Short Paper

Efficient Exploratory Clustering Analyses with Qualitative Approximations

Manuel Fritz, Dennis Tschechlov, Holger Schwarz
University of Stuttgart
Stuttgart, Germany
{manuel.fritz,dennis.tschechlov,holger.schwarz}@ipvs.uni-stuttgart.de

ABSTRACT

Clustering is a fundamental primitive for exploratory data analyses. Yet, finding valuable clustering results for previously unseen datasets is a pivotal challenge. Analysts as well as automated exploration methods often perform an exploratory clustering analysis, i.e., they repeatedly execute a clustering algorithm with varying parameters until valuable results can be found. k-center clustering algorithms, such as k-Means, are commonly used in such exploratory processes. However, in the worst case, each single execution of k-Means requires a super-polynomial runtime, making the overall exploratory process on voluminous datasets infeasible in a reasonable time frame. We propose a novel and efficient approach for approximating results of k-center clustering algorithms, thus supporting analysts in an ad-hoc exploratory process for valuable clustering results. Our evaluation on an Apache Spark cluster unveils that our approach significantly outperforms the regular execution of a k-center clustering algorithm by several orders of magnitude in runtime with a predefinable qualitative demand. Hence, our approach is a strong fit for clustering voluminous datasets in exploratory settings.

1 INTRODUCTION

Clustering is a fundamental primitive for exploratory tasks. Jain identified three main general purposes of clustering, which emphasize the exploratory power of clustering analyses [15]: (1) 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. (2) Grouping entities. Clustering aims to group similar entities into the same cluster. Thus, previously unseen entities can be assigned to a specific cluster. (3) Compressing data, i.e., to use clusters and their information as summary for further steps.

Due to their runtime behavior, k-center clustering algorithms, such as k-Means [16], k-Medians [6] or k-Mode [14] are commonly used [18], especially on voluminous data. However, the expected number of clusters k has to be provided prior to their execution. Particularly for previously unknown datasets, choosing this parameter is a tremendous pitfall and requires particular caution: Wrong values lead to bad results regarding the above-mentioned purposes, i.e., imprecise structurings, groupings or compressions are performed, thus making the clustering results unusable in the worst case.

In order to achieve valuable clustering results, k-center clustering algorithms are typically executed with varying values for k. This can be performed manually by analysts or in an automated manner by exploration methods, which perform an automated exploratory process in order to find valuable clustering results [9]. However, as these approaches require several complete runs of a clustering algorithm, they tend to be very time-consuming, in particular when the clustering algorithm is repeatedly executed on voluminous datasets [11].

Regarding k-Means as instantiation of a k-center clustering algorithm, the runtime for a single execution is $O(knd\omega)$ [13], where k is the number of clusters, n is the number of entities in a dataset, d is the number of dimensions, and ω is the number of clustering iterations performed. In the worst case, i.e., when performing k-Means until convergence, ω is super-polynomial regarding n [3], i.e., a single execution of the clustering algorithm on large datasets is already very time-consuming. However, for exploratory clustering analyses, where clustering results of several parameter values are of interest, analysts typically require fast, yet adequately accurate approaches that can be used to gain a fundamental understanding of the results. As we show in this paper, efficient exploratory clustering analyses are possible by controlling the number of clustering iterations in the exploration process for promising parameter values.

Many implementations, such as sklearn[1] or Spark's MLlib[2], allow to reduce the runtime of clustering algorithms by setting a fixed threshold for the number of clustering iterations. However, it is not clear how to set this threshold such that a valuable clustering result can be achieved, since this choice highly depends on dataset characteristics and the initialization of a clustering algorithm. Hence, too few clustering iterations lead to an imprecise clustering result, whereas too many clustering iterations lead to an unacceptable runtime. In this work, we introduce a novel approach to efficiently terminate k-center clustering algorithms as soon as a predefined qualitative demand is met, thus reducing the number of clustering iterations and therefore also the runtime.

Our contributions include the following:

- We propose our novel approach to terminate k-center clustering algorithms based on a predefined qualitative demand, which is typically defined by analysts.
- We show, that our approach (i) provides negligible runtime overhead for its calculations, and (ii) can be seamlessly integrated into exploratory clustering analyses.
- The results of our comprehensive evaluation on a distributed Spark cluster unveil that our approach outperforms a regular execution of a k-center clustering algorithm with speedups of several orders of magnitude, while the given qualitative demand is met in most cases.

The remainder of this paper is structured as follows: We present related work in Section 2. In Section 3, we analyze advantages and pitfalls of a closely related approach to reduce the number of clustering iterations. Subsequently, we present our novel generic qualitative approximation approach for k-center clustering algorithms in Section 4. In Section 5, we summarize the results of a comprehensive evaluation unveiling the benefits of our method. Finally, we conclude this work in Section 6.

[1] https://git.io/Jt8lJ [2] https://git.io/Jt8lt

2 RELATED WORK

All k-center clustering algorithms proceed in an iterative manner, i.e., the same sequence of steps is repeated until a given convergence criterion is met. Each clustering iteration comprises the following three steps: (1) initialize or change the position of the centroids, which are centers of gravity for a specific cluster, (2) improve the clustering by (re-)assigning entities to the closest centroid, and (3) check for convergence. Eventually, many clustering algorithms terminate when no entities change their membership anymore. As mentioned above, k-Means as concrete instantiation of such an algorithm has the runtime complexity of $O(knd\omega)$ [13]. Since clustering results for several values for k are at the core of exploratory processes, we focus on related work, which addresses the remaining influencing factors.

In order to reduce the number of entities n in a dataset, sampling or coresets [4] 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 the dataset [1].

Regarding the number of clustering iterations ω, there are three categories of related work, which address the internals of a clustering iteration: (a) It has been shown that the initialization of k-center clustering algorithms is crucial to reduce the number of clustering iterations [5, 10]. The initial centroids of state-of-the-art initialization techniques are close to their optimum position, therefore requiring less clustering iterations until convergence. (b) Several works address how a single clustering iteration can be accelerated, i.e., by making distance calculations faster [7] or by caching previously calculated distances [12]. (c) Undoubtedly, reducing ω is crucial since it subsumes approaches from (a) and (b). Therefore, the "check for convergence" step is of paramount interest when reducing ω and thereby reducing the runtime of the clustering algorithm.

As each iteration comprises costly distance calculations, it is practically not feasible to perform a clustering algorithm on large datasets until convergence due to the excessive runtime. Hence, 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 high runtime. A generic threshold for all datasets is not feasible, because of too many influencing factors, such as the feature space or data distribution.

3 META-LEARNING TERMINATION (MTL)

In a previous work [8], we proposed a generic meta-learning approach to terminate k-center clustering algorithms early based on an arbitrary definable qualitative demand $q \in [0; 1]$. This approach relies on a correlation between the quality of intermediate clustering results throughout several clustering iterations and corresponding clustering validity measures (CVMs). To this end, the meta-learning procedure requires an offline phase, which gathers the necessary meta-knowledge, and an online phase, which applies the meta-knowledge on previously unseen datasets.

In the offline phase, a clustering algorithm is executed with varying parameter values on datasets with different characteristics. Throughout these executions, values of selected CVMs are recorded for each clustering iteration. The resulting values of these CVMs are often not normalized, i.e., the value ranges can be unbounded. In order to make these values tangible for analysts, we introduced the notion of quality $q \in [0; 1]$. That is, q indicates the quality of the intermediate clustering result after a certain clustering iteration in contrast to the quality of the last clustering iteration, which only becomes available after convergence. Hence, the clustering quality for each clustering iteration can only be investigated in retrospective after convergence. We proposed to create a correlation between the quality q and the CVMs, e.g., with a regression function that is trained during the offline phase based on several executions.

In the online phase, the analyst defines the expected qualitative demand q for the clustering result of a previously unseen dataset. Subsequently, the above-mentioned correlation is exploited to terminate the clustering algorithm earlier than convergence, while aiming to achieve the desired qualitative demand.

We showed that considerable runtime savings are possible, while regularly meeting the qualitative demand for several CVMs. However, we face two pitfalls regarding this method: (1) In order to exploit the correlation, it is first necessary to perform the offline phase. Since the offline phase comprises several executions of a k-center clustering algorithm, the high runtimes are solely moved from the online phase to the mandatory offline phase. (2) Creating a sound correlation between the quality and the corresponding CVM is an optimization problem. Influencing factors are for example datasets and their characteristics, the selected CVM, or chosen parameter values in the offline phase. Furthermore, formalizing the correlation itself, i.e., choosing a promising correlation method, poses another optimization problem.

To the best of our knowledge, this proposed approach is nonetheless currently the only one to limit the number of clustering iterations ω based on an arbitrarily predefinable qualitative demand. In the next section, we propose a novel method to reduce the number of clustering iterations based on a qualitative demand, which avoids the two mentioned pitfalls, yet still provides a tangible notion of quality for analysts.

4 GENERIC QUALITATIVE APPROXIMATION TERMINATION (GQA)

Before detailing on our new approach, we briefly summarize the basics of k-center 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 k-center clustering algorithms is to group $\mathcal{X}$ into k disjoint clusters, such that each entity is assigned to the closest centroid $c \in C$. As this problem is NP-hard [2], several heuristics exist, which aim to approximate the solution. One of these heuristics is the k-Means algorithm [16]. The goal of k-Means is to find the set C of k centroids which minimizes the objective function in Equation 1.

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

Here, the Euclidean distance from an entity $x \in \mathcal{X}$ to the closest centroid $c \in C$ is calculated. $\phi_{\mathcal{X}}(C)$ denotes the sum of these distances over all entities in $\mathcal{X}$ and is also called sum of squared errors (SSE) for k-Means or variance for all k-center clustering algorithms. k-center algorithms minimize their notion of variance by moving these k centroids to a better position in each clustering iteration until a certain convergence criterion is met.

4.1 Intuition of our Approach

The goal of our approach is to exploit a tangible qualitative demand q in order to terminate the clustering algorithm as soon

as q is met. In contrast to MTL [8], we explicitly aim for no preparations, e.g., no prior meta-learning step.

Our novel approach GQA draws on two properties, which are valid independent of datasets and eventualities of k-center clustering algorithms, thus preserving generality.

Property 1: Monotonically decreasing variance. According to the objective function in Equation 1, the variance ϕ decreases throughout each clustering iteration, which is applicable for all k-center clustering algorithms. Manning et al. discuss this property for k-Means in detail [17], which can be transferred to other k-center clustering algorithms analogously.

Since ϕ is monotonically decreasing, we derive that the quality of the clustering is becoming better in each iteration (cf. Equation 1). 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, k-center 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 can conclude that $\sigma_i \in [1; \infty]$.

Property 2: Notion of quality. Similarly to MTL, we also draw on a qualitative demand $q \in [0; 1]$ of the approximated clustering result, yet in a different way. That is, q can be specified by an analyst, where the choice of q has impact on the clustering result: If a very accurate clustering result is of interest, q should be set larger than for exploratory purposes, where already potentially moderate results may lead to valuable insights.

Yet, the question remains how to combine the qualitative demand $q \in [0; 1]$ of the approximated clustering result with $\sigma_i \in [1; \infty]$, while avoiding time-consuming preparations. Putting both above-mentioned properties together, we can formalize our approach as follows: During each clustering iteration, σ_i (and therefore ϕ_i) has to be calculated. Subsequently, terminate the k-center 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 $1 - q$. Note, that our approach preserves generality, since both properties are generally valid regarding dataset characteristics or k-center clustering algorithms.

4.2 Algorithm

Algorithm 1 outlines how GQA can be incorporated in the generic procedure of k-center clustering algorithms. The algorithm proceeds in four steps: (1) The centroids are initialized in line 1 according to a specific initialization, e.g., random, or k-Means‖ [5]. (2) The clustering is improved, i.e., the entities are assigned to the closest centroid (cf. lines 4-6). (3) The centroids are moved to the center of the cluster. To this end, the new position of the centroids $c \in C$ and the corresponding variance for each cluster are determined according to an objective function (cf. line 9). (4) Finally, the algorithm converges in line 16.

Changes in contrast to the regular procedure of k-center clustering algorithms are depicted underlined. As the variance for each single cluster is calculated in line 9, the overall variance for the current iteration ϕ_i is the sum of these individual values. Subsequently, σ_i is calculated in line 13, if a previous iteration was already performed. It is necessary to check this state, since our approach draws on the change of ϕ between two subsequent

Algorithm 1: k-center clustering algorithms with GQA.

Input: $\mathcal{X}$ - dataset, k - number of clusters, q - qualitative demand
Output: $\mathcal{K}$ - a combination ($\circ$) between $\mathcal{X}$ and the assigned centroid for each entity

```
/* initialize centroids                                          */
```
1 $C \leftarrow$ initialize a set of k centroids;
2 $i \leftarrow 0$;
3 **repeat**
```
    /* improve clustering                                        */
```
4 **for** $\forall x \in \mathcal{X}$ **do**
5 $\mathcal{K} \leftarrow \{x \circ c\}$, where c denotes the closest centroid to x according to an objective function;
6 **end**
7 $\phi_i(C) \leftarrow 0$;
```
    /* change centroids                                          */
```
8 **for** $\forall c \in C$ **do**
9 $c, \phi(c) \leftarrow$ new c and its corresponding variance according to an objective function, where $\{x \circ c\} \in \mathcal{K}$;
10 $\phi_i(C) \leftarrow \phi_i(C) + \phi(c)$;
11 **end**
12 **if** $i > 0$ **then**
13 $\sigma_i \leftarrow \phi_{i-1}(C)/\phi_i(C)$;
14 $\phi_{i-1}(C) \leftarrow \phi_i(C)$;
15 $i \leftarrow i + 1$;
16 **until** $i > 1$ *and* $1 - q \geq \sigma_i - 1$; `// check for convergence`
17 **return** $\mathcal{K}$;

iterations. After σ_i is calculated, ϕ is adjusted properly for the next iteration, i.e., ϕ of the previous iteration is set as ϕ of the current iteration. Finally, the convergence criterion is set according to Inequality 2 in line 16. Note however, that this convergence criterion can only be met after at least 2 clustering iterations are performed, since σ_i draws on the variance ϕ of two subsequent iterations. Hence, we introduce the additional check for $i > 1$ in the convergence criterion in line 16.

4.3 Analysis and Discussion

The goal of our approach is to keep additional calculations as cheap as possible. As already mentioned, the computations required by our approach are underlined in Algorithm 1. The computations rely on (a) allocations of variables (lines 7, 10, 13 and 14), (b) comparisons (lines 12 and 16), as well as (c) arithmetic operations (lines 10, 13, 16). Allocations and comparisons can be performed in $O(1)$. For the arithmetic operations, we reuse byproducts of the k-center clustering algorithm, such as ϕ_i. These values are used throughout several iterations with simple additions (line 10) or divisions (line 13). Therefore, the runtime complexity for the arithmetic operations is $O(1)$. Concluding, the overall runtime complexity of our approach is $O(1)$, thus preserving the runtime complexity of a k-center clustering algorithm without a significant runtime overhead.

In contrast to MTL, our novel approach GQA (i) can be used for ad-hoc exploratory clustering analyses, since it does not rely on a time-consuming meta-learning step, (ii) preserves generality regarding dataset characteristics, which may not be the case for MTL, since it obeys an optimization problem (cf. Section 3), and (iii) directly addresses the objective function of k-center clustering algorithms instead of additional CVMs as MTL does. As discussed above, addressing the objective function can be

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

Table 1: 27 synthetic datasets for the evaluation.

Dataset	Name	n	d	c
i	Skin segmentation	245,057	3	2
ii	Poker hand	1,025,010	10	10
iii	Individual household electric power consumption	2,049,280	7	-
iv	US census data (1990)	2,458,285	68	-
v	KDD Cup 1999 data	4,898,431	33	23
vi	SUSY	5,000,000	18	2
vii	Gas sensor array under dynamic gas mixtures	8,386,765	19	-
viii	HEPMASS	10,500,000	28	2
ix	HIGGS	11,000,000	28	2
x	Heterogeneity activity recognition	33,741,500	5	6

Table 2: 10 real-world datasets as used in the work of MTL [8], where c denotes #classes, if available.

done very efficiently, whereas using an additional CVM in each clustering iteration may require noticeable runtime overhead [8].

Furthermore, it should be emphasized that GQA can be easily used by analysts and automated exploration methods due to its striking resemblance to the regular procedure of k-center clustering algorithms. Because solely q should be defined, the additional complexity of our novel approach is clearly manageable.

5 EVALUATION

In our evaluation, we investigate the benefits of our proposed approach for exploratory clustering analyses. To this end, we will compare our novel approach GQA with its closest competitor MTL [8] and a regular execution of a k-center clustering algorithm. We present the setup for our evaluation, before presenting the runtime and quality results.

5.1 Experimental Setup

Hardware and Software. We conducted all of our experiments on a distributed Apache Spark cluster. This cluster 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, Hadoop 3.2.0 as well as Spark 2.4.0.

Synthetic Datasets. We implemented a synthetic dataset generator in order to perform a systematic evaluation with controlled dataset characteristics. This tool generates datasets based on the following input parameters: The number of entities in the dataset (n), the number of dimensions (d) and the number of clusters in a dataset (c), 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 randomly chosen and the clusters are non-overlapping. Table 1 depicts the characteristics of the 27 synthetic datasets.

Real-World Datasets. We use the same datasets as in our prior work for MTL [8], which are publicly available[3]. Table 2 summarizes the characteristics of these 10 datasets. Note, that not all of them have class labels, i.e., the number of classes is unknown for some datasets (indicated by "-" in Table 2).

Implementation. We base our implementation on Apache Spark. We focus on k-Means as instantiation of a k-center clustering algorithm due to its overwhelming popularity [18]. To this end, we implemented several methods to terminate k-Means.

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

The baseline (BASE) for this experiment is Spark's MLlib implementation of k-Means. This implementation uses k-Means‖ [5] for the initialization step and terminates after 20 clustering iterations at most. We explicitly remove the threshold for the number of clustering iterations, since we want to highlight the differences in quality when performing k-Means until convergence.

For MTL, it should be noted that this obeys an optimization problem regarding (i) the used datasets and their characteristics, and (ii) the choice of the used correlation technique. However, for synthetic datasets, we have closely followed the approach described in our previous work [8]. For the offline phase, we used datasets from Table 1 where $d = 50$. We clustered these datasets with k in 11 equidistant values in $[2; 2c]$ and let each clustering run until Spark's convergence criterion is met. We performed three runs per value of k. For each clustering iteration, we measured the SSE as well as the separation (SEP) between the centroids, since we achieved the best results with the SEP metric in our previous work [8]. Subsequently, we trained a second-degree polynomial regression between the change rate of SEP in contrast to the relative error of the SSE regarding the current clustering iteration and the final clustering iteration. The whole process took 11.17 hours and is thus not feasible for ad-hoc exploratory clustering analyses. Regarding the real-world datasets, we use the same ones as in [8], i.e., we also use the same regression function. The meta-learning process on those real-world datasets required several days, which emphasizes the impractical runtime for the offline phase of MTL. Furthermore, we implemented our novel GQA approach as described in Section 4.

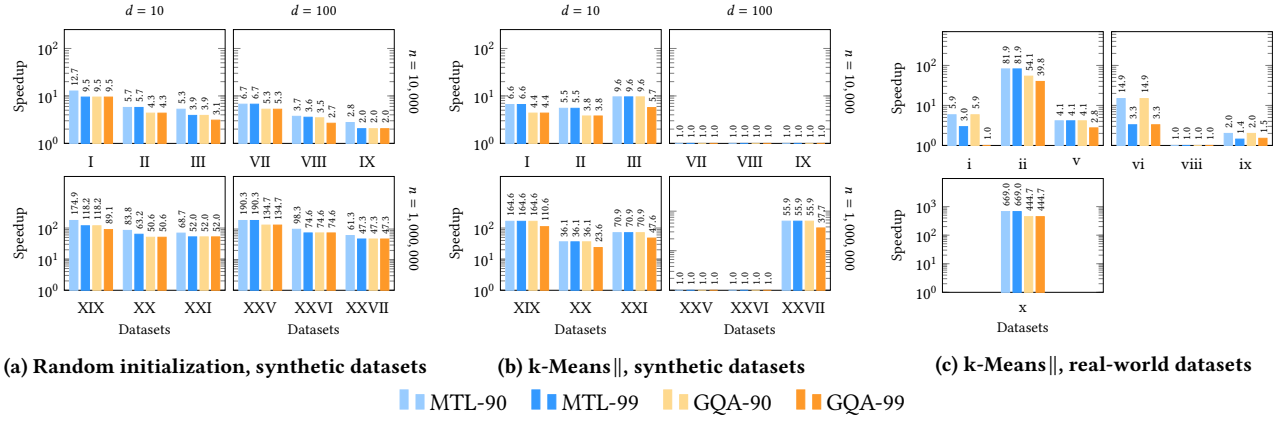
For MTL and GQA, we set the respective qualitative demands to 90 % and 99 % in order to achieve valuable results. Hence, we compare Spark's implementation (BASE) to MTL-90, MTL-99, GQA-90 and GQA-99 on synthetic and real-world datasets.

Furthermore, we use different initialization techniques in order to investigate the differences in the results. We run k-Means with $k = c$ for each dataset, where c is known. For each method, we performed three runs and present median values.

5.2 Runtime Results

Figure 1 summarizes the results regarding the speedups in contrast to the baseline, where Figure 1a focuses on the results with random initialization and Figure 1b addresses the results with the initialization via k-Means‖. While both of them show the results on synthetic datasets, Figure 1c unveils the results with k-Means‖ initialization on real-world datasets.

It can be seen that for random initialization, speedups can be achieved for all datasets in contrast to the initialization via k-Means‖. Since k-Means‖ initializes centroids closer to their optimum, less clustering iterations are necessary than for random



(a) Random initialization, synthetic datasets **(b) k-Means‖, synthetic datasets** **(c) k-Means‖, real-world datasets**

Figure 1: Speedup of MTL and GQA in contrast to BASE for all synthetic and real-world datasets where $k = c$.

initialization. Hence, MTL and GQA lead to remarkable speedups for random initialization of up to 190.3 (MTL) and 134.7 (GQA).

On the other hand, k-Means‖ provides an $O(log\ k)$-approximation to the final clustering result [5]. Therefore, some speedups of 1 are observed, i.e., BASE terminates k-Means as early as MTL and GQA. However, as shown in Figure 1b, MTL and GQA still provide strong speedups in many cases, such as for datasets XIX and XXVII. We observe speedups of roughly up to 164.6. When comparing dataset IX with dataset XXVII, it is evident that the latter has 100× more entities, where the remaining dataset characteristics, such as d and c, remain unchanged. As k-Means‖ does not select promising initial centroids for dataset XXVII, several clustering iterations are necessary for BASE. Yet, MTL and GQA terminate k-Means earlier than convergence, thus achieving significant speedups. More generally speaking, k-Means‖ can only select promising initial centroids, if the underlying dataset characteristics and number of clusters are a strong fit for its procedure.

Regarding the results of the real-world data (cf. Figure 1c), we make very similar observations. Here, even more significant speedups of up to 669.0 (MTL) and 444.7 (GQA) can be observed for the largest real-world dataset x.

In general, MTL and GQA achieve similar speedups, however MTL is faster in some cases. Note, that MTL addresses the SEP, whereas GQA addresses the variance (= SSE for k-Means) in order to approximate clustering results. Since the SEP typically changes more significantly in the first few iterations than the SSE, we argue that GQA requires a few additional clustering iterations.

Regarding the different qualitative demands, we observe only small deviations in the resulting speedups. These differences mostly occurred on real-world datasets (cf. Figure 1c). Here, the higher qualitative demand of 99 % requires more iterations and therefore leads to lower speedups.

We conclude that both approaches are well-suited for rather voluminous datasets, since k-center clustering algorithms require more clustering iterations on these datasets until convergence, which is explicitly addressed by MTL and GQA.

5.3 Quality Results

Since MTL and GQA are able to speedup the execution of a clustering algorithm significantly, the question remains how these approximations affect the clustering quality.

As clustering aims to provide compact and well-separated clusters, we focus on the compactness and the separation (SEP) of the centroids. We use the sum of squared errors (SSE) as instantiation of the compactness, since the smaller the SSE for k-Means, the less variance in the clusters, i.e., the more compact are the clusters. For a better comparison, we focus on the relative error δ of SSE and SEP of the clustering results of MTL and GQA in contrast to the baseline, i.e., the smaller the better.

Figure 2 summarizes the results for MTL and GQA. Note the different y-axes throughout the figures. The results unveil that the qualitative demands of 90 % and 99 % are mostly met in average for MTL and GQA. It should be noted, that the qualitative demand of GQA addresses the SSE (left), whereas the qualitative demand of MTL addresses the SEP (right). A more detailed investigation of the results unveiled that for all synthetic datasets, the respective qualitative demands are always met. Regarding the real-world datasets, MTL and GQA rarely terminated the clustering algorithm too early, thus omitting better clustering results. This happened for both approaches only once for a qualitative demand of 90 % and twice for a qualitative demand of 99 %. This observation supports the practical feasibility of our novel approach GQA, i.e., addressing the decreasing trend of σ_i leads to satisfying clustering results in practice.

The quality of the clustering results achieved by MTL and GQA differ only marginally from the baseline, i.e., the regular execution of k-Means. Furthermore, the initialization via k-Means‖ leads to more compact and better separated clustering results than initializing at random, since the values for δSSE and δSEP are mostly smaller (cf. Figure 2a and b). Hence, this initialization is better suited for exploratory clustering analysis, since it allows more correct insights into clustering results.

Moreover, the results achieved by GQA are always better than the corresponding pendant of MTL in terms of δSSE and δSEP. The reason can be found in the additional clustering iterations that GQA performed in contrast to MTL (cf. Section 5.2).

Concluding, MTL and GQA perform similar in terms of runtime, yet GQA does not rely on a previously conducted offline phase for meta-learning. Remember, that MTL required several executions of a k-center clustering algorithm on several datasets, which required more than 11 hours for synthetic datasets and several days for real-world datasets in our scenario. In addition, MTL and GQA lead to compact and well-separated clustering results. However, GQA achieves better clustering results, because

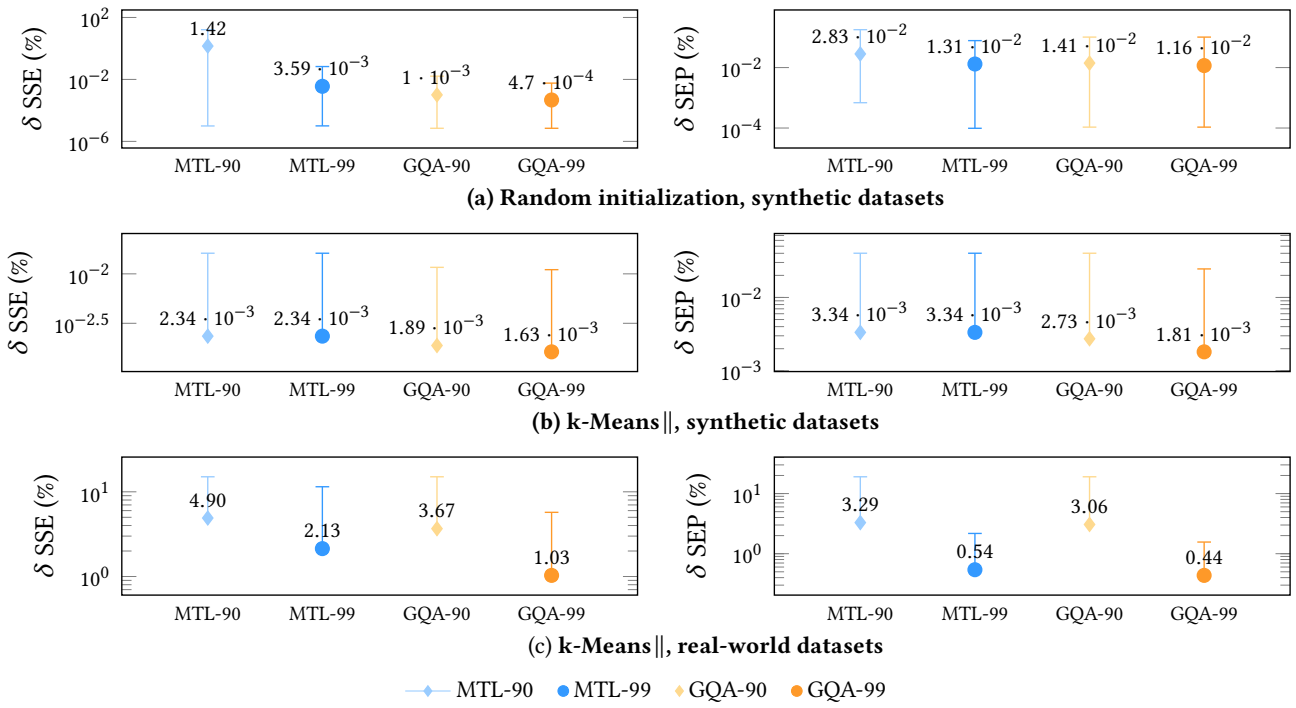


Figure 2: Relative error δ of SSE and SEP compared to the baseline. Average values are depicted per error bar. Smaller values indicate more similar clustering results w.r.t. the baseline, i.e., they show better clustering results.

it typically requires a few more clustering iterations than MTL. Therefore, our novel approach GQA is well-suited for ad-hoc exploratory clustering analyses, especially on voluminous datasets, since it provides very accurate results in a short time period.

6 CONCLUSION

In this work, we proposed a novel approach to terminate k-center clustering algorithms as soon as a predefined qualitative demand of the clustering results is met. Our approach aims to trade off quality of clustering results, while achieving them in a short time frame. We showed that our approach is generic, i.e., it can be used with several k-center clustering algorithms and initialization strategies. In our comprehensive evaluation, we unveiled that our approach significantly outperforms state-of-the-art executions of k-center clustering algorithms in terms of runtime, yet achieves very similar clustering results. Therefore, it is of particular interest for exploratory clustering analyses. Future work will address to what extent automated exploration methods benefit from our novel approach.

ACKNOWLEDGMENTS

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.

REFERENCES

[1] Amirali Abdullah, Ravi Kumar, Andrew McGregor, Sergei Vassilvitskii, and Suresh Venkatasubramanian. 2016. Sketching, embedding, and dimensionality reduction for information spaces. In *AISTATS 2016*, Vol. 41. 948–956.

[2] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. 2009. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning* 75, 2 (may 2009), 245–248.

[3] David Arthur and Sergei Vassilvitskii. 2006. How slow is the k-means method?. In *Proceedings of the Annual Symposium on Computational Geometry*.

[4] Olivier Bachem, Mario Lucic, and Andreas Krause. 2018. Scalable k-means clustering via lightweight coresets. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1119–1127.

[5] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. 2012. Scalable K-Means++. *PVLDB* 5, 7 (2012), 622–633.

[6] P S Bradley, O L Mangasarian, and W. N. Street. 1997. Clustering via concave minimization. In *Advances in Neural Information Processing Systems*. 368–374.

[7] Charles Elkan. 2003. Using the Triangle Inequality to Accelerate k-Means. *International Conference on Machine Learning* (2003), 147–153.

[8] Manuel Fritz, Michael Behringer, and Holger Schwarz. 2019. Quality-driven early stopping for explorative cluster analysis for big data. *SICS Software-Intensive Cyber-Physical Systems* 34, 2-3 (jun 2019), 129–140.

[9] Manuel Fritz, Michael Behringer, and Holger Schwarz. 2020. LOG-Means: Efficiently Estimating the Number of Clusters in Large Datasets. *PVLDB* 13, 11 (2020), 2118 – 2131.

[10] Manuel Fritz and Holger Schwarz. 2019. Initializing k-means efficiently: Benefits for exploratory cluster analysis. In *Lecture Notes in Computer Science*, Vol. 11877 LNCS. Springer, 146–163.

[11] Manuel Fritz, Dennis Tschechlov, and Holger Schwarz. 2020. Learning from past observations: Meta-learning for efficient clustering analyses. In *Lecture Notes in Computer Science*, Vol. 12393 LNCS. Springer, 364–379.

[12] Greg Hamerly. 2010. Making k-means even faster. In *Proceedings of the 2010 SIAM international conference on data mining*. 130–140.

[13] Greg Hamerly and Jonathan Drake. 2015. Accelerating Lloyd's Algorithm for k-Means Clustering. In *Partitional Clustering Algorithms*. Springer, 41–78.

[14] Zhexue Huang. 1997. A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining. *Research Issues on Data Mining and Knowledge Discovery* (1997), 1–8.

[15] Anil K. Jain. 2010. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters* 31, 8 (jun 2010), 651–666.

[16] James B. Macqueen. 1967. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* 1 (1967), 281–297.

[17] Christopher D. Manning, Prabhakar. Raghavan, and Hinrich. Schütze. 2008. *Introduction to information retrieval.* Cambridge University Press. 482 pages.

[18] Xindong Wu, Vipin Kumar, Quinlan J. Ross, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S Yu, Zhi Hua Zhou, Michael Steinbach, David J Hand, and Dan Steinberg. 2008. Top 10 algorithms in data mining. *Knowledge and Information Systems* 14, 1 (2008), 1–37.