Scalable Distributed K-Means Clustering Using the Firefly Algorithm with Tree- and Hash-Based Optimization for Big Data

Shivalingappa Battur
KLE Technological University, Hubli, Karnataka, India
shivalingappa@kletech.ac.in (corresponding author)

Shashikumar Totad
KLE Technological University, Hubli, Karnataka, India
shashikumar.totad@kletech.ac.in

Received: 4 February 2026 | Revised: 5 April 2026 | Accepted: 18 April 2026

 DOI: https://doi.org/10.48084/etasr.17969

ABSTRACT

The rapid growth of digital data has exposed significant limitations in traditional clustering methods, particularly with respect to scalability, computational overhead, and clustering quality. To address these challenges, this paper proposes Firefly–K-Means with Tree- and Hash-based optimization (FKTH), a scalable distributed clustering framework that integrates an adaptive Firefly Algorithm (FA) with K-Means, enhanced through KD-Tree–based distance computation, hash map–based constant-time centroid updates, and Hadoop MapReduce–based parallel processing. The adaptive Firefly component dynamically adjusts attraction, absorption, and randomness parameters during optimization to balance exploration and exploitation and avoid premature convergence. The proposed framework is evaluated on large-scale real-world datasets ranging from 100K to over 1M records across varying cluster node configurations. Experimental results demonstrate that FKTH achieves superior scalability and consistently outperforms existing metaheuristic-based clustering methods in terms of execution time, Silhouette Score, Davies–Bouldin Index (DBI), and F1-score, making it well suited for large-scale distributed data analytics.

Keywords-distributed clustering; adaptive Firefly Algorithm; K-Means; Hadoop MapReduce; KD-Tree; large-scale data analytics; metaheuristic optimization

I. INTRODUCTION

The exponential growth of digital data generated from heterogeneous sources such as social networks, sensor systems, healthcare platforms, and cloud services has resulted in data volumes that exceed the processing capabilities of conventional analytical tools. This rapid expansion has driven the adoption of scalable distributed computing frameworks, including Hadoop and Apache Spark, to enable efficient storage, processing, and large-scale analytics. Authors in [1] analyze the evolving landscape of big data applications and identify scalability, computational efficiency, and fault tolerance as key challenges motivating ongoing research in distributed data analytics.

Unsupervised machine learning techniques, particularly clustering, play a crucial role in extracting meaningful patterns from large volumes of unlabeled data. Among classical clustering algorithms, K-Means remains one of the most widely used due to its simplicity and low computational cost. However, K-Means suffers from several inherent limitations, including sensitivity to initial centroid selection, convergence to local optima, and reduced effectiveness when applied to high-dimensional or large-scale datasets. These challenges are further amplified in distributed environments, where iterative processing and repeated synchronization significantly impact performance.

To address these limitations, metaheuristic optimization techniques have been increasingly integrated with clustering algorithms. In particular, hybrid metaheuristic–K-Means frameworks leverage global search capabilities to improve centroid initialization and clustering quality. Authors in [2] proposed a scalable Firefly Algorithm (FA)–based clustering framework for distributed environments, demonstrating improved clustering performance through adaptive centroid optimization. Despite these advantages, the population-based and iterative nature of such hybrid approaches introduces substantial computational overhead, which poses scalability challenges for very large datasets.

Reducing the computational cost of distance calculations, the most expensive operation in K-Means clustering, has therefore become a critical research focus. Tree-based optimization techniques have been shown to significantly improve efficiency by structuring data and reducing redundant

computations. Authors in [3] demonstrated the effectiveness of tree-based clustering mechanisms in distributed settings, achieving improved performance through structured data organization. In this context, KD-Tree–based centroid search methods further reduce computational complexity by pruning unnecessary distance evaluations, making them particularly suitable for large-scale clustering tasks.

Metaheuristic clustering methods have been extensively studied in the literature. Authors in [4] provided a comprehensive review of K-Means–based nature-inspired metaheuristic algorithms, highlighting their ability to improve clustering robustness and quality. Alternative clustering paradigms have also been explored. Authors in [5] proposed an accelerated spectral clustering approach using graph filtering techniques, achieving improved accuracy at the cost of increased computational complexity. Authors in [6] reviewed ensemble clustering methods that combine multiple clustering solutions to enhance robustness, although maintaining efficiency in distributed environments remains challenging.

Nature-inspired metaheuristic algorithms have demonstrated strong potential for solving complex optimization problems in high-dimensional spaces. Authors in [7] highlighted the effectiveness of swarm-based and evolutionary algorithms for clustering optimization. Authors in [8] introduced the Sparrow Search Algorithm, which exhibits strong exploration capabilities and competitive performance compared to traditional swarm intelligence techniques. Authors in [9] proposed an enhanced Particle Swarm Optimization (PSO) framework integrated with autoencoders to address high-dimensional clustering, achieving improved convergence speed and clustering accuracy. Nevertheless, the computational cost of these population-based methods continues to limit their scalability for massive datasets.

Hybrid metaheuristic–K-Means models aim to balance global optimization and local refinement. Authors in [10] introduced an FA-based hybrid K-Means approach that improves centroid initialization and convergence behavior. However, such hybrid methods still incur additional computational overhead, necessitating further optimization for large-scale deployment.

Distributed computing platforms have been widely adopted to enhance clustering scalability. Authors in [11] proposed an efficient MapReduce-based K-Means algorithm that parallelizes distance computation and centroid updates across distributed nodes. While effective for handling large datasets, MapReduce-based implementations suffer from frequent disk I/O and synchronization overhead, making them less suitable for iterative algorithms. Authors in [12] reviewed distributed clustering techniques on Apache Spark, highlighting the advantages of in-memory computation while identifying communication overhead as a major limitation. Authors in [13] demonstrated the feasibility of PSO-based clustering on Apache Spark but reported scalability constraints due to inter-node communication costs.

Motivated by the above limitations, there remains a need for scalable clustering frameworks that simultaneously improve clustering quality and computational efficiency in distributed environments. This work addresses this gap by integrating metaheuristic optimization with efficient data structures and distributed processing.

The main contributions of this paper are summarized as follows:

- Design a scalable distributed clustering framework that integrates the FA with K-Means, incorporating KD-Tree–based centroid search and hash map–based indexing to improve convergence behavior and reduce computational overhead.
- Implement the proposed framework on a Hadoop MapReduce platform, enabling efficient large-scale parallel processing for big data clustering tasks.
- Evaluate the effectiveness of the proposed approach on large datasets, demonstrating improvements in execution time and clustering quality compared to existing metaheuristic-based clustering methods.

II. PROPOSED FIREFLY ALGORITHM–BASED DISTRIBUTED K-MEANS FRAMEWORK

This section presents the proposed scalable distributed clustering framework that integrates the FA with K-Means, enhanced through KD-Tree–based centroid search and hash map–based indexing for computational efficiency. The framework is designed for large-scale data processing and is implemented on a Hadoop MapReduce platform. The proposed approach follows a hybrid optimization strategy in which the FA is employed to perform global optimization of cluster centroids, whereas K-Means is used for local refinement. To address the high computational cost associated with distance calculations in large datasets, the framework incorporates KD-Tree–based centroid search and hash-based indexing. The overall workflow consists of data partitioning, centroid optimization using FA, accelerated distance computation, and distributed centroid updates.

A. Design of the Firefly Algorithm–Based Distributed K-Means Model

The proposed framework integrates the FA with K-Means to achieve scalable and efficient clustering in distributed environments. The FA performs global optimization of cluster centroids, mitigating sensitivity to initialization, whereas K-Means provides local refinement for faster convergence. To reduce the computational cost of distance calculations, a KD-Tree–based centroid search mechanism is employed, significantly pruning unnecessary comparisons. Additionally, a hash map–based indexing strategy is used to enable fast centroid access and improve data locality during distributed execution. The entire framework is implemented on a Hadoop MapReduce platform, allowing parallel processing of large-scale datasets with improved clustering quality and execution efficiency.

The FA–K-Means hybrid framework integrates an adaptive FA with efficient data structures on Hadoop MapReduce to achieve scalable, fast, and accurate clustering for large-scale datasets. The proposed framework follows a modular design that combines distributed processing, adaptive optimization,

and efficient data structures. Data are partitioned using the Hadoop Distributed File System (HDFS) to ensure scalability and fault tolerance, followed by preprocessing through normalization to enhance clustering stability. An adaptive FA is employed for centroid initialization, dynamically tuning its control parameters to improve global exploration and avoid local optima. KD-Trees are used to accelerate nearest-centroid searches, whereas hash maps enable constant-time cluster labeling at mapper nodes. Finally, K-Means clustering is executed in a MapReduce manner, where mappers assign data points to centroids, reducers update centroids, and a global FA-based refinement step improves convergence. Let $D = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$ be a dataset of n data points, where each x_i is d-dimensional. The objective is to partition D into k clusters $C = \{C_1, C_2, \ldots, C_k\}$ by minimizing intra-cluster distance.

$$J = \sum_{j=1}^{k} \sum_{x_i \in C_j} \left\| x_i - \mu_j \right\|^2 \quad (1)$$

Firefly Movement: Each firefly f_i represents a candidate solution corresponding to a set of cluster centroids. The movement of firefly f_i toward a brighter firefly f_j is defined as follows:

$$f_i^{t+1} = f_i^{(t)} + \beta e^{-\gamma r_{ij}^2} \left(f_j^{(t)} - f_i^{(t)} \right) + \alpha(\text{rand} - 0.5) \quad (2)$$

Algorithm 1: FA-K-Means Hybrid Clustering with Adaptive Parameters
Input: Dataset $D = \{x_1, x_2, \ldots, x_n\}$, number of clusters k, number of fireflies m, maximum iterations T
Output: Final cluster centroids and cluster assignments

1. Partition and distribute dataset D across nodes using HDFS.
2. Preprocess data chunks (normalization and noise removal).
3. Initialize firefly population $F = \{f_1, f_2, \ldots, f_m\}$.
4. Set initial FA parameters: α_0, β_0, γ_0, and decay factor δ.
5. For t = 1 to T do:
6. Update adaptive parameters:
   $\alpha_t = \alpha_0 \cdot \delta^t$
   $\beta_t = \beta_0 \cdot (1 + t / T)$
   $\gamma_t = \gamma_0 \cdot (1 + D_t / D_{max})$, where D_t denotes population diversity at iteration t.
7. Construct KD-Tree T_{KD} using the centroids of all fireflies F.
8. Broadcast T_{KD}, α_t, β_t, and γ_t to mapper nodes.
9. Execute Map tasks using Algorithm 2.
10. Execute Reduce task using Algorithm 3.
11. Update firefly population F with newly computed centroids.
12. End for.
13. Compute clustering validation metrics (Davies-Bouldin Index (DBI) and Silhouette Score).
14. Return final cluster centroids and assignments.

As discussed in Algorithm 1, the Master node serves as the controller of the distributed clustering process. It initializes the firefly population, where each firefly represents a candidate solution. To improve convergence behavior and avoid local optima, the FA's core parameters—randomness (α), attractiveness (β), and light absorption coefficient (γ)—are dynamically updated at each iteration t as follows:

$$\alpha_t = \alpha_0 \cdot \delta^t \quad (3)$$

$$\beta_t = \beta_0 \cdot \left(1 + \frac{t}{T}\right) \quad (4)$$

$$\gamma_t = \gamma_0 \cdot \left(1 + \frac{D_t}{D_{max}}\right) \quad (5)$$

Algorithm 2: Mapper Algorithm – Firefly Evaluation Using KD-Tree of Centroids
Input: Local data chunk Di, KD-Tree T_{KD} of firefly centroids, firefly IDs, α_t, β_t, γ_t
Output: Local centroid sums and counts

1. Receive KD-Tree T_{KD} constructed by the Master node.
2. For each firefly f_i (identified by its ID) do:
3. Evaluate f_i using local data and the clustering objective.
4. For each brighter firefly f_j do:
5. Update f_i using the Firefly movement (3).
6. End for.
7. Use T_{KD} for efficient nearest-centroid search.
8. End for.
9. Select the best firefly f* with the minimum objective D_{vi}.
10. For each data point x in D_i do:
11. Assign x to the nearest centroid of f* using T_{KD}.
12. Update local centroid sum and count using a hash map.
13. End for.
14. Emit <centroid_id, (sum, count)> to the reducer.

In Algorithm 2, each Mapper receives a KD-Tree constructed from the current firefly centroids. The Mapper uses this structure to efficiently evaluate the clustering performance of each firefly on its local data using the objective function. Firefly positions are then updated using the FA's movement logic based on relative brightness (fitness).

Once the best firefly is selected, it is used to assign data points to the nearest cluster centroid. Local centroid sums and counts are updated using a hash map and emitted as intermediate key-value pairs to the Reducer.

Algorithm 3: Reducer Algorithm – Global Centroid Aggregation
Input: Key-value pairs <centroid_id, (sum, count)> received from all Mapper tasks

```
Output: Updated global cluster centroids
1. For each centroid ID do:
2. Initialize total_sum = 0,
   total_count=0.
3. For each received value (sum, count)
   do:
4.    total_sum+=sum
5.    total_count+=count
6. End for.
7. Compute new centroid: c =
   total_sum/total_count.
8. End for.
9. Return updated centroids to the Master
   node for the next iteration
```

Algorithm 3 defines the Reducer's role in aggregating the results from all Mappers. For each cluster ID, it sums the local centroid vectors and corresponding counts received from Mappers. These are used to compute updated global centroids. The new centroids are then used by the Master to refresh the firefly population for the next iteration, thereby ensuring coordinated global updates.

B. Dataset Description

To evaluate the scalability and effectiveness of the proposed Firefly–K-Means with Tree- and Hash-based optimization *(*FKTH) clustering model, experiments were conducted using four real-world datasets, including Electricity [14], Cover Type [15], Online Shoppers Purchasing Intention Dataset [16], and HHAR [17]. Table I summarizes the characteristics of the datasets used for experimental evaluation. To assess scalability, the datasets were expanded to 100K, 500K, and 1M records using synthetic data generation techniques. Gaussian noise was added to preserve the statistical distribution of the original datasets.

TABLE I. DESCRIPTION OF DATASETS USED FOR EXPERIMENTAL EVALUATION

Dataset	Description
Electricity [14]	A large time-series dataset used for predicting electricity price changes based on demand and supply conditions. It contains >2M records with temporal features, making it ideal for streaming and big data analysis.
Cover Type [15]	A classification dataset used to predict forest cover types based on cartographic and environmental features. It has >581K instances and 54 attributes, suitable for large-scale clustering and classification tasks.
Online Shoppers Purchasing Intention Dataset [16]	A real-world e-commerce dataset used to predict whether a visitor will make a purchase based on browsing behavior and session features. It contains 12,330 instances with 18 features, making it suitable for classification, clustering, and customer behavior analysis.
HHAR [17]	A real-world dataset capturing human activities using smartphone and smartwatch sensor data. It includes millions of time-series records, making it suitable for big data, streaming, and activity recognition tasks.

III. RESULTS AND DISCUSSION

This section compares the proposed FKTH algorithm with standard K-Means, Firefly-initialized K-Means, FA-KMeans with KD-Tree optimization, and other hybrid methods, including PSO-KMeans, BFOA-KMeans, and SOS-KMeans, to evaluate clustering quality, scalability, and computational efficiency. All experiments were conducted on a Hadoop cluster consisting of up to 11 nodes. Each node was equipped with a 12th Generation Intel Core™ i5-12400 processor (2.5 GHz) and 16 GB of RAM. The Hadoop version used was 3.3.6. The parameters for all algorithms were selected based on standard practices and empirical tuning. K-Means uses dataset-dependent cluster size, fixed iterations, and Euclidean distance.

The FA-based variants employ a moderate population size with tuned α, β, and γ parameters, whereas KD-Tree is used for faster distance computation. The PSO, Bacterial Foraging Optimization Algorithm (BFOA), and Symbiotic Organisms Search (SOS) algorithm methods use standard parameter ranges to ensure convergence and diversity. The proposed FKTH integrates Firefly optimization with KD-Tree and hashing, maintaining consistent settings across datasets.

A. Performance Analysis Considering Execution Time

Table II summarizes the execution times across all dataset sizes and cluster configurations. The proposed FKTH algorithm consistently demonstrated superior performance, achieving the lowest runtime in every scenario. For example, when clustering 1M records on an 11-node Hadoop cluster, FKTH completed the task in just 61.3 s, whereas standard K-Means required 121.0 s, indicating nearly a 50% reduction in computation time. This consistent efficiency highlights the benefits of combining Firefly-based optimization, KD-Tree acceleration, and hash map–based labeling within a distributed MapReduce framework.

Figure 1 shows that for 100K samples, execution time decreases for all algorithms as the number of nodes increases, indicating effective parallelization. The proposed FKTH consistently achieves the lowest execution time across all node configurations, with reductions of approximately 20–60% compared to K-Means and other metaheuristic-based methods, demonstrating superior efficiency and scalability even for smaller datasets.

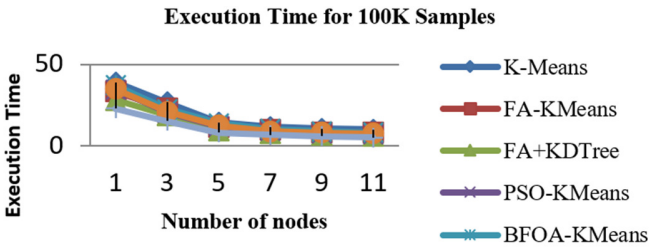

Fig. 1. Performance evaluation: execution time for the 100K dataset.

Figure 2 shows that FKTH consistently achieves the lowest execution time across all node configurations, reducing runtime by approximately 20–44% compared to standard K-Means and other metaheuristic variants. The improvement becomes more evident as node parallelism increases, confirming faster convergence and better scalability for medium-scale datasets.

TABLE II. EXECUTION TIME (IN SECONDS) OF CLUSTERING ALGORITHMS FOR VARYING DATASET SIZES AND NODE COUNTS

Dataset size	Nodes	K-Means	FA-KMeans	FA+KDTree	PSO-KMeans	BFOA-KMeans	SOS-KMeans	FKTH (proposed)
100K	1	39.1	33.4	28.2	34.7	37.1	35.0	22.8
	3	26.7	22.9	18.7	20.4	22.6	21.0	15.3
	5	14.3	11.6	9.4	12.1	13.9	12.6	8.1
	7	11.8	9.8	7.8	9.2	10.3	9.1	6.7
	9	10.5	8.5	6.9	8.0	9.0	8.1	5.9
	11	10.0	8.1	6.3	7.5	8.3	7.6	5.2
500K	1	215.2	190.1	151.3	175.4	196.0	182.2	121.6
	3	113.1	96.6	78.5	88.2	99.0	91.5	61.4
	5	81.3	69.2	55.1	61.4	70.2	63.8	42.2
	7	66.9	55.0	43.8	48.9	56.5	50.8	33.6
	9	60.3	50.1	39.2	44.1	50.7	45.3	30.0
	11	58.2	48.9	37.8	42.6	48.2	43.7	28.5
1M	1	468.7	409.6	338.4	372.8	410.5	390.3	278.2
	3	238.3	206.7	165.9	178.2	201.4	187.6	138.3
	5	168.5	141.0	109.6	122.7	139.4	126.1	95.5
	7	139.1	115.4	87.6	98.2	113.7	102.5	74.1
	9	125.7	105.8	77.2	88.9	103.5	91.0	66.2
	11	121.0	100.3	72.0	83.1	96.2	85.7	61.3

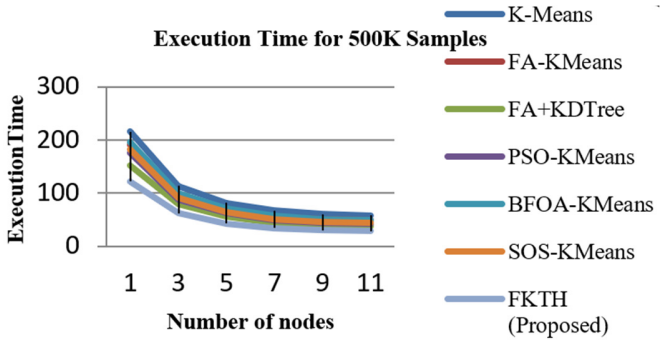

Fig. 2. Performance evaluation: execution time for the 500K dataset.

Figure 3 demonstrates that for 1M samples, FKTH outperforms all competing methods with execution time reductions of approximately 18–41%, particularly at lower node counts. These results highlight the framework's ability to efficiently handle large-scale data while maintaining strong scalability with increasing parallelism.

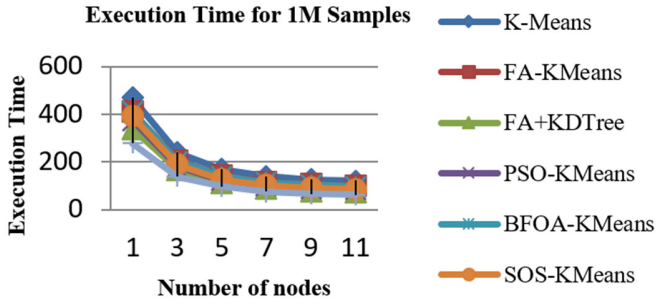

Fig. 3. Performance evaluation: execution time for the 1M dataset.

B. Silhouette Scores and Davies–Bouldin Index

Table III shows that the proposed FKTH method achieves the highest average Silhouette Score (0.672) and the lowest DBI (0.783) among all compared algorithms, indicating superior cluster cohesion and separation. Compared to traditional K-Means (Silhouette = 0.609, DBI = 0.891) and other hybrid methods, FKTH consistently delivers improved clustering quality while maintaining scalability.

TABLE III. COMPARISON OF AVERAGE SILHOUETTE SCORES AND DBI VALUES ACROSS ALGORITHMS

Algorithm	Avg. Silhouette Score (↑)	Avg. DBI (↓)
K-Means	0.609	0.891
FA-KMeans	0.632	0.854
FA + KD-Tree	0.657	0.826
PSO-KMeans	0.643	0.838
BFOA-KMeans	0.636	0.843
SOS-KMeans	0.64	0.84
FKTH (proposed)	0.672	0.783

Figure 4 shows that FKTH achieves the highest average Silhouette Score (≈0.672) and the lowest DBI (≈0.783) among all compared algorithms, indicating superior cluster cohesion and separation. In contrast, traditional K-Means records a lower Silhouette Score (≈0.609) and higher DBI (≈0.891), confirming the clear improvement in clustering quality achieved by the proposed method.

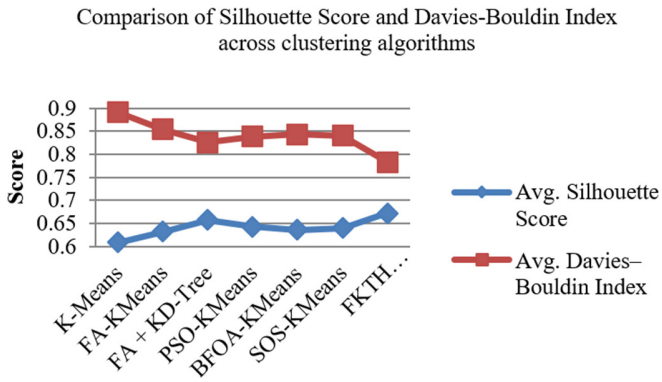

Fig. 4. Comparison of Silhouette Score and DBI across clustering algorithms.

C. Precision, Recall, and F1-Score Analysis

Table IV shows that the proposed FKTH algorithm achieves the highest average precision and F1-score among all compared methods while maintaining competitive recall performance. This statistically confirms its superior accuracy and robustness in correctly grouping data points, further validating its effectiveness for real-world clustering tasks.

TABLE IV. COMPARISON OF AVERAGE PRECISION, RECALL, AND F1-SCORE ACROSS ALGORITHMS

Algorithm	Avg. Precision (%)	Avg. Recall (%)	Avg. F1-score (%)
K-Means	86.5	82.7	86.1
FA-KMeans	88.2	84.2	87.1
FA + KD-Tree	88.9	85.3	86.9
PSO-KMeans	88.1	85.8	86.3
BFOA-KMeans	87.9	84.8	86.4
SOS-KMeans	88	85	88.4
FKTH (proposed)	91.2	84.6	89.8

IV. CONCLUSION

This paper presented Firefly–K-Means with Tree- and Hash-based optimization (FKTH), a scalable Firefly Algorithm (FA)–based optimization framework for K-Means clustering, enhanced with KD-Tree and hash map–based acceleration techniques and implemented in a distributed Hadoop environment. Extensive experiments conducted on datasets of varying sizes and cluster configurations demonstrate that FKTH consistently outperforms both conventional K-Means and state-of-the-art hybrid metaheuristic clustering methods in terms of execution efficiency and clustering accuracy.

Experimental results show that FKTH achieves runtime reductions of 20–60% on small-scale datasets (100K samples), 20–44% on medium-scale datasets (500K samples), and 18–41% on large-scale datasets (1M samples), highlighting its strong scalability and efficiency across increasing data volumes. In terms of clustering quality, FKTH attains the highest average Silhouette Score (0.672) and the lowest Davies–Bouldin Index (DBI) (0.783) among all evaluated methods, indicating superior cluster cohesion and separation. Compared to traditional K-Means, which achieved a Silhouette Score of 0.609 and a DBI of 0.891, FKTH demonstrates a significant improvement in clustering performance.

Furthermore, the effectiveness of the proposed framework is validated using both internal evaluation measures, including Silhouette Score and DBI, and external metrics such as precision, recall, and F1-score. Scalability is further confirmed through consistent performance gains with increasing dataset sizes and higher degrees of node-level parallelism. These results establish FKTH as an efficient and scalable solution for large-scale clustering in distributed big data environments.

DECLARATION OF COMPETING INTERESTS

The authors declare no conflicts of interest.

ACKNOWLEDGMENT

Not applicable to this work.

DATA AVAILABILITY

The datasets used in this study are publicly available at[14-17].

REFERENCES

[1] A. Badshah, A. Daud, R. Alharbey, A. Banjar, A. Bukhari, and B. Alshemaimri, "Big data applications: overview, challenges and future," *Artificial Intelligence Review*, vol. 57, no. 11, Sept. 2024, Art. no. 290, https://doi.org/10.1007/s10462-024-10938-5.

[2] S. Battur, N. Tejas, B. Naveenkumar, K. Aditi, T. V, and S. G. Totad, "Scalable Data Clustering Using Firefly Algorithm in Distributed Environment," in *6th International Conference on Data Science and Applications*, Jaipur, India, 2025, pp. 348–358, https://doi.org/10.1007/978-3-032-12827-0_27.

[3] N. Sikarwar and R. S. Tomar, "A New Approach for Wireless Sensor Networks based on Tree-based Routing using Hybrid Fuzzy C-Means with Genetic Algorithm," *Engineering, Technology & Applied Science Research*, vol. 14, no. 3, pp. 14141–14147, June 2024, https://doi.org/10.48084/etasr.7078.

[4] A. M. Ikotun, M. S. Almutari, and A. E. Ezugwu, "K-Means-Based Nature-Inspired Metaheuristic Algorithms for Automatic Data Clustering Problems: Recent Advances and Future Directions," *Applied Sciences*, vol. 11, no. 23, Dec. 2021, Art. no. 11246, https://doi.org/10.3390/app112311246.

[5] N. Tremblay, G. Puy, P. Borgnat, R. Gribonval, and P. Vandergheynst, "Accelerated spectral clustering using graph filtering of random signals," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing*, Shanghai, China, 2016, pp. 4094–4098, https://doi.org/10.1109/ICASSP.2016.7472447.

[6] K. Golalipour, E. Akbari, S. S. Hamidi, M. Lee, and R. Enayatifar, "From clustering to clustering ensemble selection: A review," *Engineering Applications of Artificial Intelligence*, vol. 104, Sept. 2021, Art. no. 104388, https://doi.org/10.1016/j.engappai.2021.104388.

[7] X.-S. Yang, *Nature-inspired Metaheuristic Algorithms*, 2nd ed. Beckington, Somerset, UK: Luniver Press, 2010.

[8] J. Xue and B. Shen, "A novel swarm intelligence optimization approach: sparrow search algorithm," *Systems Science & Control Engineering*, vol. 8, no. 1, pp. 22–34, Jan. 2020, https://doi.org/10.1080/21642583.2019.1708830.

[9] S. Battur, R. H. Shrinidhi, A. Kinagi, D. G. Nayana, M. Priya, and S. G. Totad, "Enhancing the Performance of PSO Algorithm for Clustering High-Dimensional Data Using Autoencoders," in *International Conference on Data Science and Applications*, Jaipur, India, 2023, pp. 515–534, https://doi.org/10.1007/978-981-99-7817-5_38.

[10] T. Hassanzadeh and M. R. Meybodi, "A new hybrid approach for data clustering using firefly algorithm and K-means," in *The 16th CSI International Symposium on Artificial Intelligence and Signal Processing*, Shiraz, Iran, 2012, pp. 007–011, https://doi.org/10.1109/AISP.2012.6313708.

[11] Q. Li, P. Wang, W. Wang, H. Hu, Z. Li, and J. Li, "An Efficient K-means Clustering Algorithm on MapReduce," in *19th International Conference on Database Systems for Advanced Applications*, Bali, Indonesia, 2014, pp. 357–371, https://doi.org/10.1007/978-3-319-05810-8_24.

[12] M. M. Saeed, Z. A. Aghbari, and M. Alsharidah, "Big data clustering techniques based on Spark: a literature review," *PeerJ Computer Science*, vol. 6, Nov. 2020, Art. no. e321, https://doi.org/10.7717/peerj-cs.321.

[13] M. Sherar and F. Zulkernine, "Particle swarm optimization for large-scale clustering on apache spark," in *2017 IEEE Symposium Series on Computational Intelligence*, Honolulu, HI, USA, 2017, pp. 1–8, https://doi.org/10.1109/SSCI.2017.8285208.

[14] A. Trindade, "ElectricityLoadDiagrams20112014." UCI Machine Learning Repository, 2015, https://doi.org/10.24432/C58C86.

[15] J. Blackard, "Covertype." UCI Machine Learning Repository, 1998, https://doi.org/10.24432/C50K5N.

[16] Y. K. C. Sakar, "Online Shoppers Purchasing Intention Dataset." UCI Machine Learning Repository, 2018, https://doi.org/10.24432/C5F88Q.

[17] S. B. Henrik Blunck, "Heterogeneity Activity Recognition." UCI Machine Learning Repository, 2015, https://doi.org/10.24432/C5689X.