Apache Spark Implementation of the Constrained K-Means Clustering Algorithm

Nguyen Quang Huy[1], Vu Thu Diep[2], and Phan Duy Hung[1(✉)]

[1] FPT University, Hanoi, Vietnam
huy23mse13113@fsb.edu.vn, hungpd2@fe.edu.vn
[2] HaNoi University of Science and Technology, Hanoi, Vietnam
diep.vuthu@hust.edu.vn

Abstract. The essential task of extracting useful patterns from large datasets enables multiple industries to implement data-driven decisions throughout the current significant data era. Unsupervised machine learning contains clustering as one vital technique which identifies data similarities through shared common features to produce groupings. Traditional efficient algorithms such as K-Means struggle to handle complex datasets because distance-based relationships do not describe all pertinent data associations. The proposed study implements Constrained K-Means because its algorithm utilizes must-link and cannot-link constraints to preserve domain-specific relationships in cluster groupings. We conducts tests of the proposed algorithm through Apache Spark platform due to its ability to maximize distributed processing capabilities for large-scale datasets. Experimental results demonstrate that the Constrained K-means algorithm achieves better clustering performance and interpretability and marks superior results compared to K-means when dealing with constraint-aware data sets on synthetic and real-world datasets. The studied combination of constraint-based clustering and distributed computing systems demonstrates potential to create stronger yet scalable machine learning solutions for big data environments.

Keywords: Clustering · Constrained Clustering · Big Data Clustering · Apache Spark

1 Introduction

1.1 Problem and Motivation

In the field of exploratory data analysis in many fields solving the task of the unsupervised classification of patterns into clusters, which is of interest and of wide utility [1]. Generally, clustering aims to divide a set of patterns to the clusters such that patterns remain within clusters more than patterns within the different clusters. This methods uses wordless patterns, and provide them data and then gives a category label to this patterns. Clustering is a very general activity, which is used wherever we use pattern analysis, grouping, decision making or even machine learning in data mining, document retrieval, image segmentation, etc.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15886, pp. 212–225, 2026.
https://doi.org/10.1007/978-3-031-97576-9_14

However, there are four general types of traditional clustering methods [2], including Partitioning methods, Hierarchical Clustering, Density-based Clustering, and Model based clustering. Yet, these existing methods are commonly used across multiple disciplines due to their lack of capability to utilize prior domain knowledge, or to incorporate specific form of data restrictions. There are many real life cases in which there is extra information or prior knowledge which provides extremely important addition to the problem of demand, forcing researchers to come with methods of clustering processes that take the constraints into account as a part of the clustering [3].

Constrained clustering has emerged as a powerful data grouping methodology since it puts user defined constraints to assign the partitioning process in accounting with more contextually relevant, use case specific credible values of clustering. Previous studies have shown that the much more domain specific clusters can be achieved leveraging prior information in the form of must link and cannot link constraints [3]. In the past years, various constrained clustering algorithms have been introduced in order to enhance the clustering performance on structured and semi structured data. Specific among these approaches are COP-KMeans [4], MPCK-Means [5], and PCCC (Pairwise-Confidence-Constraints-Clustering) [6] as they all have distinguished methods for solving constraints within the cluster improvement strategy. However, by applying constraint based guidance in the clustering process, these algorithms provide improved accuracy and flexibility in a dataset where relationship among data entities contributes to define the effective cluster structure in these datasets.

Nevertheless, traditional constrained clustering algorithms have proved to be scalable only for datasets that exponentialally increase [3]. There are requirements for processing enormous amounts of data such that applications often involve big data and real time processing and the related frameworks are Hadoop and Spark in such applications. It also allows running algorithms in a matter of milliseconds, better resource usage, and scales to millions of records [7]. In this work, we deal with Constrained K-Means, a constrained clustering algorithm which is adapted in such a way to make it an efficient algorithm for a big data setting, so that it can be applied to a large number of large real world problems.

1.2 Related Works

The conventional K-Means [8] is a popular clustering algorithm, where K represents the number of clusters that are found by minimizing the sum of the square of distances between the data points and the cluster centroid. This will conclude the first step of our entire iterative process initialization, assignment, and centroid update until convergence. But K-Means is not without its drawbacks, despite being simple and efficient. Many issues arise like it is sensitive to initial placement of centroids, it has difficulties with non-spherical clusters, and it has not taken constraints into account which eliminates its usage in structured clustering problems [9].

We conclude by identifying how each of these limitations has been collectively addressed in the successive evolution of constrained clustering algorithms. Figure 1 illustrates in a simple way how constrained clustering works.

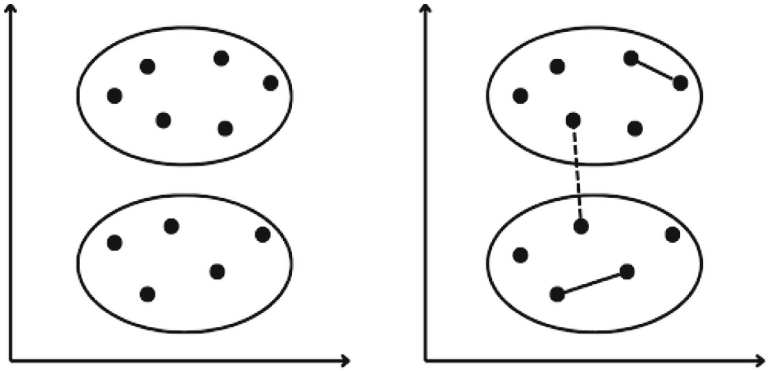

Fig.1. Hard pairwise constraints: must-link and cannot-link constraints are indicated by solid and dashed lines respectively.

A noticeable foundational approach is COP-Kmeans from Wagstaff et al. (2001) [4]. The traditional KMeans algorithm is extended with pairwise constraints, or must link and cannot link relationships that specify if instance pairs should be in the same cluster or in different clusters, with the consequent addition of the COP-Kmeans algorithm. In this more refined approach, it is required that this constraint is strictly enforced in the assignment phase such that an instance is assigned to a cluster only if none of the constraints are violated by it. Although this strict constraint satisfaction guarantees that constraints are satisfied when the constraints are dependable, it may lead to the algorithm being unable to converge to a feasible or optimal solution when the constraints contradict, conflict with the underlying data distribution. Additionally, COP-Kmeans is very sensitive to noisy constraints, which implies just a few erroneously placed constraints can adversely affect clustering performance as much as an order of magnitude. However, COP-Kmeans is still useful when constraint satisfaction is essential, especially if background knowledge is trusted.

Advances constrained clustering is based on metric learning within K-Means concepts that were built upon ideas by Bilenko, Basu, Mooney (2004) [5]. An innovative integration of such distance metric with unlabeled data and pairwise constraints allows the algorithm to adapt such metric as a function of unlabeled data and pairwise constraints. The main idea consists in warping the data space so as to force closer the observations in any same cluster and at the same time keep the distances between different clusters. MPCK-Means iteratively updates distance metric in clustering, and modifies it to assure better constraints satisfaction and better cluster cohesion. It also learns metrics per cluster and is able to accommodate different shapes and orientations of the data which is especially important for the data distribution with more complicated shapes and orientations. MPCK-Means finds better accuracy and robustness to noisy constraints under learned metrics than COP-KMeans, by minimizing an objective function that penalizes constraint violation for a small coefficient, while encouraging lower cluster dispersion by the other coefficient under the learned metrics. On the other hand, the iterative metric learning process brings more computational complexity in high dimension dataset.

By applying it to hard and soft constraints, with associated confidence values,a cellular Pairwise-Confidence-Constraints-Clustering, PCCC as it is called in [6] by Baumann and Hochbaum, provides an elegant way to increase the flexibility and scalability of constrained clustering. Both types of constraints are trained by PCCC in modeling object assignment problem as an integer program to minimize the within cluster sum of squares. The hard constraints that need to be satisfied are separated from the soft constraints on which we have penalties for violation with this framework. Every soft constraint has associated confidence value that multiplies violation penalty, weighting more reliable constraints. Furthermore, PCCC reduces model size by moving nodes to clusters of their q's nearest, while PCCC also adds to its moves for nodes around cluster centres a variant of multi start approach. The flexibility and scalability of the algorithm is expanded in all of these approaches as PCCC can be employed in an area where the reliability of the constraint is a very very loose or very tight. PCCC is in general computationally expensive, but model size reduction strategies are employed to reduce this cost, and PCCC is thus a promising constrained clustering task solver.

As the data continues to grow rapidly in the modern application, there is a demand for scalable clustering methods that are able to manage large scale datasets. Due to its iterative structural nature, traditional clustering (and constrained clustering) methods, such as K-means, suffer from high computational demands (especially in the case of high dimensions and vast numbers of features) [10]. In today's Big data world, Apache Spark has come up as one of key distributed computing platform with the preferred in memory computation model that is also fault tolerant. Spark [7] defines the Resilient Distributed Dataset (RDD) abstraction which provides the ability to execute parallel jobs and Spark MLlib [11] provides the implementation of several machine learning algorithms such as clustering, which can scale. K-Means|| [12] which builds upon K-Means + + [9] to embed parallelism into the initialization phase and reducing the number of sequential passes, is the fastest among distributed approaches and is efficient in case of large datasets. Nonetheless, still Kmean remains susceptible to issues resulting from a sensitive centroid initialization as well as problem associated with identifying clusters that are not convex or uneven sized, primarily, we see the development of some adaptive clustering methods in Spark. To address these problems, thus various enhanced clustering algorithms have been integrated in the Spark's ecosystem. For example, Jędrzejowicz et al. (2016) [14] provided a Fuzzy C-Means classifier that has been parallelized by the use of RDD based computations to obtain better scalability and efficiency, and distance measures among the kernel and eigenvalue based cluster detection to have better performance at large scales. Parallel to that, Gaussian Mixture Model (GMM) is used in distributed soft clustering, and is applied to pair with entropy driven feature selection to analyze the spread of Zika virus [16] to deal with large dataset and to find epidemiological results. Han et al. (2018) [15] uses a scalable parallel DBSCAN on Spark with improvements to overcome the problems of fault tolerance (e.g., job rescheduling), workload balancing, and merging overheads exposed in MPI or Hadoop. To perform rapidly neighbor searches the authors use kd-tree, the method based on partitioning that balances sub-domains, and a new merging framework, that logically combines partial cluster, showing obvious scaling advantages for wide ranged clustering tasks. Taken together, they show that Apache Spark is a workable and flexible distributed computing

platform that is also scalable and capable of high performance in a number of data intensive domain.

Spark-based clustering has been advanced well, but it does not have the mechanisms to meet must-link or cannot-link constraints. Such major drawbacks hinder the applicability into areas where users need to retain existing relations or structured relationships. A comprehensive survey of existing studies shows that existing methods either do not address the problem of integrating constraints into K-Means under the Spark framework, or they only address part of the problem, hence indicating a significant research gap to fill. Therefore this work proposes a Constrained K-Means algorithm that runs within Spark, where must-link and cannot-link constraints can be embedded in the clustering process. The focus of this study will be centered on augmenting K-Means by the ability of constraint enforcements but keeping the scalability and efficiency of Spark and the interpretability as well as the involvement of domain experts.

1.3 Contribution

In this work, we propose and implement a Constrained K-Means clustering algorithm over Apache Spark framework for processing large scale data with the aim to process the data in the most effective way while taking care of processing the must-link and cannot-link constraints. In this study, we develop such an algorithm that is scalable and retains the study criteria within Spark's MLlib framework. Upon spark distributed data structures and spark parallel processing capabilities are applied, whole things gets optimized, and spark dramatically improves the algorithm performance. Besides, the proposed Constrained K-Means algorithm is also evaluated in terms of its performance as compared to the best clustering methods considered namely K-Means and K-Means|| in respect to accuracy, convergence time and computation speed.

2 Methodology

The Constrained K-Means algorithm is created for the system and based on a structured workflow of four phases, called preprocessing, initialization, assignment and iterative constraint validation, executed on top of Apache Spark. Sections 2.1 to 2.4 give detailed explanation on each step of the proposed method, and Algorithms 1 to 4 give high level pseudocode for the method.

We have given input of n objects of d-dimensions feature vector $x_i \in R^d$ and number of clusters k assigned to each of the object. This algorithm is like the standard K-Means algorithm, but to preserve referred to by the domain knowledge, this algorithm uses must link (ML) and cannot link (CL) constraints such that some of the points stuck or stuck together. Because Spark DataFrames can be efficiently processed in parallel and optimized for query with Spark Catalyst optimizer, we use Spark DataFrames to process the underlying data. This algorithm begins with preprocessing in where superpoints is fed in to reduce computationally the number of useful data points, and after that it is computationally efficient by using in to superpoints. The super points are stored in Spark DataFrames so that data transformations can be done efficiently between Spark DataFrame and Spark context, and also cluster centroid initialization can be parallelized

with the help of Spark's distributed execution model using the K-Means|| algorithm. It is the projection of cluster assignment to original data points and being consistent with must link constraints in the third phase. In the last phase, we finally enforce the cannot link constraints using Spark DataFrame operation as an iterative constraint enforcement. For breaking cluster assignments, this algorithm will yield to this, and when a violating data point touches any constraint, it relegates it to another cluster. We have reassignments and no violation is possible at this stage, and the iteration continues to converge to that. It also uses checkpointing and caching to stop doing redundant computations.

2.1 Preprocessing

In the preprocessing phase of the Constrained K-Means, we apply must-link constraints that join connected data points by grouping them into superpoints, prior to clustering (Algorithm 1). Formally, given a dataset $D = \{x_1, \ldots, x_N\}$ and a set of must-link constraints $ML = (x_i, x_j)$ with x_i and x_j must belong to the same cluster, the algorithm represents D as a graph $G = (V, E)$ using GraphFrames [21], where each $x_i \in D$ is a node in V, and edges in $(x_i, x_j) \in E$ exists if $(x_i, x_j) \in ML$. The method applies a connected components algorithm to G, extracting point groups that needs to be from the same cluster, associating a unique root ID as a reference for each group. Its runs through all connected components and generates one superpoint for each connected component by aggregating all the member feature vectors. This extracts $\tilde{D}$, which reduces the size of the data without losing must-link constraints. These constraints are applied during preprocessing so there are no must-link violations in the subsequent steps and the algorithm is scalable to large datasets by using distributed processing.

Algorithm 1: Preprocessing	
Input: Dataset $D = \{x_1, \ldots, x_N\}$, must-link constraints ML	
Output: Reduced dataset $\tilde{D}$ with superpoints	
1.	Represent dataset as a graph $G = (V, E)$
2.	Apply connected components algorithm to G
3.	Assign a unique root ID to each connected component
4.	Aggregate feature vectors of each component (compute centroid)
5.	Construct reduced dataset $\tilde{D}$
6.	**return** $\tilde{D}$

2.2 Initialization

In the initialization step (Algorithm 2), the cluster initialization is performed over the superpoints $\tilde{D}$ that are preprocessed, and it keeps must-link constraints already present from the outset. In this case, the algorithm uses parallelized seeding to pick centroids via K-Means|| over $\tilde{D}$ with the goal of finding centroids more effectively than random initialization. The superpoints are clustered and consequentially give rise to initial cluster assignments for these superpoints. Subsequent steps will repeat this procedure in adjusted updates and constrained enforcement (including cannot-link conditions) of the clustering solution in all subsequent updates. The use of K-Means|| makes the algorithm more stable

as it improves the cluster level, such that the final clustering solution is computationally efficient while complying with must-link and cannot-link constraint.

Algorithm 2: Initialization
Input: Superpoints $\widetilde{D}$, number of clusters k
Output: Initial cluster assignments
1. K-Means|| on $\widetilde{D}$ to initialize cluster centroids
2. **return** Initial clusters

2.3 Assignment

After the initialization of clusters on superpoints, we propagate these assignments to the original data points with the guarantee that must link constraints are satisfied (Algorithm 3). Since preprocessing merged mustlinked points into superpoints, all data points in the same superpoint therefore have to inherit the cluster assignment of its root ID, it means the constraint consistency is preserved. This is made possible by mapping the assigned cluster of each superpoint to its root ID and joining with the original data, whereby each data point can be assigned independently. Because the number of superpoints is much lower than the original dataset, this is conducted as broadcast joins to avoid communication overheads in a distributed environment. In the next phase, the algorithm finishes the initial clustering step after all data points have been labeled to a cluster and checks if must-link constraints are strictly enforced; then, it proceeds to the constraint validation. An early enforcement helps ensuring convergence of the algorithm, sustaining the structural consistency, and promotes satisfaction of the constraint based clustering objectives.

Algorithm 3: Assignment
Input: Superpoints $\widetilde{D}$ with assigned clusters, original dataset D
Output: Cluster assignments for all data points
1. Map each superpoint's cluster assignment to its root ID
2. Join this mapping with the original dataset D
3. Assign each data point the cluster of its corresponding superpoint
4. Optimize using broadcast joins to minimize communication overhead
5. **return** Cluster assignments for all data points

2.4 Constraint Validation

Algorithm 4: Constraint Validation

Input: Cluster assignments, ML constraints, CL constraints.
Output: Refined cluster assignments

```
repeat
    Compute cluster centroids
    Detect cannot-link violations (same cluster for forbidden pairs)
    if no violations exist then
        Terminate process
    else
        Reassign violating points to valid clusters
    end if
until converge or max iterations reached
return Final cluster assignments
```

The iterative constraint validation step ensures that cannot-link constraints are strictly enforced by continuously refining cluster assignments until convergence (Algorithm 4). Let the dataset $D = \{x_1, \ldots, x_N\}$ be given, where each data point $x_i \in R^D$ is initially assigned a cluster label $\hat{k}(x_i) \in \{1, \ldots, K\}$. In the iterative constraint validation process, the first step is to compute the centroids of the clusters. For each cluster j (with j $=$ $1, \ldots,$ K), let

$$C_j = \{x_i \in D \mid \hat{k}(x_i) = j\}$$

denote the set of data points in cluster j. The centroid of cluster j is then calculated as

$$c_j = \frac{1}{|C_j|} \sum x_i \in C_j x_i.$$

These centroids serve as the reference points for reassigning data points that violate cannot-link constraints. The cannot-link constraints are defined by the set $CL = (x_i, x_j)$ with x_i and x_j must not belong to the same cluster.

A violation occurs when, for any pair $(x_i, x_j) \in$ CL, the condition $\hat{\mathrm{k}}(\mathrm{x}_i) = \hat{\mathrm{k}}(x_j)$ is true. For each data point x_i involved in a violation, the algorithm computes the Euclidean distance to every centroid c_m for m $\neq \hat{\mathrm{k}}(\mathrm{x}_i)$:

$$d(x_i, c_m) = \sqrt{\sum_{p=1}^{D} (x_{i,p} - c_{m,p})^2}.$$

The algorithm then selects the closest alternative cluster for $\mathrm{x_i}$ by determining:

$$r = arg \min_{m \neq \hat{k}(x_i)} \left\{ d(x_i, c_m) \mid \forall (x_i, x_j) \in CL,\ \hat{k}(x_i) \neq m \right\},$$

and updates the cluster assignment $\hat{\mathrm{k}}(\mathrm{x}_i) \leftarrow$ r. This step of reassigning is repeated iteratively; recomputing centroids, and reassigning any remaining violating points, until

there are no more cannot link violation or a predefined maximum number of iterations is reached.

It is noted that must-link constraints are already satisfied, since in preprocessing we first form superpoints to guarantee that all substances of a must-link group are assigned to the same set. As all these operations are done in distributed manner, we are rapidly converging onto probable solution to K-Means clustering problem since the algorithm can be run iteratively until the centroid values stop changing. The result is a constrain compliant clustering solution of dataset D that preserves the structure of must-link constraints and the constrain compliance, while being scalable clustering result.

3 Experiments and Results

3.1 System Environment

The experiments are conducted in MacBook Pro 14-inch equipped with Apple M1 Pro chip and 16 GB unified memory on macOS Sonoma 14.6.1. To prepare the virtual environment, Oracle VirtualBox was used to set up the virtual machine; an Ubuntu 64-bit (ARM) virtual machine that had 8GB RAM, 10 virtual processor, 50GB storage. Standalone mode of Apache Spark was deployed and used system hardware resources for the distributed data processing.

3.2 Datasets

In this study, the datasets are adopted from the work of Baumann and Hochbaum [6] which lists a broad range of classification datasets with predefined ground truth labels. These datasets span across a number of domains, such as biological, medical, and synthetic, which gives rise to different level of complexities for assessing how a method performs in clustering. Thanks to the availability of ground truth labels, one may use objective external validation metrics based on the Adjusted Rand Index (ARI) [20], for clustering accuracy assessment. Also, the constraint sets are produced based on the methodology presented by González-Almagro et al. (2020) [19]. The constraints represent must-link pairs of data points that have the same ground truth class and cannot-link pairs of data points that have different classes. Table 1 summarizes the characteristics of the datasets, i.e., the number of instances, attributes, and classes, as did in the experiments.

3.3 Experiments and Results

Experiments with real and synthetic datasets that evaluate the performance of the proposed algorithm by means of the Adjusted Rand Index (ARI) and by comparison with the MLlib K-Means|| algorithm are performed.

Results from the Table 2 show that, at most ratio of constraints to unlabelled instances, Constrained K-Means offers better clustering accuracy than MLlib KMeas||. The results improve substantially on the ARI across breast cancer, new thyroid, and moons datasets,

Table 1. Structure of the datasets

Dataset	Instances	Attributes	Classes
Breast Cancer	569	30	2
Ecoli	336	7	8
Ionosphere	351	33	2
Monk-2	432	6	2
Moons	300	2	2
Newthyroid	215	5	3
Sonar	208	60	2
Spectfheart	267	44	2
Spiral	300	2	2
Vehicle	846	18	4

especially where the constraints are set at approximately 15% or 20% (perfect match with the ground truth labels). This again confirms that constraints effectively guide the clustering by encoding meaningful relations among the data points. Additionally, in datasets like Monk-2 and Moons, accuracy is increased and computation time is reduced by a well formed constraint set. For instance, in Monk-2, ARI increases from 0.12 (5%) to 1.0 (20%) while execution time decreases from 46s to 33.79s, which indicates that an effective must-link formation eliminates redundancies and speeds up convergence.

However, the execution time of Constrained K-Means increases with the increase of constraints and the cost of the computation of Constrained K-Means is higher than the one of MLlib K-Means||. This is expected because verifying must-link and cannot-link constraints involves forming superpoints, validating constraints, and reassigning when violations are found. For example, in Ecoli, execution time grows from 46.33s (5%) to 76s (15%) and the same pattern is observed in Newthyroid, and Monk-2. It was also run in a pseudo distributed environment which could have added to longer runtimes because of local communication overhead. However, execution time is not guaranteed to scale linearly with constraint levels and benefits are achieved when must link constraints are well formed so that they can allow less active data points and partially offset constraint enforcement cost.

This confirms that further constraint levels increase the accuracy of clustering, but at the cost of additional execution time as more constraint processing is involved. In addition, consideration of constraints, most especially must links, can contribute to accuracy and computational efficiency, thus demonstrating how constrained clustering over well structured sets of constraints may achieve high quality results.

Table 2. Performance evaluation of the proposed constrained K-Means algorithm and the K-Means|| algorithm from MLlib with different constraint ratios

Dataset	Constraint Set (%)	Constrained Kmeans		MLlib Kmeans||	
		ARI	Time (s)	ARI	Time (s)
Breast Cancer	5	0.77	54.2	0.65	1.73
Breast Cancer	10	0.06	67.6	0.65	1.73
Breast Cancer	15	1	37.7	0.65	1.73
Breast Cancer	20	1	37	0.65	1.73
Ecoli	5	0.55	46.33	0.39	1.98
Ecoli	10	0.65	54.22	0.39	1.98
Ecoli	15	0.69	76	0.39	1.98
Ecoli	20	0.86	56.5	0.39	1.98
Ionosphere	5	0.2	50.68	0.17	1.89
Ionosphere	10	0.57	58.7	0.17	1.89
Ionosphere	15	0.044	58	0.17	1.89
Ionosphere	20	1	31.79	0.17	1.89
Monk-2	5	0.12	46	−0.0023	1.78
Monk-2	10	0.004	56.75	−0.0023	1.78
Monk-2	15	1	30.28	−0.0023	1.78
Monk-2	20	1	33.79	−0.0023	1.78
Moons	5	0.56	46.69	0.47	1.37
Moons	10	0.76	48.43	0.47	1.37
Moons	15	0.92	45.25	0.47	1.37
Moons	20	1	28.52	0.47	1.37
Newthyroid	5	0.73	33.75	0.58	1.47
Newthyroid	10	0.89	44.99	0.58	1.47
Newthyroid	15	0.85	47.29	0.58	1.47
Newthyroid	20	1	38.73	0.58	1.47
Sonar	5	0.02	48.14	0.01	1.82
Sonar	10	0.07	50.23	0.01	1.82
Sonar	15	0.65	54.95	0.01	1.82
Sonar	20	0.98	41.77	0.01	1.82
Spectfheart	5	0.0036	52.98	−0.1	1.92
Spectfheart	10	0.053	56.9	−0.1	1.92
Spectfheart	15	0.25	57.88	−0.1	1.92

(*continued*)

Table 2. (*continued*)

Dataset	Constraint Set (%)	Constrained Kmeans		MLlib Kmeans‖	
		ARI	Time (s)	ARI	Time (s)
Spectfheart	20	1	44.65	−0.1	1.92
Spiral	5	0.058	50.52	0.008	1.68
Spiral	10	0.42	52.36	0.008	1.68
Spiral	15	0.95	45.9	0.008	1.68
Spiral	20	0.99	48.69	0.008	1.68
Vehicle	5	0.058	51.15	0.09	1.93
Vehicle	10	0.063	57.49	0.09	1.93
Vehicle	15	0.15	62.92	0.09	1.93
Vehicle	20	1	28.87	0.09	1.93

4 Conclusions and Perspectives

This paper shows the way in which Constrained K-Means algorithm can be used to deal with the larger problem of must link and cannot link constraints when working on large scale clustering problems on Apache Spark. We propose a method that makes use of the Spark distributed architecture to generate valid solutions while the process of constraints can be preserved and they retain scalability and efficiency. On standard K-Means, advantages in terms of constraint handling to produce tighter clusters and possible higher interpretability are shown experimentally. Moreover, clustering accuracy also enhances with the enhancement of constraint level which indicates that the clustering process is predicated on the well structured constraint. These features can be offered, but with the cost of additional execution time which implies that there should be no naive solution to handing constraints.

Consequently, the current approach is proven vindicated with positive results which create more room for more work on this approach. The key priority of improving the constraint validation phase aims to minimize computational overhead especially in the setting where there are many constraints. Improving the formation of the superpoint mitigates redundancy and is seen to reduce computation and as well as enhance performance and scalability. However, we can also parallelize the constraint verification process in such a way that it does not impart an extra load on runtime from Constrained K-Means to deal with large data. We also extend the algorithm with soft constraints that eventually gave rise to the confidence values that we later apply to become more agile to various clustering scenarious. On the massive data's, the efficiency and scalability of it is tested finally on a multi node Spark cluster environment. Finally, the approach will consume the real world data to find out further deep insights of how does the approach works for the real big data clustering. If all these problems are solved, constrained clustering algorithms will be proven effective for use in big data environments. The work

can be applied when dealing with time-series data such as biomedical signals [23, 24], insurance expenditures [25, 26], etc.

References

1. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering. ACM Comput. Surv. **31**(3), 264–323 (1999). https://doi.org/10.1145/331499.331504
2. Madhulatha, T.S.: An overview on clustering methods. arXiv (2012). https://doi.org/10.48550/arxiv.1205.1117
3. Basu, S., Davidson, I., Wagstaff, K.: Constrained clustering: Advances in Algorithms, Theory, and Applications. CRC Press (2008)
4. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S.: Constrained K-means Clustering with Background Knowledge. In: International Conference on Machine Learning, pp. 577–584 (2001). https://web.cse.msu.edu/~cse802/notes/ConstrainedKmeans.pdf
5. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In: Proceedings of the Twenty-First International Conference on Machine Learning (ICML '04), p. 11. Association for Computing Machinery (2004). https://doi.org/10.1145/1015330.1015360
6. Baumann, P., Hochbaum, D.S.: PCCC: the pairwise-confidence-constraints clustering algorithm. arXiv (2022). https://doi.org/10.48550/arxiv.2212.14437
7. Zaharia, M., et al.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: Networked Systems Design and Implementation, 2 (2012). http://cobweb.cs.uga.edu/~squinn/mmd_s15/papers/nsdi12-final138.pdf
8. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
9. Arthur, D., Vassilvitskii, S.: K-means++: the advantages of careful seeding. Symp. Discrete Algorithms **1027–1035**,(2007). https://doi.org/10.5555/1283383.1283494
10. Kozyriev, A., Norkin, V.: robust clustering on high-dimensional data with stochastic quantization. arXiv (2024). https://doi.org/10.48550/arxiv.2409.02066
11. Meng, X., et al.: MLLIB: Machine learning in apache spark. arXiv (2015). https://doi.org/10.48550/arxiv.1505.06807
12. Bahmani, B., Moseley, B., Vattani, A., Kumar, R., Vassilvitskii, S.: Scalable K-Means++. arXiv (2012). https://doi.org/10.48550/arxiv.1203.6402
13. Blömer, J., Lammersen, C., Schmidt, M., Sohler, C.: Theoretical analysis of the k-means algorithm a survey. arXiv (2016). https://doi.org/10.48550/arxiv.1602.08254
14. Jędrzejowicz, J., Jędrzejowicz, P., Wierzbowska, I.: Apache spark implementation of the distance-based kernel-based fuzzy c-means clustering classifier. In: Smart Innovation, Systems and Technologies, pp. 317–324 (2016). https://doi.org/10.1007/978-3-319-39630-9_26
15. Han, D., Agrawal, A., Liao, W., Choudhary, A.: Parallel DBSCAN algorithm using a data partitioning strategy with spark implementation. In: Proceedings of the IEEE International Conference on Big Data (Big Data), pp. 305–312 (2018). https://doi.org/10.1109/bigdata.2018.8622258
16. Lavanya, K., Banu, J.S., Jain, P.: Clustering of Zika virus epidemic using gaussian mixture model in spark environment. Biomed. Res. **30**(1) (2019). https://doi.org/10.35841/biomedicalresearch.30-18-1132
17. Apache Software Foundation. Hadoop (2010). https://hadoop.apache.org
18. Dean, J., Ghemawat, S.: MapReduce. Commun. ACM **51**(1), 107–113 (2008). https://doi.org/10.1145/1327452.1327492

19. González-Almagro, G., Luengo, J., Cano, J., García, S.: DILS: constrained clustering through dual iterative local search. Comput. Oper. Res. **121**, 104979 (2020). https://doi.org/10.1016/j.cor.2020.104979
20. Hubert, L., Arabie, P.: Comparing partitions. J. Classif. **2**, 193–218 (1985)
21. Dave, A., Jindal, A., Li, L.E., Xin, R., Gonzalez, J., Zaharia, M.: Graphframes: an integrated api for mixing graph and relational queries. In: Proceedings of the fourth international workshop on graph data management experiences and systems, pp. 1–8 (2016)
22. The Constrained K-Means algorithm code is available on GitHub. https://github.com/huynqcharles/spark-constrained-kmeans
23. Hung, P.D.: Detection of central sleep apnea based on a single-lead ECG. In: Proceedings of the 5th International Conference on Bioinformatics Research and Applications (ICBRA '18), pp. 78–83. Association for Computing Machinery, New York, NY, USA (2018)
24. Hung, P.D.: Central sleep apnea detection using an accelerometer. In: Proceedings of the 1st International Conference on Control and Computer Vision (ICCCV '18), pp. 106–111. Association for Computing Machinery, New York, NY, USA (2018)
25. Hai, P.N., Hieu, H.T., Hung, P.D.: An empirical examination on forecasting VN30 short-term uptrend stocks using LSTM along with the ichimoku cloud trading strategy. In: Sharma, H., Shrivastava, V., Kumari Bharti, K., Wang, L. (eds.) Communication and Intelligent Systems . Lecture Notes in Networks and Systems, vol. 461. Springer, Singapore (2022)
26. Dat, D.Q., Hung, P.D.: Improvement for time series clustering with the deep learning approach. In: Luo, Y. (eds.) Cooperative Design, Visualization, and Engineering. CDVE 2021. Lecture Notes in Computer Science(), vol. 12983. Springer, Cham (2021)