An Improved K-means Clustering Algorithm for Multi-dimensional Multi-cluster data Using Meta-heuristics

Conference Paper · January 2022

DOI: 10.1109/ICCIT54785.2021.9689836

4 authors, including:

Faisal Bin Ashraf
University of California, Riverside
45 PUBLICATIONS **377** CITATIONS

SEE PROFILE

Abdul Matin
University of Technology Sydney
11 PUBLICATIONS **87** CITATIONS

SEE PROFILE

Muhammad Usama Islam
University of Louisiana at Lafayette
36 PUBLICATIONS **444** CITATIONS

SEE PROFILE

An Improved K-means Clustering Algorithm for Multi-dimensional Multi-cluster data Using Meta-heuristics

Faisal Bin Ashraf* , Abdul Matin†, Md. Shafiur Raihan Shafi‡, and Muhammad Usama Islam§
*Department of Computer Science and Engineering, Brac University, Bangladesh
†Department of Computer Science and Engineering, Uttara University, Bangladesh
‡Department of Computer Science and Engineering, Southeast University, Bangladesh
§Center for Advanced Computer Studies, University of Louisiana at Lafayette, United States

***Abstract*—k-means is the most widely used clustering algorithm which is an unsupervised technique that needs assumptions of centroids to begin the process. Hence, the problem is NP-hard and needs careful consideration and optimization to get a better quality of clusters of data. In this work, a meta-heuristic based genetic algorithm is proposed to optimize the centroid initialization process. The proposed method includes tournament selection, probability-based mutation, and elitism that leads to finding the optimal centroids for the clusters of a given dataset. Nine different and diversified datasets were used to test the performance of the proposed method in terms of the davies-bouldin index and it performed better in all the datasets than the standard k-means and minibatch k-means algorithm.**

***Index Terms*—K-means, Clustering Algorithm, Meta-heuristic, Genetic Algorithm, Improved Clustering Technique**

I. Introduction

Clustering is one of the most ubiquitous data mining techniques to formulate mutually exclusive clusters from unsupervised data [1]. K-means is one of the most widely used partitional clustering techniques to find the appropriate centroids for clustering operation [2]. The centroid represents the mean of each cluster. Although it is the simplest and practical clustering method [3], this algorithm has some drawbacks due to its sensitivity to initial cluster center placement. Additionally, it requires some clusters even before starting partitionalization. These issues of the heuristic algorithm (K-means) can be solved by metaheuristic algorithms. Different variants and techniques have been proposed to address these issues.

Let, X = { x_1, x_2, …, x_n} be a set of n data points in m dimensional space. The task is to divide these n data points into k number of mutually exclusive clusters C = {C_1, C_2, …, C_k } and $c_i = i^{th}$ cluster's centroid. It creates the clusters by optimizing an objective function. The most commonly used objective function is Sum of Squared Error (SSE):

$$SSE = \sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - c_i||_2^2 \quad (1)$$

Issues with this algorithm include identifying the number of clusters that will effectively cluster the data points and also initializing the centroids for each cluster. Besides, using a more effective objective function will help to optimize faster. K-means clustering is an NP-hard problem [4]. Meta-heuristic approaches contribute significantly to solve NP-hard problems [5]–[8]. In this study, we propose a meta-heuristic approach that improves the performance of the K-means clustering algorithm and evaluate this performance improvement of the proposed algorithm on a variety of datasets.

II. Related Works

Different variations for improving the k-means algorithm have been introduced in the literature. As mentioned earlier, Metaheuristic algorithms can be a solution to solve related optimization problems [9], [10] occurred in K-means algorithm. Therefore, researchers are now choosing metaheuristic algorithms for reducing optimization problems and clustering issues.

An unsupervised learning scheme was proposed which removed the initialization without parameter selection [11]. They used the number of data points as the number of clusters at the beginning to avoid the initializing issue. During the continuation, this approach discarded extra clusters and eventually ended up with the optimal number of clusters. They proposed an objective function adding entropy as a penalty term. The performance of this approach was found better than k-means, Xmeans, R-EM, C-FS for some synthetic and real datasets.

Evolutionary metaheuristic algorithms were introduced to seek the optimal cluster number and appropriate clustering structure of scalable datasets in a work [12]. In this paper, k-means clustering was iteratively enhanced by evolutionary operators to manage distributed data. A hybrid clustering method using Harmony search and K-means was presented in [13] with a claim of satisfactory clustering outcomes.

When the k-means algorithm shows sensitivity to select the initial centers, the global optimum may be hampered by the poor choice of centers. [14]. Some Genetic K-means algorithms were introduced in [15]–[18] to minimize the sensitivity for finding the global optima. A crossover operator is employed to exchange neighbors on benchmark and

978-1-6654-9435-9/21/$31.00

simulated datasets to select centers for superior clustering performance [15]. A hybrid GA-based k-means algorithm was introduced in [16] to find a globally optimal partition. Hence, GAK (genetic K-means algorithm) is used as an inexpensive search operator compared to crossover. Moreover, to solve the initialization problem of the k-means algorithm, the authors in [17] proposed two algorithms namely GAIK (GA Initializes to KM) and KIGA (K-means Initializes to GA).

Particle Swarm Optimization (PSO) was applied with K-means by using mutation with particles of PSO when a swarm was trapped in local optima [19]. They used eight real datasets to test the method and found better results than other versions of PSOs like PSO, PSOFKM, and PSOLF-KHM. Artificial Bee colony was also utilized to classify benchmark dataset and was found better than PSO [20], [21]. Each candidate's search direction followed Deb's rule [21]. This algorithm was tested against a version of GA, SA, TS, ACO and found to give better clusters.

Two variants of Firefly algorithms were introduced in another study [22] for reducing the problem of initialization and local optima trap of K-means. ACO–ALO algorithm was proposed to improve the intra-cluster distance of the clusters [23]. They used Ant Lion Optimization with Ant Colony Optimization and incorporated Cauchy's mutation operator.

A grey wolf optimization technique was also introduced to counter this problem in another study [24]. They managed to get a better intra-cluster distance. All researchers claimed in the above literature that metaheuristic algorithms are good enough to perform well in clustering unsupervised data. But it could be difficult to choose one which can provide the best performance to find centroids of clusters in all conditions. In this study, a genetic algorithm-based solution is proposed to find the optimal center points for the clusters of a given data set.

III. Proposed Method

To find the best centroids for clustering data points, a genetic algorithm with elitism is proposed in this section. The algorithm incorporates initializing the first generation of centroids and then, based on the fitness for each way of clustering, next generations are produced with proper selection and mutation steps. The overall idea of the solution process is shown in Figure 1.

Algorithm 1 describes the process with all the necessary parameters. In this method five parameters have been used to tune the optimization process.

- $numOfInd$: number of individual in population
- n: number of elite individual in each population
- P_s: fraction of worst fitted population that can be replaced by good candidate solution so that we can explore the good region
- P_m: probability of a gene to be mutated
- $budget$: number of the times we will run our optimization process

In each run of the optimization process, we assess the fitness of each individual of the population and find the best individual. And then, we select the n best elite individual who will directly go into the next generation as a candidate. However, after the mutation process, we generate more $(numOfInd - n)$ individuals and the next generation of candidate solution is created for optimization.

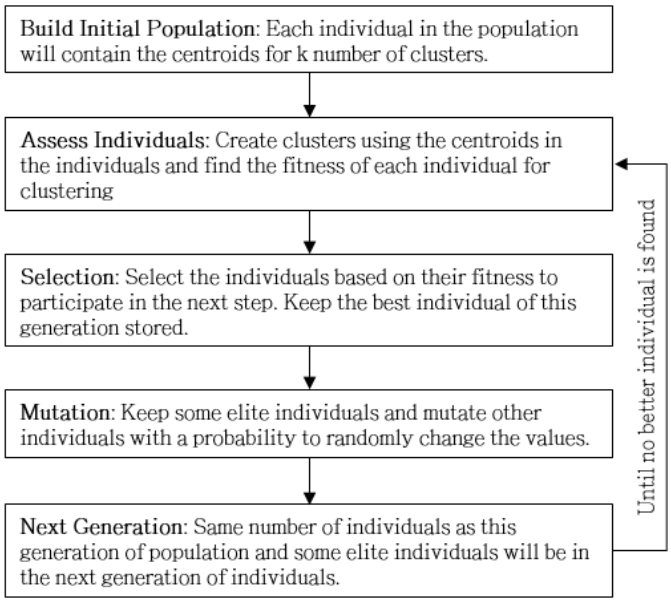

Fig. 1. Proposed Method

Algorithm 1: Proposed Algorithm

```
numOfInd ← Number of Individual in Population
P_s ← worst population to be replaced
P_m ← probability of mutation
n ← number of elite individual
budget ← number of generation
P ← BuildInitialGeneration{}
Best ← {}
while budget ≥ 0 do
    for each individual P_i ∈ P do
        AssessFitness(P_i)
        if Best = {} or Fitness(P_i) > Fitness(Best)
         then
            Best ← P_i
        end
    end
    Q ← n fittest individuals from P
    S ← Select individuals replacing the P_s worst
     individuals
    for P_i ∈ (numOfInd − n) last individuals do
        Q ← Q∪ { Mutate(P_i) }
    end
    P ← Q
    budget ← budget - 1
end
return Best
```

A. Representation and Initial Population

Our goal is to find the optimal centroid points for each cluster from the given data set. We know the number of clusters we want to create and the dimension of each data point. So, the representation of our individuals should contain k number of data points with d dimension. Hence, the length of our individual will be,

$$\text{length of individual} = k \times d$$

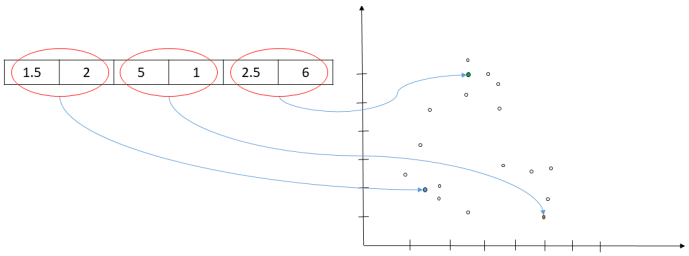

Fig. 2. Representation of Individual

Figure 2 shows, how a solution is represented in our method. Here, the value of $k = 3$ and $d = 2$. So, the length of the individual will be 6 and each d consecutive values represent a data point in the d-dimensional space. Hence, there will be k number of data points representing initial centroids of k number of clusters. For clustering, we need to pre-process data and normalize the dataset. Because in our representation we will assign a uniformly distributed random number from 0.0 to 1.0. Hence, we need to normalize the dataset. The pseudo-code is shown in Algorithm 2.

Algorithm 2: *BuildInitialGeneration*

```
numOfInd ← number of individuals in population
length ← length of each individuals
P ← {}
for numOfInd times do
    ind ← {}
    for length times do
        a ← random number between 0.0 and 1.0
          using uniform distribution
        ind ← ind ∪ a
    end
    P ← P ∪ ind
end
return P
```

B. Fitness Assessment

We need to assess the fitness of our individuals, i.e., assumed center points of each cluster. For this, the davies-bouldin index is used in this method. Davies-bouldin index is used to measure the validity of clusters from a set of given data points. It calculates the inter-cluster distance as well as intra-cluster distance and gives a ratio of average intra-cluster distance with maximum inter-cluster distance (Figure 3). So, using this measurement we can compare how well the data have been clustered. Good clustering will have minimum intra-cluster distance and maximum inter-cluster distance. Hence, a lower value of the davies-bouldin index will represent a better clustering. Algorithm 3 shows the pseudo-code for the fitness assessment.

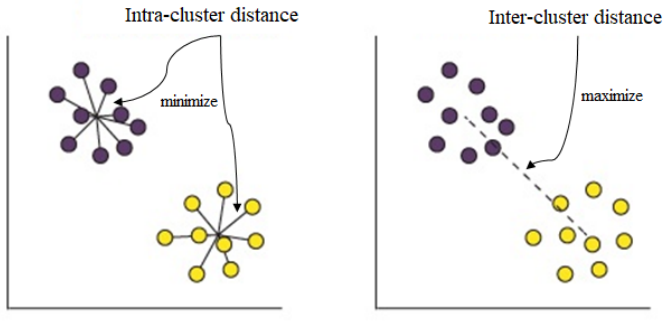

Fig. 3. Intra-cluster and Inter-cluster distance

$R_{i,j}$ determines the goodness of clustring between two cluster i and j. The formula of $R_{i,j}$ is expressed below.

$$R_{i,j} = \frac{S_i + S_j}{M_{i,j}} \tag{2}$$

S_i = average distance of all points in cluster i from its centroid
$M_{i,j}$ = distance between the centroid of cluster i and cluster j

$$D_i = Max(D_{i,j}), \forall j \tag{3}$$

So, if there are k number of clusters, the davis bouldin index will be,

$$DBIndex = \frac{1}{k}\sum_{i=1}^{k} D_i \tag{4}$$

As lower value of DBIndex represents good quality of clusters, we have defined our fitness function as,

$$Fitness = \frac{1}{DBIndex} \tag{5}$$

C. Selection

The selection step defines which individuals will take part in the next step and eventually in the next generation. This step is very crucial. We need to design the selection process meticulously so that we can explore the found best solution and at the same time, we have the scope to exploit the solution space so that we do not get stuck in the local optima. In this method, the tournament selection process has been adopted.

In the tournament selection process, we need to define another new parameter t, which determines the number of the selected individual at random from the generation. We compare those t number of individuals and find the best. This individual is selected for replacing the bad solutions in the generation. Algorithm 4 shows the pseudo-code of this method.

Algorithm 3: $AssessFitness(P_i)$

$C \leftarrow$ clusters created using the centroids P_i
$S_i \leftarrow$ distances of all points in cluster i from the centroid
$nC \leftarrow$ number of clusters
for *i in C* **do**
  $D_i \leftarrow 0.0$
  $maxR_{i,j} \leftarrow 0.0$
  for *j in C* **do**
    if $i != j$ **then**
      $d \leftarrow$ distance between centroid of i and j
      $R_{i,j} \leftarrow \frac{S_i+S_j}{d}$
      if $R_{i,j} > maxR_{i,j}$ **then**
        $maxR_{i,j} \leftarrow R_{i,j}$
      end
    end
  end
  $D_i \leftarrow D_i + maxR_{i,j}$
  $max_{i,j} = 0.0$
end
$DBIndex \leftarrow \frac{D_i}{nC}$
return $\frac{1}{DBIndex}$

Algorithm 4: Selection(P) - Tournament Selection

$P_s \leftarrow$ portion of the population to be replaced
$i \leftarrow numOfInd \times P_s$
$t \leftarrow$ tournament size
$P \leftarrow$ sorted individuals in P based on their fitness
for *i times* **do**
  $P_t \leftarrow$ best individual among random t number of individual from P
  $P[numOfInd - i] \leftarrow P_t$
end
return P

D. Mutation

To achieve more exploitation this method incorporates a probability-based mutation technique. In this step, we mutate the coordinates of center points of the clusters based on a probability, which is a parameter P_m to tune. And, in the mutation, a new value between 0.0 to 1.0 is uniformly assigned. In this way, a good amount of exploitation is included in the optimization process. The pseudo-code is given in Algorithm 5.

E. Code, Environment and Availability

The algorithm was implemented using python language on a machine of AMD Ryzen 5 with 8 GB ram. The codebase was created using the OOP concept. The implementation is available on this GitHub repository - https://github.com/fbabd/updated-kmeans.

IV. Experimentation and Result

The implemented algorithm was tested on nine different real datasets. All of these datasets are collected from the UCI machine learning repository. The number of data points, data dimension, and the number of clusters in each dataset is shown in Table I. The data sets were chosen to ensure the variation.

Algorithm 5: Mutation(P)

$P_m \leftarrow$ probability of getting mutated
$Q \leftarrow$ best n individuals
$P \leftarrow P - Q$
for *each individual* $P_i \in P$ **do**
  for *each position* j *of* P_i **do**
    $d \leftarrow$ random number between 0.0 and 1.0
    if $d <= P_m$ **then**
      $P_i[j] \leftarrow$ random value using uniform distribution
    end
  end
end
$P \leftarrow Q \cup P$
return P

TABLE I
Used Dataset and Number of Clusters

Dataset	Number of Data	Data Dimension	Number of Cluster
Iris	150	4	3
Heart Failure	299	12	2
Seed Data	210	7	2
Thyroid	215	5	2
Wine	178	13	3
Yeast	1484	8	10
Breast	699	9	2
Glass	214	9	7
Wdbc	569	32	2

The proposed genetic algorithm was implemented using the following parameter values : k = depends on the dataset, $budget$ = 100, $numOfInd$ = 20, P_s = 0.2, P_m = 0.25, t = 2, and n = 2. And the davies-bouldin index was recorded for each of the experimentation. Optimizing the davies-bouldin index for some datasets are shown in Figure 4.

If we look closely at Figure 4 we see that in all the datasets, over time it finds a better solution in the population. Sometimes, the optimization process cannot improve but our exploitation technique involving mutation and random value assigning is serving its purpose effectively. And after some generation, we start to get a better solution irrespective of the size, dimension, and the number of clusters.

Figure 5 shows the comparison of the proposed method with well-established k-means and minibatch k-means algorithm. The comparison is shown in terms of the Davies-bouldin Index. A lower value of the index represents the better quality of clusters. From the comparison, we see that the proposed method outperforms both established methods for almost all the datasets.

The proposed algorithm works better than these two because of the meta-heuristics that have been introduced in finding the better centroids for the clusters. In the process, selection and mutation processes with elitism ensure to get the better

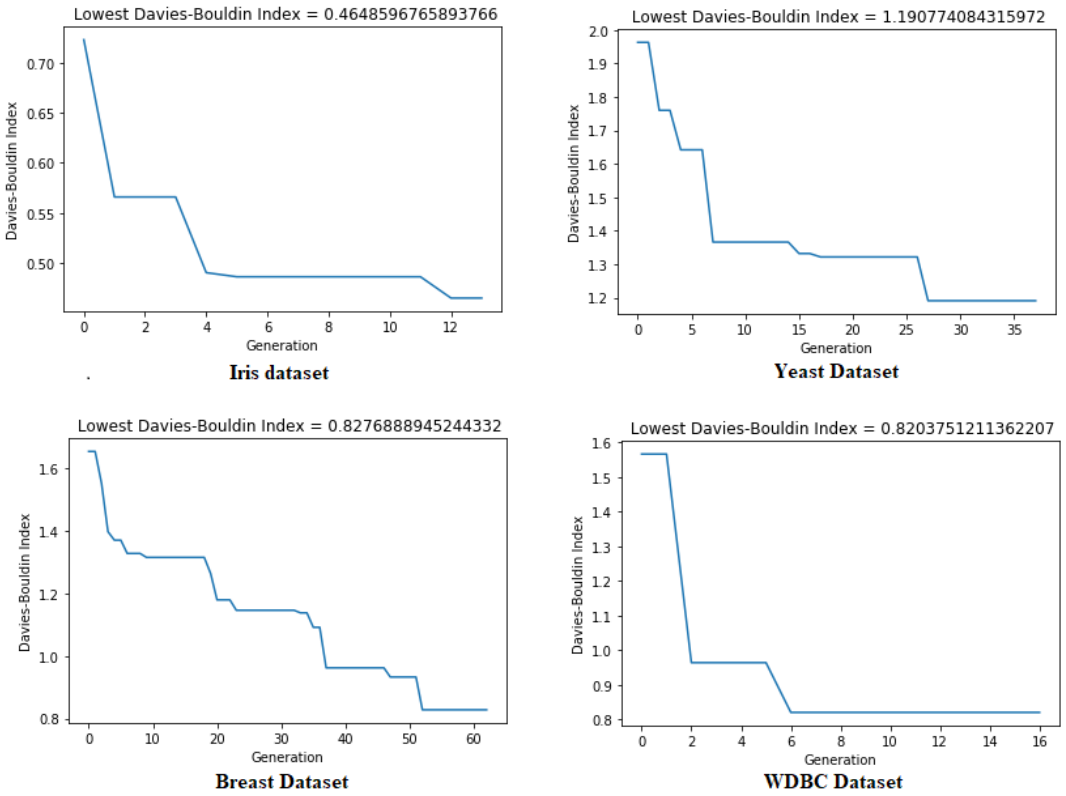

Fig. 4. Optimization process in Different Data sets

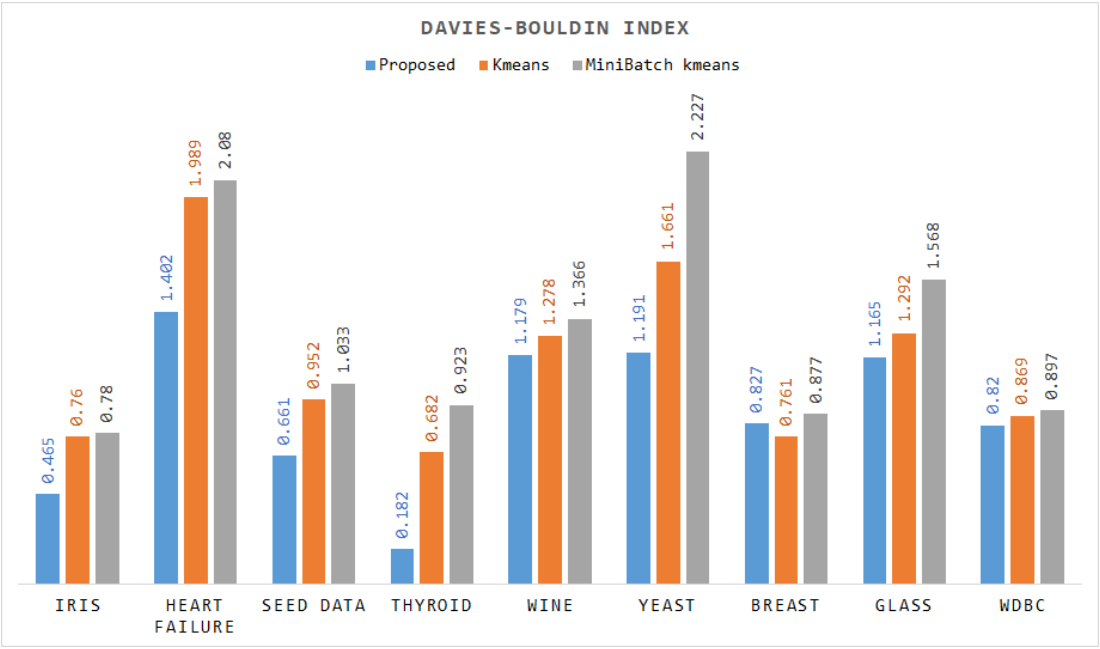

Fig. 5. Davies-Bouldin Index for different dataset

centroids for the clusters easily.

V. Conclusion

In this work, a meta-heuristic based optimization technique has been proposed to find the best centroids of the clusters for a given dataset. This technique improves the performance of the standard k-means algorithm. Nine diversified real data sets were used to test the performance of the proposed technique. For all the datasets the superior performance is found in terms of the davies-bouldin index which represents the quality of clustering. Therefore, this technique will be useful in terms of clustering multi-dimensional data. However, more meta-heuristic techniques can be incorporated to tune the optimization process.

References

[1] S. J. Nanda and G. Panda, "A survey on nature inspired metaheuristic algorithms for partitional clustering," *Swarm and Evolutionary computation*, vol. 16, pp. 1–18, 2014.

[2] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[3] W.-L. Zhao, C.-H. Deng, and C.-W. Ngo, "k-means: A revisit," *Neurocomputing*, vol. 291, pp. 195–206, 2018.

[4] A. Vattani, "The hardness of k-means clustering in the plane," *Manuscript, accessible at http://cseweb. ucsd. edu/avattani/papers/kmeans_hardness. pdf*, vol. 617, 2009.

[5] J. Melechovskỳ, C. Prins, and R. W. Calvo, "A metaheuristic to solve a location-routing problem with non-linear costs," *Journal of Heuristics*, vol. 11, no. 5-6, pp. 375–391, 2005.

[6] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian informatics journal*, vol. 16, no. 3, pp. 275–295, 2015.

[7] A. Salehipour, M. Modarres, and L. M. Naeni, "An efficient hybrid meta-heuristic for aircraft landing problem," *Computers & Operations Research*, vol. 40, no. 1, pp. 207–213, 2013.

[8] F. B. Ashraf and M. S. R. Shafi, "Mfea: An evolutionary approach for motif finding in dna sequences," *Informatics in Medicine Unlocked*, vol. 21, p. 100466, 2020.

[9] S. Harifi, M. Khalilian, J. Mohammadzadeh, and S. Ebrahimnejad, "Emperor penguins colony: a new metaheuristic algorithm for optimization," *Evolutionary Intelligence*, vol. 12, no. 2, pp. 211–226, 2019.

[10] S. Harifi, J. Mohammadzadeh, M. Khalilian, and S. Ebrahimnejad, "Giza pyramids construction: an ancient-inspired metaheuristic algorithm for optimization," *Evolutionary Intelligence*, pp. 1–19, 2020.

[11] K. P. Sinaga and M.-S. Yang, "Unsupervised k-means clustering algorithm," *IEEE Access*, vol. 8, pp. 80 716–80 727, 2020.

[12] G. V. Oliveira, F. P. Coutinho, R. J. Campello, and M. C. Naldi, "Improving k-means through distributed scalable metaheuristics," *Neurocomputing*, vol. 246, pp. 45–57, 2017.

[13] D. Raval, G. Raval, and S. Valiveti, "Optimization of clustering process for wsn with hybrid harmony search and k-means algorithm," in *2016 International Conference on Recent Trends in Information Technology (ICRTIT)*. IEEE, 2016, pp. 1–6.

[14] L. Bottou and Y. Bengio, "Convergence properties of the k-means algorithms," in *Advances in neural information processing systems*, 1995, pp. 585–592.

[15] M. Laszlo and S. Mukherjee, "A genetic algorithm that exchanges neighboring centers for k-means clustering," *Pattern Recognition Letters*, vol. 28, no. 16, pp. 2359–2366, 2007.

[16] K. Krishna and M. N. Murty, "Genetic k-means algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 3, pp. 433–439, 1999.

[17] B. Shboul and S. H. Myaeng, "Initializing k-means using genetic algorithms," in *International conference on computational intelligence and cognitive informatics (ICCICI 09)*, 2009, pp. 114–118.

[18] C. Pizzuti and N. Procopio, "A k-means based genetic algorithm for data clustering," in *International Joint Conference SOCO'16-CISIS'16-ICEUTE'16*. Springer, 2016, pp. 211–222.

[19] C. Ratanavilisagul, "A novel modified particle swarm optimization algorithm with mutation for data clustering problem," in *2020 5th International Conference on Computational Intelligence and Applications (ICCIA)*. IEEE, 2020, pp. 55–59.

[20] D. Karaboga and C. Ozturk, "A novel clustering approach: Artificial bee colony (abc) algorithm," *Applied soft computing*, vol. 11, no. 1, pp. 652–657, 2011.

[21] C. Zhang, D. Ouyang, and J. Ning, "An artificial bee colony approach for clustering," *Expert systems with applications*, vol. 37, no. 7, pp. 4761–4767, 2010.

[22] H. Xie, L. Zhang, C. P. Lim, Y. Yu, C. Liu, H. Liu, and J. Walters, "Improving k-means clustering with enhanced firefly algorithms," *Applied Soft Computing*, vol. 84, p. 105763, 2019.

[23] C. Mageshkumar, S. Karthik, and V. P. Arunachalam, "Hybrid meta-heuristic algorithm for improving the efficiency of data clustering," *Cluster Computing*, vol. 22, no. S1, p. 435–442, 2018.

[24] R. Ahmadi, G. Ekbatanifard, and P. Bayat, "A modified grey wolf optimizer based data clustering algorithm," *Applied Artificial Intelligence*, vol. 35, no. 1, pp. 63–79, 2021.