BIC and X-Means, we used an existing implementation.[2] Since multiple scoring criteria can be used for X-Means, we used the Akaike Information Criterion (XAI) and the Bayesian Information Criterion (XBI) separately. We used Spark's variation of the Silhouette coefficient.[3] Regarding LOG-Means, we set $\varepsilon = 0$, i.e., only logarithmic search is performed. In our prior work, we showed that $\varepsilon > 0$ leads to better estimates, yet with the trade-off of a higher runtime [23].

If an estimation method failed to provide an estimation within a predefined time budget of 30 minutes, we stopped the execution and mark the corresponding estimation as failed. This time budget solely includes the runtime for identifying which parameter to execute and to evaluate the clustering result, and not the runtime for the repetitive execution of k-Means.

k-Means methods

We focus on k-Means as instantiation of a centroid-based clustering algorithm due to its overwhelming popularity [53]. To this end, we implemented 6 methods to perform k-Means (note the two qualitative demands for each GQA variant).

BASE The baseline for our evaluation is Spark's MLlib implementation of k-Means. By default, this implementation uses k-Means|| for the initialization step, performs at highest $\omega = 20$ clustering iterations or terminates, as soon as the centroids change less than $eps = 1 \times 10^{-4}$ in the Euclidean space within two subsequent clustering iterations. The values

[2] https://git.io/Je1sm

[3] https://git.io/JfnPa

Table 4 Estimation methods for the evaluation

Abbr.	Name	Parameters	Characteristic
AIC	Akaike Information Criterion [2]		Exhaustive, automatic
BIC	Bayesian Information Criterion [47]		
CHI	Calinski-Harabasz Index [12]		
CJI	Coggins-Jain Index [15]		
DBI	Davies-Bouldin Index [16]		
DUI	Dunn Index [17]		
JUM	Jump Method [49]	$Y = r/2$	
SIL	Silhouette coefficient [46]		
GAP	Gap Statistic [51]	$b = 5$	Non-exhaustive, ascending, automatic
GME	G-Means [30]	$\alpha = 0.0001$	
XAI	X-Means (AIC) [44]		
XBI	X-Means (BIC) [44]		
LOG	LOG-Means [23]	$\varepsilon = 0$	Non-exhaustive, logarithmic, automatic

for *eps* and ω are arbitrarily chosen in Spark's implementation and not based on scientific works or the like. That is, without these termination criteria, even higher runtimes are expected for BASE.

DELTA We use Delta Initialization as described in Sect. 4.1. That is, we use k-Means|| for the first initialization within an exploration process conducted by each estimation method, and Delta Initialization for the subsequent initializations. Furthermore, we leave the termination criteria as defined for BASE.

GQA90 & GQA99 We use GQA for terminating an execution of k-Means as described in our previous work [25] and in Sect. 4.2. That is, we use the same initialization as for BASE, yet change the convergence step. To this end, we use two qualitative demands, i.e., 90 % and 99 %, since we evaluated the impact of both approaches for single clustering results in our previous work.

DELTA+GQA90 & DELTA+GQA99 Lastly, we combine DELTA and GQA. That is, we use DELTA for the subsequent initialization in the exploration process and terminate the clustering algorithm with GQA as soon as the qualitative demands of 90 % and 99 % are met, respectively.

5.1.4 Experiments

Regarding the search space, we conducted two different experiments. For each experiment, we will present the runtime measurements as well as the accuracy results. Note, that we performed all runs three times and present median or average values in the results.

Experiment 1 The goal of this experiment is to simulate rather strong domain knowledge of the analyst. Based on this domain knowledge, the analyst is able to drastically reduce the search space. We simulate this by setting the search space $\mathcal{R}$ to $[2; 2c]$ for synthetic datasets and $\mathcal{R}$ to $[0.5c; 2c]$ for real-world datasets across all estimation methods, where c denotes the actual number of clusters in a dataset. Note, that we set $\mathcal{R}$ different on real-world datasets, since we are able to exploit available domain knowledge, i.e., we know that those datasets contain more than 2 classes and are commonly used for multi-class classification problems. In total, we performed almost 7,500 runs (= 32 datasets $\times$ 13 estimation methods $\times$ 6 k-Means methods $\times$ 3 repetitions).

Experiment 2 Here, we simulate less prior knowledge of the analyst, as it is typical for novice analysts. The goal is to demonstrate the benefits of LOG-Means for rather inexperienced analysts, who have only little domain knowledge and can therefore limit the search space only very vaguely. Therefore, we set the search space $\mathcal{R} = [2; 10c]$ for synthetic datasets and $\mathcal{R} = [0.5c; 10c]$ for real-world datasets and perform solely LOG-Means with $\varepsilon = 0$. In total, almost 600 runs (= 32 datasets $\times$ 1 estimation method $\times$ 6 k-Means methods $\times$ 3 repetitions) are performed. The results for experiment 2 are marked with an asterisk (LOG*) in the presentation of the results.

Throughout both experiments, we performed almost 8,100 runs. The overall runtime of our experiments was more than 81 full days.

5.2 Runtime

Figure 4 unveils the overall runtime of estimation methods combined with each k-Means method over all synthetic datasets. Exhaustive estimation methods are shown in subfigures (a) to (h), whereas non-exhaustive estimation methods are shown in subfigures (i) to (n). The percentage of failed

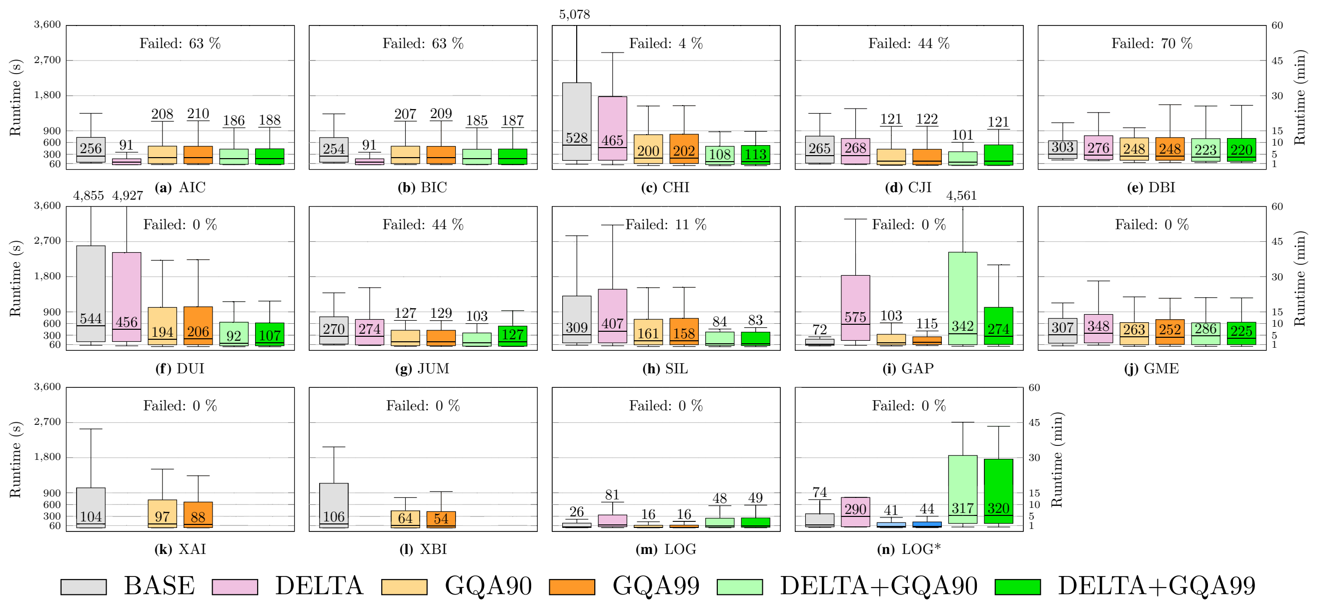


Fig. 4 Runtime of successful estimates across all synthetic datasets. Estimation methods in combination with k-Means methods are shown per subfigure. Median runtimes are depicted in box plots. Note, that DELTA cannot be used within X-Means (XAI, XBI), since the used dataset changes per exploration iteration for X-Means. Values for cut-off whiskers are shown on top of the corresponding subfigures

estimations due to the given time frame is denoted at the top of each subfigure. Note, that this reflects the failed estimations for BASE. For the remaining k-Means methods, we solely investigated those datasets, where BASE was able to provide an estimate within the given time budget to make the results comparable.

We group our observations according to the key messages (cf. Table 1) and explain them in more detail in the following.

KM1 For BASE as k-Means method, LOG-Means significantly outperforms existing estimation methods. Especially exhaustive estimation methods typically suffer from long runtimes, since they explore the whole parameter search space. Therefore, these methods often fail to provide an estimate in the given time frame of 30 minutes. Note, that for those exhaustive estimation methods, that only failed a few times (e.g., CHI and DUI), the runtime is significantly higher for the remaining methods, since they were able to provide estimates for rather large datasets as well. In contrast to that, existing non-exhaustive estimation methods always succeed to provide an estimate in the given time frame. Yet, as our later evaluation unveils, their accuracy is often very low.

Comparing the median runtimes of LOG with DUI, a speedup of almost $21\times$ can be achieved, which corresponds to absolute runtime savings of more than 8 minutes for exploratory clustering analyses. For the largest synthetic dataset S-XXVII, a speedup of $27\times$ can even be achieved, which corresponds to absolute runtime savings of more than 7 hours. These runtime savings are expected due to the greedy logarithmic-search alike procedure of LOG-Means. Remember, that in the worst case, all existing estimation methods perform an exhaustive search in the parameter search space, resulting in long overall runtimes.

KM2 Existing estimation methods benefit from the proposed k-Means methods in terms of runtime. While Delta Initialization leads in most cases to rather smaller runtime savings or even a rise in the runtime, GQA achieves strong runtime savings. This is expected, since the goal of GQA is to reduce the number of clustering iterations, where in each clustering iteration several distance calculations are performed. Note, that the increased qualitative demand for GQA leads to slightly higher runtimes in most cases, as more clustering iterations have to be performed.

DELTA+GQA achieves the most tremendous runtime savings for most estimation methods. This is expected, since DELTA re-uses previous clustering results and provides therefore a faster initialization (cf. Sect. 4.1), whereas GQA leads to less clustering iterations (cf. Sect. 4.2). Hence, especially for exhaustive estimation methods (subfigures a–h in Fig. 4), DELTA+GQA is in most cases the fastest k-Means method. The sole exception to this rule is AIC and BIC on synthetic datasets, where Delta Initialization without GQA leads to the fastest results. However, as our later evaluation unveils, the results of these methods are quite inaccurate.

For non-exhaustive estimation methods (subfigures i-l), the results must be considered in a more differentiated manner. Remember, that methods in subfigures i-l perform an ascending search in the underlying search space $\mathcal{R}$ and stop as soon as subsequent clustering results differ only marginally.

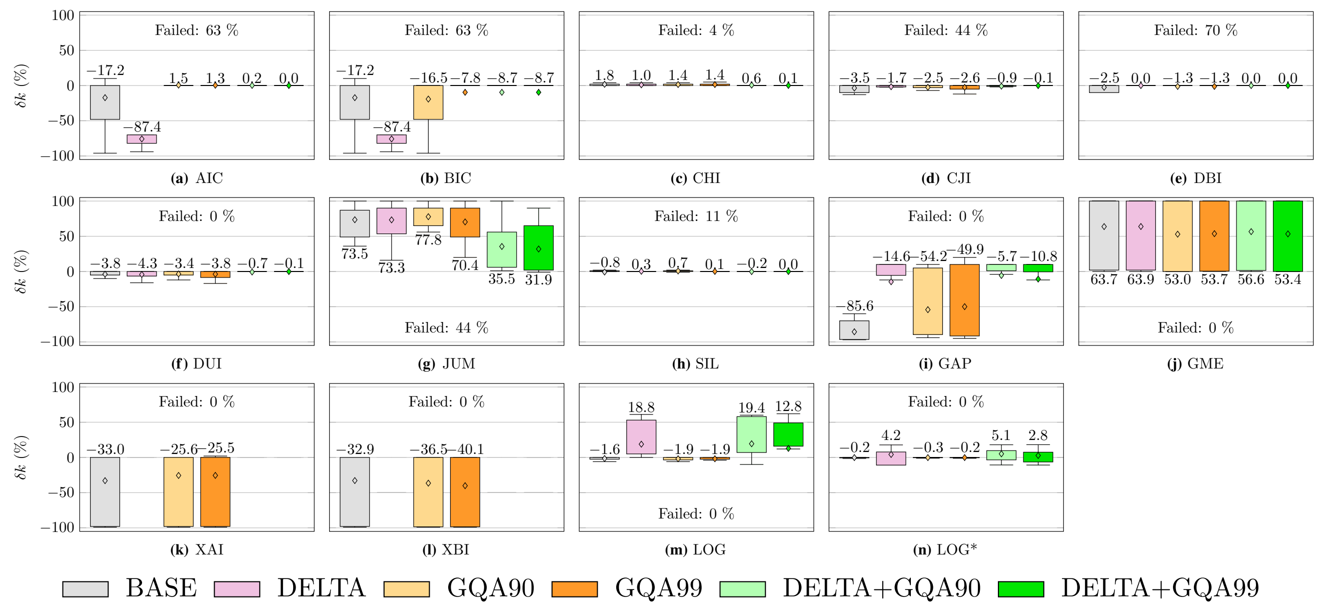


Fig. 5 Comparison of the accuracy of estimation methods across all k-Means methods. Average value for δk across all synthetic datasets is depicted per box plot. Note, that DELTA cannot be used within X-Means (XAI, XBI), since the used dataset changes per exploration iteration for X-Means

For GAP, we observe a rise of the runtime for all k-Means methods compared to BASE. As the later evaluation unveils, these longer runtimes however lead to more precise estimates. For GME, XAI and XBI, the runtime savings are rather smaller, since they split the clusters into smaller clusters in each exploration iteration. Hence, the clustered data becomes smaller in each exploration iteration, i.e., there are less possibilities for improvements.

We observe the strongest runtime savings of almost 6× for DUI when using DELTA+GQA90 instead of BASE, i.e., absolute runtime savings of more than 7 minutes. For the largest synthetic dataset S-XXVII, the same speedup was observed, yet the absolute runtime savings correlate to more than 6 hours.

KM3 Regarding the shortest runtime, we can clearly state that LOG with GQA as k-Means method leads to the shortest median runtime throughout all results. That is, it requires 16 seconds in median, whereas other combinations of estimation methods and k-Means methods have longer runtimes. Interestingly, using LOG with k-Means methods containing DELTA lead to rather longer runtimes, i.e., even longer than for BASE. Since Δk is rather large for the underlying logarithmic search alike procedure of LOG, we argue that adding the missing Δk centroids to the infimum of already performed clustering results is not a promising approach for this estimation method. However, reducing the number of clustering iterations via GQA leads to promising speedups of 34×, when compared to DUI with BASE as k-Means method. This correlates to absolute runtime savings of almost 9 minutes. For the largest synthetic dataset S-XXVII, we also observe a speedup of 34×, yet this speedup correlates to runtime savings of more than 7 hours.

KM4 Concerning the second experiment for LOG* (cf. Fig. 4n), we remember that LOG* explores a five times larger parameter search space than the remaining estimation methods. However, we observe that the runtime for BASE is comparable to most estimation methods, which emphasizes the feasibility of LOG-Means for very large parameter search spaces. Again, GQA is able to significantly reduce the runtime of LOG-Means even for these large search spaces by almost the half. Thus, the combination of LOG* and GQA even outperforms most existing estimation methods with combinations of the presented k-Means methods.

These results emphasize that especially the combination of (i) novel estimation methods, which reduce the number of exploration iterations, and (ii) novel k-Means methods, which reduce the number of clustering iterations lead to significant runtime savings compared to state-of-the-art methods for exploratory clustering analyses. Yet, the accuracy of the estimates is also very relevant, which we investigate next.

5.3 Accuracy

Since the number of clusters is known for all datasets in our evaluation, we exploit this and define the accuracy as relative error $\delta k \in [-1; 1]$. Equation 3 formalizes our notion of the relative error, where k_{est} denotes the estimated value for the number of clusters provided by the respective estimation method. Hence, the closer δk to zero, the more accurate the estimation method.

$$\delta k = \frac{k_{est} - c}{max\{max(\mathcal{R}) - c; c - min(\mathcal{R})\}} \tag{3}$$

This notion of the relative error allows us to identify to what extent certain estimation methods tend to under- or overestimate the number of clusters. Note, that δk solely addresses successful estimations, i.e., estimations within the given time budget.

Figure 5 summarizes the results across all investigated estimation methods and k-Means methods for synthetic datasets. The percentage of failed estimations for BASE are again depicted within each subfigure. Again, we group our observations according to key messages KM1-4.

KM1 LOG-Means is at least en par, yet mostly more accurate than existing estimation methods for BASE. As the results in Fig. 5 show, solely CHI, SIL and LOG are able to achieve accurate estimates with a relative error below 2 % for BASE. However, as our investigation of the runtime unveiled (cf. Sect. 5.2), CHI and SIL have a significantly higher runtime than LOG. The results emphasize that exploiting the underlying elbow graph leads to accurate estimates for LOG-Means. Compared with the least accurate estimation method for BASE (GAP), LOG achieves 54× more accurate estimates. For the largest synthetic dataset S-XXVII, LOG-Means even achieves 100× more accurate estimates than existing estimation methods.

In our previous paper, we unveiled that exploiting ε within LOG-Means can lead to even more accurate estimates, yet with the trade-off of a higher runtime [23].

KM2 Estimation methods generally benefit from the proposed k-Means methods regarding their accuracy. For almost all exhaustive estimation methods on synthetic datasets, very accurate estimates can be achieved with a relative error of almost 0.0 % for DELTA+GQA99 (cf. Fig. 5a, c–f, h). Solely BIC and JUM achieve less accurate estimates, yet significantly more accurate estimates than with BASE. Non-exhaustive estimation methods also benefit, yet to another extent. For example, the accuracy of GAP improves from −85.6 % (BASE) to −5.7 % (DELTA+GQA90).

The most accurate results for each estimation method are achieved with different k-Means methods, yet in most cases for DELTA+GQA99. The reason for these improvements in accuracy are twofold: a) Regarding the benefits of DELTA, we unveiled in our previous paper that it leads to better separated clusters [24], which may be favorable for specific estimation methods. Furthermore, DELTA draws on less randomness compared to an initialization from scratch for each $k \in \mathcal{R}$, since DELTA re-uses previous clustering results and adds Δk centroids in a deterministic manner. Therefore, the clustering results of the exploration process are more comparable, thus achieving more accurate estimates. b) Regarding the benefits of GQA, it is clear that it performs a regularization due to its approximation of the final clustering result. Remember, that GQA terminates the clustering algorithm as soon as a given qualitative demand is met, thus avoiding many clustering iterations [25]. This regularization is a strong fit for several estimation methods, as it enables them to achieve more accurate estimates than for BASE.

Putting the effect of DELTA and GQA together, the relative error δk can be reduced to 0.1 % or even less (0.2 % on dataset S-XXVII) for most estimation methods, as seen in Fig. 5.

KM3 LOG and GQA achieve accurate estimates with a relative error of -1.9 %. Remember, that this combination also leads to very fast estimates and outperforms all other combinations of estimation methods and k-Means methods in terms of runtime (cf. Fig. 4m). Therefore, we can clearly argue, that LOG-Means with GQA is very fast and at the same time achieves accurate estimates.

In order to assess the improvements over state-of-the-art, we compare LOG and GQA with the least accurate existing estimation method and BASE, i.e., GAP. Therefore, around 45× more accurate estimates can be achieved by LOG and GQA (50× for the largest synthetic dataset S-XXVII). Hence, we can state that LOG-Means and GQA lead to fast and accurate estimates, thus emphasizing its feasibility for exploratory clustering analyses.

KM4 Regarding the second experiment for LOG* (cf. Fig. 5n), we make a very similar observation. Here, LOG* achieves with BASE already very accurate estimates with a relative error of $\delta k = -0.2$ %. With GQA99, LOG* achieves the same accurate estimation.

This result emphasizes that LOG-Means can also achieve very accurate estimates in large parameter search spaces. Especially in comparison to existing estimation methods, it is evident that LOG-Means significantly outperforms existing estimation methods in terms of runtime and accuracy.

5.4 Evaluation on real-world data

As the previous evaluations focus on synthetic datasets, we also investigate the effects on real-world datasets from Table 3. Figures 6 and 7 unveil the runtime and accuracy results ove these real-world datasets.

In general, it should be noted that the runtime and accuracy values differ from the results on synthetic datasets (cf. Figs. 4 and 5). There are two reasons for these differences: (1) We set the parameter search space $\mathcal{R}$ relative to the actual number of clusters (or classes) c in a dataset. As the real-world datasets have solely up to 10 classes, the absolute parameter search space $|\mathcal{R}|$ is smaller than for synthetic datasets, which have up to 100 clusters. (2) The characteristics of the real-world

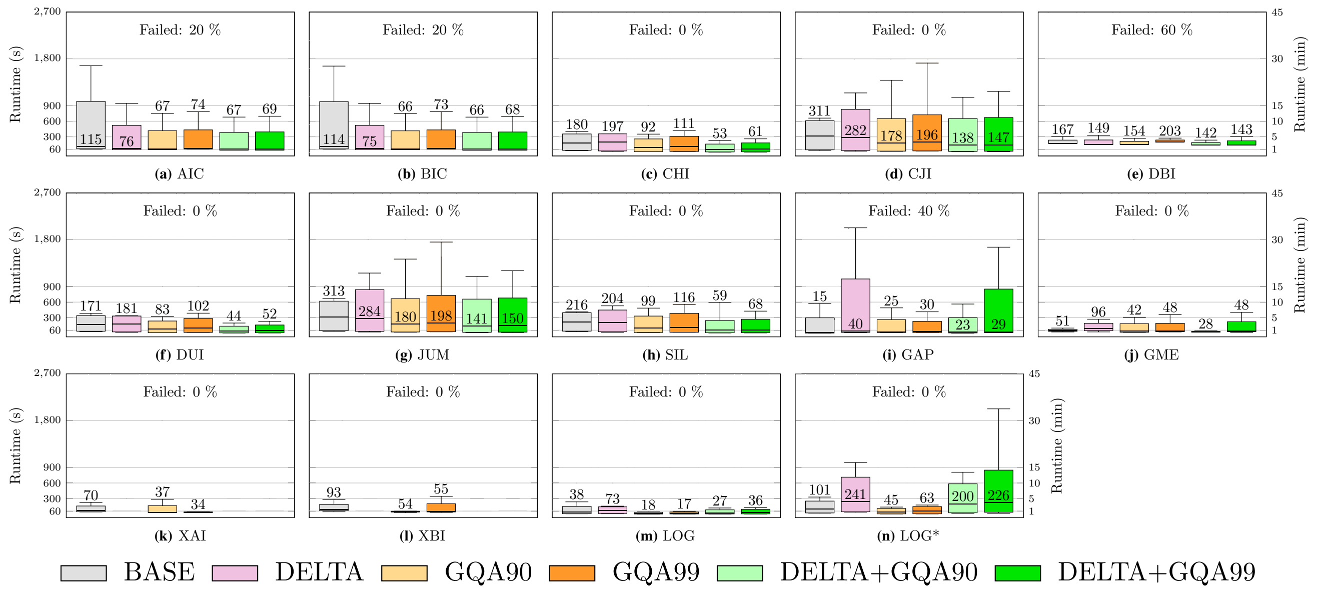


Fig. 6 Runtime of successful estimates across all real-world datasets. Note, that DELTA cannot be used within X-Means (XAI, XBI), since the used dataset changes per exploration iteration for X-Means. Median runtimes are depicted in box plots

datasets differ from the ones on synthetic datasets. Remember that on synthetic datasets, each cluster is represented as an agglomeration of entities following a Gaussian distribution. On real-world datasets however, the classes follow various distributions. Hence, it is typically more difficult to estimate the number of classes on real-world datasets.

We group our detailed observations according to the aforementioned key messages (cf. Table 1). In general, we make very similar observations and can thus verify the aforementioned effects of synthetic data on real-world data, i.e., in a practical scenario.

KM1 LOG-Means outperforms existing estimation methods in terms of runtime and accuracy for BASE. Regarding the runtime, solely GAP achieves a comparable runtime to LOG, yet with less accurate estimates. Regarding the accuracy, CJI and DUI achieve similar accurate estimates, yet they require a multiple of the runtime of LOG. Hence, LOG-Means achieves the best combination of runtime and accuracy in contrast to existing estimation methods.

KM2 Estimation methods benefit from the presented k-Means methods in terms of runtime and accuracy. For example, on DUI, the median runtime of 171 seconds (BASE) can be reduced to 44 seconds (DELTA+GQA90), i.e., a speedup of almost $4\times$ can be achieved. Again, using DELTA+GQA over BASE leads in most cases to the highest runtime savings. Similarly, DELTA+GQA also leads to more accurate estimates for most estimation methods.

KM3 LOG (LOG-Means) benefits from GQA in terms of runtime. That is, the runtime can again be more than halved from 38 seconds for BASE to 17 seconds for GQA90. However, the estimates become more inaccurate. While LOG-Means achieves estimates with a relative error of $\delta k = 11.7\ \%$ for BASE, the estimates with GQA90 worsen to $\delta k = 34.0\ \%$. The reason for these rather imprecise estimates can be found in the underlying procedure of LOG-Means: Remember, that LOG-Means exploits the elbow graph (cf. Fig. 3). Since the data distribution in real-world datasets is dependent on the specific use case and often contains noise, the elbow graph can contain several outliers in its decreasing trend. Furthermore, when using GQA, i.e., performing an approximation to each clustering result, the spikes in the elbow graph become more severe, resulting in potentially less accurate estimates.

KM4 Regarding the results of LOG* from experiment 2 (cf. Figs. 6n and 7n), we make the same observations as on synthetic datasets: LOG* achieves comparable runtimes than existing estimation methods, yet it explores a five times larger parameter search space. Furthermore, with a relative error of $\delta k = -4.1\ \%$, it achieves very accurate estimates. Furthermore, using GQA within LOG* halves the median runtime and still achieves similarly accurate estimates. Hence, we can verify that LOG-Means achieves fast and accurate results in large parameter search spaces on real-world datasets.

5.5 Large-scale exploration processes

As the initial motivation of this paper is to perform efficient exploratory clustering analyses in large-scale exploration processes, we investigate this aspect in detail. Hence, the following discussion especially details aspects for KM4.

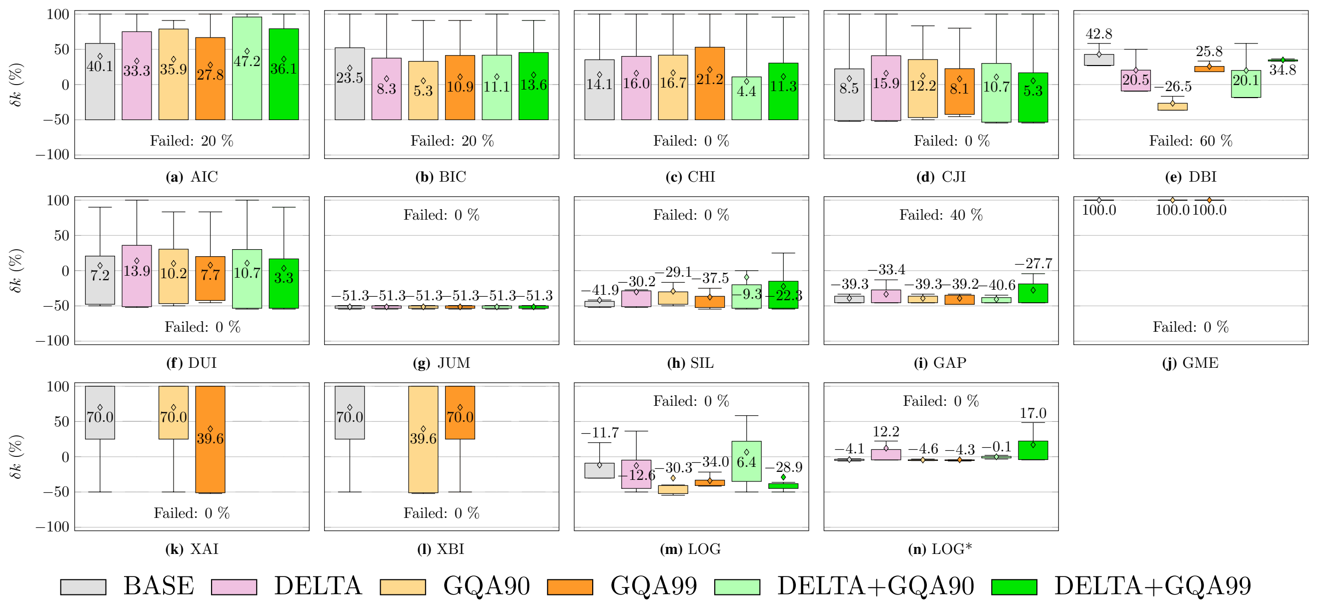


Fig. 7 Comparison of the accuracy of estimation methods across all k-Means methods. Average value for δk across all real-world datasets is depicted per box plot. Note, that DELTA cannot be used within X-Means (XAI, XBI), since the used dataset changes per exploration iteration for X-Means

Figure 8 shows the results for all estimation methods on BASE in comparison to LOG* with all k-Means methods on the largest synthetic dataset S-XXVII and the largest real-world dataset R-V. Remember, that we set the parameter search space $\mathcal{R}$ relative to the actual number of clusters in a dataset. As dataset S-XXVII contains 100 clusters, $\mathcal{R} = [2; 200]$ for all estimation methods in Fig. 8a, e. We set the search space for the real-world dataset R-V very similarly (cf. Sect. 5.1). For LOG*, we increased these search spaces by a factor of 5.

We make similar observations as in the previous subsections: Many existing estimation methods fail to provide an estimate in the given time frame, since they perform an exhaustive search or draw on complex metrics to make an estimate. However, if they make an estimation, they require a long runtime (e.g., DUI in Fig. 8a), or they are very imprecise (e.g., GME in Fig. 8e). Similar observations apply to real-world data in Fig. 8c and g. Solely LOG is able to achieve fast and accurate estimates for synthetic and real-world data.

Moreover, we can clearly state that LOG* can compete with existing estimation methods with BASE as k-Means method, while still providing very accurate estimates. Remember, that LOG* explores a five times larger parameter search space than the remaining estimation methods. This emphasizes the feasibility of LOG-Means for very large parameter search spaces, such as they might be defined by novice analysts.

Furthermore, LOG* can also be combined with the presented k-Means methods. Again, k-Means methods with DELTA lead to rather long runtimes due to the logarithmic search alike procedure of LOG-Means. However, combining LOG* with GQA leads to a speedup of roughly 3×, while the estimates are still very accurate. Hence, this combination significantly outperforms state-of-the-art approaches for exploratory clustering analyses, since it is able to explore five times larger parameter search spaces in a shorter time frame than most existing estimation methods, while also providing more accurate estimates in many cases.

Concluding, it is evident that the interdependency between LOG-Means and GQA is a very strong fit for large-scale exploration processes, since it significantly outperforms state-of-the-art approaches in terms of runtime and accuracy. Therefore, especially novice analysts achieve profound support for exploratory clustering analyses in a reasonable time frame with very accurate estimates for the number of clusters.

6 Conclusion

Clustering is commonly used for manifold purposes, such as assessing the structure of data, grouping entities or compressing data. Especially large parameter search spaces (C1), large datasets (C2), or large-scale exploration processes as the combination thereof (C3) pose a particular pitfall, since exploratory clustering analyses require huge overall runtimes until a valuable clustering result can be achieved.

We summarized state-of-the-art approaches to tackle challenges C1 and C2. These challenges were so far addressed

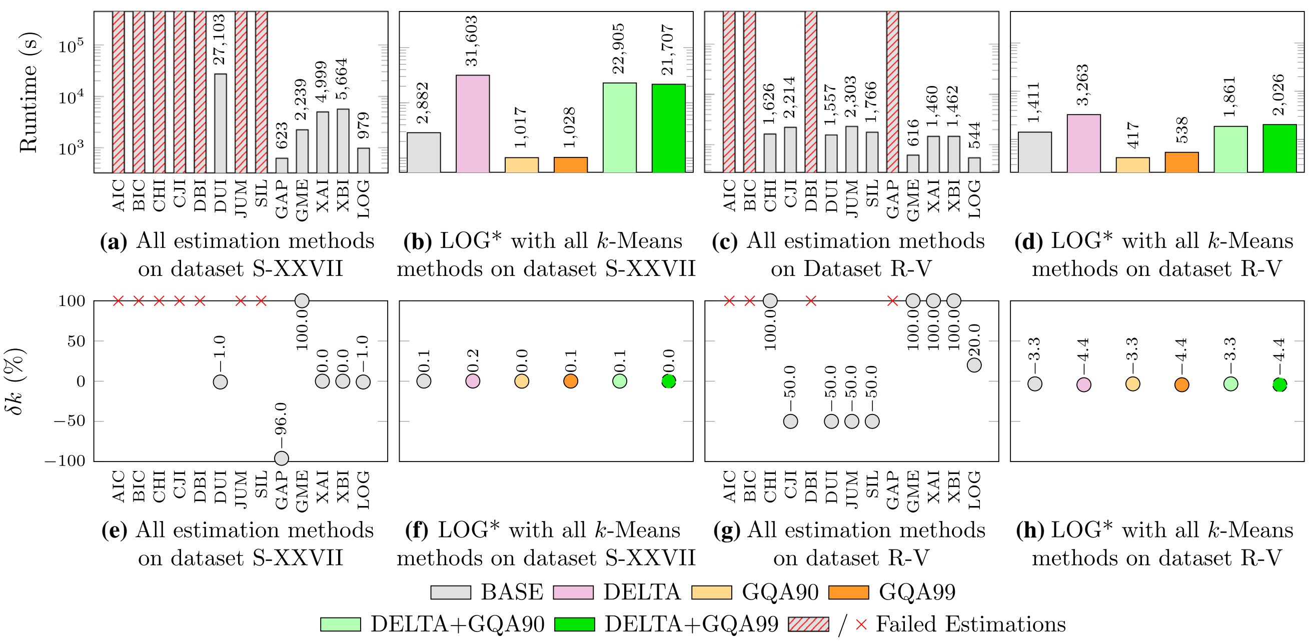


Fig. 8 Runtime (**a**–**d**) and accuracy (**e**–**h**) comparison on the largest synthetic dataset S-XXVII (left) and the largest real-world dataset R-V (right). Median values over 3 runs are depicted

independently from another. However, their interdependency (C3) is of particular interest for large-scale exploration processes, since especially novice analysts typically define large parameter search spaces on large datasets.

In this paper, we showed that existing contributions regarding both challenges can be seamlessly integrated with each other. Furthermore, we systematically evaluated the benefits of this interdependency on large-scale exploration processes. Our comprehensive evaluation compares LOG-Means, our novel estimation method for valuable clustering results, with 12 commonly used estimation methods on large datasets and large search spaces. To the best of our knowledge, this is the most systematic comparison as of today. The results unveil that the proposed methods significantly outperform existing approaches for exploratory clustering analyses. Furthermore, these methods provide accurate and fast estimates even for large-scale exploration processes and are therefore of paramount interest for novice analysts.

Future work in this area should investigate, whether clustering results, which arise in each exploration iteration, can be assembled to ensembles in order to achieve even better clustering results [33,34]. Hence, it would be very appealing to investigate whether the intermediate clustering results achieved by LOG-Means could be used to further improve its accuracy.

Acknowledgements This research was partially funded by the Ministry of Science of Baden-Württemberg, Germany, for the Doctoral Program "Services Computing". Some work presented in this paper was performed in the project "INTERACT" as part of the Software Campus program, which is funded by the German Federal Ministry of Education and Research (BMBF) under Grant No.: 01IS17051. Finally, we would like to thank Tim Niederhausen and Julius Voggesberger for their implementation work.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Abdullah, A., Kumar, R., McGregor, A., Vassilvitskii, S., Venkatasubramanian, S.: Sketching, embedding, and dimensionality reduction for information spaces. In: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, vol. 41, pp. 948–956 (2016)
2. Akaike, H.: A new look at the statistical model identification. IEEE Trans. Autom. Control **19**(6), 716–723 (1974)
3. Aloise, D., Deshpande, A., Hansen, P., Popat, P.: NP-hardness of Euclidean sum-of-squares clustering. Mach. Learn. **75**(2), 245–248 (2009)

4. Arthur, D., Vassilvitskii, S.: How slow is the k-means method? In: Proceedings of the Annual Symposium on Computational Geometry, vol. 2006, pp. 144–153 (2006)
5. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1025 (2007)
6. Bachem, O., Lucic, M., Krause, A.: Scalable k-means clustering via lightweight coresets. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1119–1127 (2018)
7. Bachem, O., Lucic, M., Lattanzi, S.: One-shot coresets: the case of k-clustering. In: International Conference on Artificial Intelligence and Statistics (AISTATS) (2018)
8. Bahmani, B., Moseley, B., Vattani, A., Kumar, R., Vassilvitskii, S.: Scalable K-means++. PVLDB **5**(7), 622–633 (2012)
9. Baldi, P., Hatfield, G.W.: DNA Microarrays and Gene Expression: From Experiments to Data Analysis and Modeling. Cambridge University Press, Cambridge (2002)
10. Bhatia, S.K., Deogun, J.S.: Conceptual clustering in information retrieval. IEEE Trans. Syst. Man Cybern. B Cybern. **28**(3), 427–436 (1998)
11. Bradley, P.S., Mangasarian, O.L., Street, W.N.: Clustering via concave minimization. In: Advances in Neural Information Processing Systems, pp. 368–374 (1997)
12. Caliñski, T., Harabasz, J.: A dendrite method for cluster analysis. Commun. Stat. **3**(1), 1–27 (1974)
13. Ceccarello, M., Pietracaprina, A., Pucci, G.: Solving k-center clustering (with outliers) in MapReduce and streaming, almost as accurately as sequentially. PVLDB **12**(7), 766–778 (2019)
14. Celebi, M.E., Kingravi, H.A., Vela, P.A.: A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert Syst. Appl. **40**(1), 200–210 (2013)
15. Coggins, J.M., Jain, A.K.: A spatial filtering approach to texture analysis. Pattern Recognit. Lett. **3**(3), 195–203 (1985)
16. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-1**(2), 224–227 (1979)
17. Dunn, J.C.: Well-separated clusters and optimal fuzzy partitions. J. Cybern. **4**(1), 95–104 (1974)
18. Elkan, C.: Using the triangle inequality to accelerate k-means. In: Proceedings of the Twentieth International Conference on Machine Learning, pp. 147–153 (2003)
19. Forgy, E.: Cluster analysis of multivariate data: efficiency versus interpretability of classification. Biometrics **21**(3), 768–769 (1965)
20. Fränti, P., Sieranoja, S.: How much can k-means be improved by using better initialization and repeats? Pattern Recognit. **93**, 95–112 (2019)
21. Frigui, H., Krishnapuram, R.: A robust competitive clustering algorithm with applications in computer vision. IEEE Trans. Pattern Anal. Mach. Intell. **21**(5), 450–465 (1999)
22. Fritz, M., Behringer, M., Schwarz, H.: Quality-driven early stopping for explorative cluster analysis for big data. SICS Softw.-Intensive Cyber-Phys. Syst. **34**(2–3), 129–140 (2019)
23. Fritz, M., Behringer, M., Schwarz, H.: LOG-means: efficiently estimating the number of clusters in large datasets. Proc. VLDB Endow. **13**(11), 2118–2131 (2020)
24. Fritz, M., Schwarz, H.: Initializing k-means efficiently: Benefits for exploratory cluster analysis. In: Proceedings of OnTheMove Federated Conferences and Workshops (OTM), 27th International Conference on Cooperative Information Systems (CoopIS 2019), vol. 11877 LNCS, pp. 146–163. Springer, Cham (2019)
25. Fritz, M., Tschechlov, D., Schwarz, H.: Efficient exploratory clustering analyses with qualitative approximations. In: International Conference on Extending Database Technology (EDBT) (2021)
26. Garey, M.R., Johnson, D.S., Witsenhausen, H.S.: The complexity of the generalized Lloyd-max problem. IEEE Trans. Inf. Theory **28**(2), 255–256 (1982)
27. Guan, H., Ding, Y., Shen, X., Krim, H.: Reuse-centric k-means configuration. In: IEEE 34th International Conference on Data Engineering, ICDE 2018, pp. 1228–1231. Institute of Electrical and Electronics Engineers Inc. (2018)
28. Hamerly, G.: Making k-means even faster. In: Proceedings of the 2010 SIAM International Conference on Data Mining, pp. 130–140 (2010)
29. Hamerly, G., Drake, J.: Accelerating Lloyd's algorithm for k-means clustering. In: Partitional Clustering Algorithms, pp. 41–78. Springer (2015)
30. Hamerly, G., Elkan, C.: Learning the k in k-means. Adv. Neural. Inf. Process. Syst. **17**, 1–8 (2004)
31. Hochbaum, D.S., Shmoys, D.B.: A best possible heuristic for the k-center problem. Math. Oper. Res. **10**(2), 180–184 (1985)
32. Hu, J., Ray, B.K., Singh, M.: Statistical methods for automated generation of service engagement staffing plans. IBM J. Res. Dev. **51**(3–4), 281–293 (2007)
33. Huang, D., Wang, C.D., Lai, J.H.: Locally weighted ensemble clustering. IEEE Trans. Cybern. **48**(5), 1460–1473 (2018)
34. Huang, D., Wang, C.D., Peng, H., Lai, J., Kwoh, C.K.: Enhanced ensemble clustering via fast propagation of cluster-wise similarities. IEEE Trans. Syst. Man Cybern.: Syst. (2018)
35. Huang, Z.: A fast clustering algorithm to cluster very large categorical data sets in data mining. Research Issues on Data Mining and Knowledge Discovery, pp. 1–8 (1997)
36. Iwayama, M., Tokunaga, T.: Cluster-based text categorization: a comparison of category search strategies. In: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 273–280. Association for Computing Machinery (ACM) (1995)
37. Jain, A.K.: Data clustering: 50 years beyond K-means. Pattern Recognit. Lett. **31**(8), 651–666 (2010)
38. Jain, A.K., Dubes, R.C.: Algorithms for Data Clustering. Prentice Hall, London (1988)
39. Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., Wu, A.: An efficient k-means clustering algorithm: analysis and implementation. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 881–892 (2002)
40. Lloyd, S.P.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
41. Macqueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297 (1967)
42. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
43. Mexicano, A., Rodríguez, R., Cervantes, S., Montes, P., Jiménez, M., Almanza, N., Abrego, A.: The early stop heuristic: a new convergence criterion for K-means. In: AIP Conference Proceedings, vol. 1738 (2016)
44. Pelleg, D., Moore, A.: X-means: extending K-means with efficient estimation of the number of clusters. In Proceedings of the 17th International Conference on Machine Learning, pp. 727—-734 (2000)
45. Punj, G., Stewart, D.W.: Cluster analysis in marketing research: review and suggestions for application. J. Mark. Res. **20**(2), 134–148 (1983)
46. Rousseeuw, P.J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**(C), 53–65 (1987)
47. Schwarz, G.: Estimating the dimension of a model. Ann. Stat. **6**, 461–464 (1978)
48. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)

49. Sugar, C.A., James, G.M.: Finding the number of clusters in a dataset: an information-theoretic approach. J. Am. Stat. Assoc. **98**(463), 750–763 (2003)
50. Thorndike, R.L.: Who belongs in the family? Psychometrika **18**(4), 267–276 (1953)
51. Tibshirani, R., Walther, G., Hastie, T.: Estimating the number of clusters in a data set via the gap statistic. J. R. Stat. Soc. Ser. B: Stat. Methodol. **63**(2), 411–423 (2001)
52. Tschechlov, D., Fritz, M., Schwarz, H.: AutoML4Clust: efficient AutoML for clustering analyses. In: International Conference on Extending Database Technology (EDBT) (2021)
53. Wu, X., Kumar, V., Ross, Q.J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., Zhou, Z.H., Steinbach, M., Hand, D.J., Steinberg, D.: Top 10 algorithms in data mining. Knowl. Inf. Syst. **14**(1), 1–37 (2008)
54. Zhang, L., Guan, H., Ding, Y., Shen, X., Krim, H.: Reuse-centric k-means configuration. Inf. Syst. **100**, 101787 (2021)