Opposition Learning Based Improved Bee Colony Optimization (OLIBCO) Algorithm for Data Clustering

Srikanta Kumar Sahoo[1], Priyabrata Pattanaik[2], Mihir Narayan Mohanty[3], Dilip Kumar Mishra[4]
Institute of Technical Education and Research, SOA deemed to be University, Bhubaneswar, India[1, 2, 3, 4]

***Abstract*—Clustering of data in case of data mining has a major role in recent research as well as data engineers. It supports for classification and regression type of problems. It needs to obtain the optimized clusters for such application. The partitional clustering and meta-heuristic search techniques are two helpful tools for this task. However the convergence rate is one of the important factors at the time of optimization. In this paper, authors have taken a data clustering approach with improved bee colony algorithm and opposition based learning to improve the rate of convergence and quality of clustering. It introduces the opposite bees that are created using opposition based learning to achieve better exploration. These opposite bees occupy exactly the opposite position that of the mainstream bees in the solution space. Both the mainstream and opposite bees explore the solution space together with the help of Bee Colony Optimization based clustering algorithm. This boosts the explorative power of the algorithm and hence the convergence rate. The algorithm uses a steady state selection procedure as a tool for exploration. The crossover and mutation operation is used to get balanced exploitations. This enables the algorithm to avoid sticking in local optima. To justify the effectiveness of the algorithm it is verified with the open datasets from the UCI machine learning repository as the benchmark. The simulation result shows that it performs better than some benchmark as well as recently proposed algorithms in terms of convergence rate, clustering quality, and exploration and exploitation capability.**

Keywords*—Bee colony optimization; BCO based clustering; data clustering; partitional clustering; meta-heuristic search

I. Introduction

Clustering is one of several notable research areas in data mining. It is an art of grouping elements with a goal that elements in a cluster are dynamically similar compared to the elements of other clusters [1]. There are two significant variants of clustering principles, hierarchical and partitional clustering [1,2,3]. The key issue with hierarchical clustering is once the cluster arrangements are made it can't be altered, as a result reallocation of elements is difficult. In this paper, our focus is on partitional clustering. k-Means, k-Medoids, PAM, and CLARA [4] are some examples of partitional clustering techniques. Because of simplicity and relatively low time complexity, the k-Means algorithm has gain popularity over the years. But the results of the k-Means clustering largely depend on the initially chosen centroids, and it usually converges to local optima. It is also unable to handle higher dimensional datasets.

To address this issue, several optimization techniques have been proposed and found to be effective in this respect. Over the last two decades several optimization based clustering techniques proposed for different applications. Following are some of them. The clustering applications based on Ant Colony Optimization (ACO) introduced in [5,6,7] are based on the foraging behavior of artificial ants. Particle Swarm Optimization (PSO) based clustering algorithms that mimic a flock of swarming particles' food-finding behavior are presented in [8,9,10,11,12]. In [13,14,15,16,17], the Bee Colony Optimization (BCO) based clustering methods are proposed. These are modeled on the way artificial bees collect food. The clustering methods based on Genetic Algorithms (GA) proposed in [18,19,20] are motivated by biological processes as crossing, mutation, and inheritance.

Despite the presence of several optimization techniques for clustering new techniques are encouraged because every technique has its pros and cons. For example; ACO has slow convergence, falls in local optima, low similarity and high time complexity [21]; PSO depends on predefined cluster centroids and is trapped in local optima for higher dimensional datasets [22]; BCO has a low convergence rate and imbalanced exploration and exploitation [23]; and the slowness of genetic algorithms (GA) is frequently questioned. Again, no technique guarantees proper grouping for a wide variety of applications. The overall study of the algorithms concludes that clustering a wide verity of applications is still an open problem and new algorithms are always appreciated. The new meta-heuristic techniques for the clustering problem should address some of the common issues such as dependency on the initial cluster centroids, slow convergence rate, more emphasis on exploration than exploitation, falling in local optima of larger datasets, low accuracy, and higher computational complexity. In recent years, many optimization techniques for clustering problem proposed that successfully addressed these issues. Some of these are discussed in related works section.

The rate of convergence of a clustering algorithm mainly depends on the initially chosen centroids. The convergence rate is faster if these initially chosen centroids are closer to the optimal centroids. Otherwise, it takes a considerably large time to converge. To improve the convergence rate and get better clustering results keeping a balance between exploration and exploitation in this paper, we have proposed an Opposition Learning based Improved Bee Colony Optimization (OLIBCO) algorithm for data clustering. The OLIBCO algorithm considers an opposite set of bees along with the initially chosen mainstream bees. First a set of mainstream bees initialized with randomly selected centroids, then the corresponding opposite bees are created with opposite centroids using opposition based learning technique. Both the mainstream bees and opposite bees explore the solution space together using Bee Colony Optimization based clustering algorithm. This intensifies the exploration, thereby helps in finding better cluster centroids at an early stage of the solution. Every bee generates its cluster in *K* (number of clusters) stages. In every stage, once a feasible solution found the bees come back to hive to compare their solution, thereby recruiter bees recruit their followers and go to next stage. After all stages complete the best solution found and remaining data objects allocated to this and referred as local best. The local best considered to be the global best if it is better than the previous global best. The use of opposition based learning helps in getting faster convergence and the crossover and mutation operations applied to the local best helps in further exploitation of the result.

The rest part of the paper is organized as follows. The related works is discussed in Section II. Section III presents the proposed algorithm with complexity analysis. Section IV shows the simulation and comparative analysis of the algorithm for different applications. Finally, Section V presents the conclusion and future work.

II. Related Works

Several new optimization techniques for the clustering problem have been proposed over the years. The Ant Colony Optimization (ACO) [24] is based on the foraging behavior of artificial ants. In [5], an energy-efficient clustering routing method based on the enhanced ACO algorithm introduced to address the issue of energy consumption in UWSNs. For no-wait flow shop scheduling, a hybrid ant colony algorithm based on crossover and mutation mechanisms is developed with the goal of minimizing the maximum completion time in [6]. To find the ideal CH for an energy-efficient routing protocol in WSN the Ant Colony Optimization (ACO) integrated Glowworm Swarm Optimization (GSO) technique (ACI-GSO) proposed in [7]. The Particle Swarm Optimization (PSO) is a population-based heuristic search method proposed in [25]. In order to achieve flawless clustering with balanced load and energy-efficient optimization, the authors in [8] used a GWO-PSO-based clustering method. For Cooperative PSO based clustering, a new initialization approach has been put forth in [9]. The suggested data clustering strategy in [10] is based on the KDE and PSO clustering algorithm resolves the issues of PSO-based clustering approaches. The work in [11] illustrates a secure technique for clustering using Improved Particle Swarm Optimization Algorithm in IoT. To get the best network performance, a method based on GA and PSO is proposed in [12] for the CH selection and to optimize the sink mobility.

The Bee Colony Optimization (BCO) is another meta-heuristic algorithm inspired by the food collection strategy of honey bee swarms [26]. An effective method for regularly finding, aggregating, analyzing, and managing important data on potential patients collected from the internet of medical things is shown in [13]. The research [14] proposes a revolutionary artificial bee colony approach for data clustering. BCO+KM clustering is a hybrid data clustering algorithm, which combines BCO and K-means proposed in [15]. In [16], a data clustering approach proposed called Modified BCO algorithm where a bee that is with a better fitness receives the high preference. The study [17] proposes an enhanced bee colony optimization algorithm with a document clustering application. The work in [27] looks into the multi-objective system reliability optimization problem using fuzzy parameters. In recent years some other optimization techniques proposed for different task. These include: Whale Optimization Algorithm (WOA) [28,29], Cuckoo Search Optimization (CSO) [30,31], Kho-Kho optimization algorithm [32], P-spline based clustering [33], Moth-Flame Optimization (MFO) algorithm [34], Brain Storm Optimization (BSO) algorithm [35], and Class Topper Optimization (CTO) algorithm [36].

Opposition based learning is another helpful technique in the field of optimization. There are several researches done in the literature using opposition based learning for optimization of different tasks. Some of them are given here. In [37], an approach discussed for dividing the population into several memeplexes using shuffled differential evolution. To improve the convergence, the authors in [38], used this technique with harmony search meta-heuristic optimization. In [39], the authors have employed opposition-based learning to break down a multi-objective optimization problem into a number of scalar optimization sub-problems, which they are then simultaneously optimizing with the evolutionary algorithm. To overcome the limitations of PSO, the authors in [40], have utilized the generalized opposition based learning. For data clustering problems, the authors have used Chimp Optimization Algorithm, Generalized Normal Distribution Algorithm, and Opposition-Based Learning method in [41]. In [42], the authors have proposed an augmented arithmetic optimization technique with the help of levy flight distribution and opposition based learning for data clustering. In [43], authors have combined it with chaotic maps, and PSO for text clustering.

III. Proposed OLIBCO Algorithm for Data Clustering

A. Mapping of OLIBCO for Clustering Problem

The Bee Colony Optimization (BCO) algorithm [26] considers three kinds of bees, namely scout bees, employed bees, and onlooker bees. First, the scout bees move around the search space looking for food sources. After such sources found, the scout bees become employed bees; they evaluate the quality of the found food sources (the nectar amount); and back to their hive. The employed bees go to the dance floor one by one and show others the quality of the food source they discovered (through the waggle dance). This way, every employed bee recruits some onlooker bees and travel to the same source for food collection. Once the food from the source exhausted all these bees become scout bees. The complete set of possible food sources that artificial bees can explore represents the solution space. A bee moves in the solution space and discovers a food source as a candidate solution. Again, we know that the ultimate goal of a clustering problem is to find a set of mutually exclusive groups called clusters. Hence, a bee discovers a candidate solution (a set of clusters). That means every bee has a set of clusters and a set of centroids as its identifier. For a multi-dimensional dataset containing *K* different classes of data objects, a clustering solution is a set of *K* clusters identified by *K* centroids *(C_1,C_2,...,C_K)*, where each C_i is an *m*-dimensional data object in the dataset. In OLIBCO, it is assumed that all the bees are of a similar category. Every bee goes out for discovering feasible solutions at the same time. They select some data objects from the dataset and add them to their respective clusters. After coming back to their hive they compare the solutions using an estimate 'Sum of Intra Cluster Distances' (*SICD*). Thus, they find the local best solution. The *SICD* can be computed as follows [16,17]:

$$SICD = \sum_{i=1}^{k} \sum_{j=1}^{n} d(c_i, o_j)\ , \tag{1}$$

where *d(c,o)* is Euclidean distance between two *n*-dimensional data objects *c* and *o*. The terms *k*, *n*, *c*, and *o* represent the number of clusters, number of data objects in each cluster, the cluster center and the data objects respectively. Finally, the local best is compared with the global best, update the global best if required, and continue to next iteration.

B. Proposed OLIBCO Clustering Algorithm

The OLIBCO is a population-based meta-heuristic optimization algorithm inspired by opposite bees. The opposite bees are capable to explore on the opposite side of the mainstream bees. They play an important role in improving the explorative power of the algorithm. The exploration is again powered by a steady-state algorithm which adds better particles into different clusters. The algorithm uses a probability-based selection approach to give every bee (both mainstream and opposite bees) a fair chance to become a recruiter, that provides diversity in solution. For further enhancement in exploitation, the algorithm uses the crossover and mutation concept in the local best solution. The high level flow diagram is shown in Fig. 1. The algorithm is presented in Algorithm 1 and steps are explained in detail below.

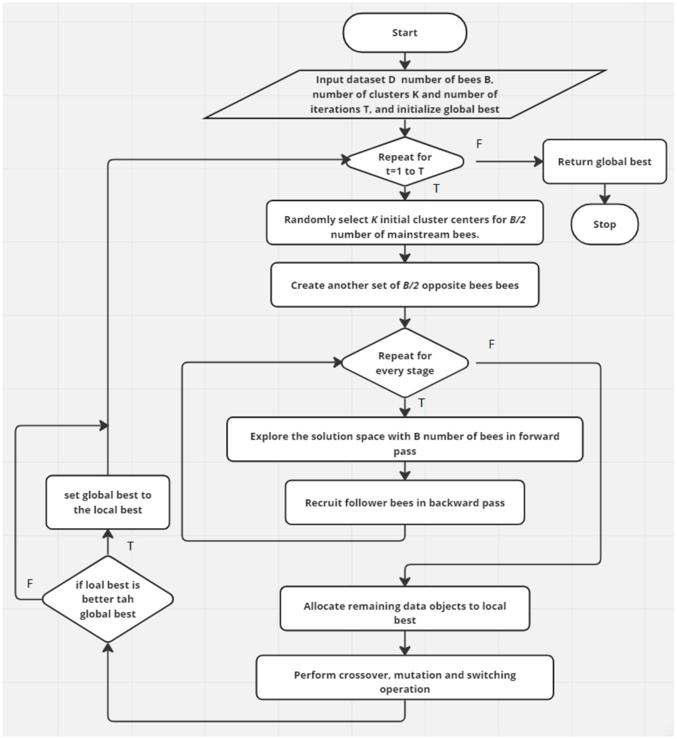
Start
Input dataset D number of bees B, number of clusters K and number of iterations T, and initialize global best
Repeat for t=1 to T
F
T
Return global best
Stop
Randomly select K initial cluster centers for B/2 number of mainstream bees.
Create another set of B/2 opposite bees bees
Repeat for every stage
F
T
Explore the solution space with B number of bees in forward pass
Recruit follower bees in backward pass
set global best to the local best
T
F
if loal best is better tah global best
Allocate remaining data objects to local best
Perform crossover, mutation and switching operation

Fig. 1. Flow diagram of OLIBCO.

1) Initialization: At first, the algorithm loads the data set D (an n × m matrix) into memory. Here, n is the number of data objects and m is the number of attributes. It takes number of bees (B), number of clusters (K), and maximum number of iteration (T) as input. The total number of bees B includes the number of mainstream and opposite bees both in an equal proportion. Every bee maintains K number of empty clusters in it. The algorithm initializes K random cluster centroids and finds its SICD value. In first step, it initializes B/2 number of mainstream bees with initial centroids. To avoid repetition in the data allocation it internally maintains an allocate array of size n. It is not an overhead for the algorithm because for setting the fields in an array take constant time.

2) Creation of opposite bees: Meta-heuristic optimization algorithms tend to make initial random guesses. The OLIBCO algorithm also initializes the mainstream bees randomly. If these mainstream bees span throughout the solution space then no need to worry. But there is a great possibility that they belong to a specific region of the solution space. If this is the case then exploration may confine to the same region or may not cover the complete solution space. In such cases, either the convergence rate is reduced or the solution evolves in a different direction and never comes near to optimal solution.

Considering a large number of mainstream bees can be an idea to this end. But we have to keep in mind that more number of bees implies more exploration. But, more weightage on exploration does not help always; instead, it increases the time complexity. Therefore, we have to choose the number of bees carefully to keep balance between the exploration quality and exploration time. Again, we cannot give guaranty that all these bees will not be in the same area, because these are created randomly.

To address this issue OLIBCO creates an equal number of opposite bees in a different direction (that of mainstream bees). Fig. 2 displays two different scenarios, first Fig. 2(a) shows a case where all are mainstream bees and are chosen in the same region, and second Fig. 2(b) shows some mainstream bees and their opposite bees together span throughout the solution space. In the first case, the exploration confined to one part of the solution space, because the general tendency of a bee is to explore nearby regions. In the second case, no matter where the mainstream bees are allocated, the corresponding opposite bees will cover the other region. By this way, the bees (mainstream and opposite bees) span throughout the solution space.

The OLIBCO algorithm creates the opposite bees after the initialization of mainstream bees using opposition based learning [44, 45]. A cluster center is a data vector from a dataset with m attribute. For all these attributes, the opposite values are computed using (2) [44, 45].

$$p' = a + b - p, \tag{2}$$

where, p is a real number in the range *[a,b]* and p' is it's opposite number. Once the opposite values found, we combine them to a data vector to represent a centroid. For example,

Algorithm 1 OLIBCO Clustering Algorithm

Require: Maximum number of iterations T, number of clusters K, number of artificial bees B, and dataset D, with n objects, each of them are of m dimensional

Ensure: Classified objects as clusters

```
1: Initialize all the $B$ bees with $K$ number of empty clusters and initialize Global Best (GB).
2: for $t = 1$ to $T$ do
3:    if t = 1 then
4:        Initialize $K$ random cluster centroids, and find $SICD$ value for it.
5.    else
6.        Consider the centroids of the global best solution.
7:    end if
8:    Randomly select $K$ initial cluster centers for $B/2$ number of mainstream bees.
9:    Create another set of $B/2$ opposite bees bees
10:   for $k = 1$ to $K$ do
11:      for $b = 1$ to $B$ do
12:          Use the steady state selection procedure to select a population of $x$ data objects
13:          Assign these data objects to the $k^{th}$ cluster of the $b^{th}$ bee
14:          Calculate $SICD$ value of the $b^{th}$ bee.
15:      end for
16:      for $b = 1$ to $B$ do
17:          Find the probability of stickiness ($P$) for the $b^{th}$ bee
18:          Generate a random number $r$ in the range $0$ to $1$.
19:          if $P < r$ then
20:              Select a recruiter using roulette wheel selection technique and follow its solution.
21:          end if
22:      end for
23:      Find the best partial solution (bee clusters with minimum $SICD$ value).
24:   end for
25:  Allocate the left-over data objects to the local best solution, and update centroids.
26:    Find the $SICD$ value of the local best solution.
27:  if Global best solution does not change for an adequate number of iterations $t$ then
28:  Perform crossover and mutation operation on the centroids of the local best solution, and find the $SICD$ value.
29:     Update the local best solution if the new result is better than the previous, otherwise discard the new result.
30:   end if
31:     Perform switching operation of data objects between the clusters of local best solution.
32:   if $t \neq 1$ then
33:      if $SICD$ value at iteration $t < SICD$ value at iteration t-1 then
34:          Update the global best solution with the
```

```
results of the local best solution.
35:         end if
36:     end if
37: end for
38: Return the global best solution.
```

Consider the following mainstream bee and its opposite bee defined with three cluster centers.

$$Bee_{mainstream} = ((c_{11}, c_{12}, ..., c_{1m}),(c_{21}, c_{22}, ..., c_{2m}), ...;$$

$$Bee_{opposite} = ((c'_{11}, c'_{12}, ..., c'_{1m}),(c'_{21}, c'_{22}, ..., c'_{2m}), ...;$$

Here, c_{ij} and c'_{ij} are the attributes of the centroids of mainstream bees and their opposite bees, that represents the j^{th} attribute of the i^{th} cluster center. Every c'_{ij} is generated using (2). Similarly, for all the centroids of each bee are found to create the corresponding opposite bees. Moreover, the number of mainstream bees, doubles the total number of bees. Therefore, we cannot create large number of mainstream bees initially. We have taken the total number of bees (*B*) as input parameter. First, *B/2* number of mainstream bees initialized then another set of *B/2* opposite bees generated.

The algorithm runs for *K* (*k=1,2,...,K*) number of clusters. In each iteration, all the bees build one cluster (k^{th}) at a time. So there is an inner loop that varies from (*b=1* to *B*). For every bee *b*, the forward and backward passes generate the feasible solutions. After the outer loop (*k* loop) ends all the bees have a partial solution. Here each iteration of the outer loop (*k* loop) is considered as a stage.

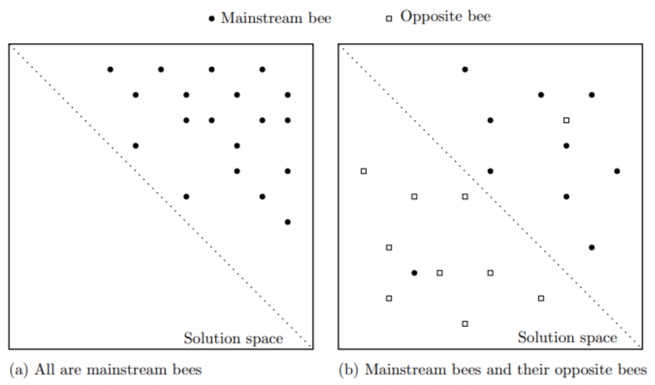
● Mainstream bee
▫ Opposite bee
Solution space
Solution space
(a) All are mainstream bees
(b) Mainstream bees and their opposite bees

Fig. 2. Two different scenarios where (a) all bees are main stream bees chosen in same region (worst case) and (b) some mainstream and their opposite bees.

3) Forward pass: Once the mainstream bees initialized and their opposite bees created, all of them start exploring the solution space. In this process, they select some data objects from the dataset D and add them into their respective clusters based on a fitness (SICD) value. The exploration process of each bee depends on this selection procedure. For this, the OLIBCO algorithm uses a steady state selection approach [35]. The steady state searching process starts with creating an initial population of x data objects. This step repeats for M number of times. Here, M represents the number of moves. In each move, the search process refines the data objects to create a better population of x data objects. More the number of moves (M), better the exploration. By increasing M the time complexity also increases. Again too small value of M implies less exploration that leads to slow convergence rate, so, it must be chosen carefully. After the forward pass completes, each bee has a feasible solution (a set of clusters). Now, all of them compute the strength of their solution using the SICD measure defined in (1).

4) Backward pass: In the backward pass, bees return to their hive, and share the information about their generated solution. Depending on the strength of the solution, the bee determine whether it will continue with its solution and be available as a recruiter, or will adopt (follow) some other's solution. The following equation defines the probability of a bee sticking to its solution [16,17].

$$P_b(k+1,t) = e^{-O_b(k,t)/(k\times t)}, \quad (3)$$

where *b* represents the current bee, *k* represents stage, and *t* represents iteration. $O_b(k,t)$ represents the normalized value of *SICD* of b^{th} bee and is computed using the equation (4) [16,17].

$$O_b(k,t) = \frac{SICD_b(k,t) - SICD_{min}(k,t)}{SICD_{max}(k,t) - SICD_{min}(k,t)}, \quad (4)$$

where $SICD_b$ is the *SICD* value of current bee *b*, $SICD_{max}$, and $SICD_{min}$ represents the maximum and minimum value of the *SICD*. From (3), we can observe *P* (probability of stickiness) and O_b are inversely proportional to each other, and the value of O_b depends on $SICD_b$. That means, when O_b increases the probability of a bee sticking to its solution decreases. The bee with smaller probability *P* may discard its solution and become a follower. If decision is to become a follower, then the bee will select another bee and adopts (copy) its solution. Every follower always wants to follow the best. This tendency of follower bees leads to local optima. The OLIBCO algorithm uses roulette wheel selection procedure [46] to avoid this. This gives every bee a fair chance to become a recruiter. It also gives diversity to the solution. This selection procedure first labels the surface of the wheel based on the proportion of the *SICD* values of each bee. Then the wheel spun. After it stops, the wheel pointer decides the result. Finally, the backward pass computes the *SICD* values of all the bees using (1).

5) Allocation of remaining data objects: After all the stages are complete, each bee has a partial clustering solution. But there is a possibility that some data objects are not allocated to any of the partial solutions. These left-over (unallocated) data objects are allocated to the best partial solution (among all the bees) using a single pass of the k-Means algorithm [47]. It takes the centroids of the best partial solution as initial cluster centroid and allocates the left-over data objects to the nearest cluster using the distance metric d. It also updates the centroids to the best candidate centroids. Note that we have used only one iteration of k-Means.

6) Crossover and mutation operation: The opposite bees and steady state selection procedure provides faster convergence. But after this, there is a possibility that the solution does not update for a longer period of time. It may have been trapped in a local optima. To take it out from the trapped region (if there is any possibility) the algorithm applies the crossover and mutation [48] operation on the cluster centroids. Moreover, swapping should be between similar type attributes. We have used a one point crossover for swapping later half m/2 attributes of the consecutive centroids. We can do this swapping in any combination. As part of mutation operation, some or all the attributes of a centroid is substituted by another set of closer values. The main aim of the mutation step in our algorithm is to move the centroids in nearby locations to check the existence of any better solution. After the new centroids generated the SICD of the solution is computed again. If this new solution is better than the current local best, then the local best solution is updated.

7) Switching operation of data objects: In the previous steps at the allocation of remaining data objects and crossover and mutation, somehow the centroids are updated. As the data objects are assigned to the specific cluster based on Euclidian distance with the centroids, after centroids update, there may be some data objects wrongly placed in a cluster. The switching operation of data objects allows them to move to other clusters where its strength is better than the current one. The chance of switching is more for the data objects located in the boundary region of the cluster. This switching operation helps in further improvements in the result, thereby contributes to the faster convergence rate of the algorithm.

8) Global best update: The algorithm runs for T number of iterations. In each iteration, it finds a solution (local best). If the SICD value of the solution at iteration t ($SICD_t$) is less than the SICD value of the global best solution at iteration t-1 ($SICD_{t-1}$) then it updates the global best solution to the solution found in the current iteration. Then it continues to the next iteration with updated global best.

C. Asymptotic Analysis of OLIBCO Clustering Algorithm

The time complexity of an algorithm is a function of input parameters. This function can be generated by computing the sum of the number of times each step of the algorithm executes. Table I shows the step counts for each individual steps of the OLIBCO clustering algorithm. Thus, the time complexity function of the algorithm is:

$$f(T, K, B) = (Mx + x + 10)TKB + 7TK + (11 + c_1)T + 1$$

We can observe that in step 12 a factor of Mx multiplied with TKB. This is because in the forward pass the bees create a population of x (fixed) data objects. For this, they are allowed to make M number of moves in the solution space. With an increased number of moves M, the exploration time also increases, without much difference in the strength of the solution (after a certain point). Keeping this in mind, we have fixed the number of moves M. In step 25, a constant factor c_1 is multiplied; because the number of left-over data objects is unknown. As most of them are assigned to different clusters of different bees in the forward pass, it must be in small numbers (so considered as a constant). The step 28 performs crossover and mutation. As it operates between K cluster centers, both these operations step count is TK and for $SICD$ computation it is T. Now, the time complexity function can be rewritten as:

$$f(T, K, B) = y_1TKB + 7TK + y_2T + 1$$

Clearly, the highest growing term in this function is TKB. Hence, ignoring the constant factor the time complexity of the algorithm is $O(TKB)$. Moreover, the data set dimensions $(n \times m)$ also has an important role, which is not considered above. Table I shows the step counts without considering dimension of the dataset n and m. Considering the dataset dimensions the time complexity of the algorithm becomes $O(TKBmn)$.

TABLE I. STEP COUNTS OF THE OLIBCO ALGORITHM

Step Number	Count	Step Number	Count
1	1	18	$T \times K \times B$
2	T+1	19	$T \times K \times B$
3	T	20	$T \times K \times B$ (worst case)
4	1	23	$T \times K$
6	T-1	25	$T \times c_1$
8	$T \times K \times B/2$	26	T
9	$T \times K \times B/2$	27	T
10	$T \times (K + 1)$	28	$T \times (2K + 1)$ (worst case)
11	$T \times K \times (B + 1)$	29	T (worst case)
12	$T \times K \times B \times (M \times x)$	31	$T \times K$
13	$T \times K \times B \times x$	32	T
14	$T \times K \times B$	33	$T - 1$
16	$T \times K \times (B + 1)$	34	$T - 1$
17	$T \times K \times B$	38	1

IV. SIMULATION AND COMPARATIVE ANALYSIS

The algorithm is implemented in Java with required parameters D, B, K and T. To realize the performance of the OLIBCO algorithm, we have analyzed it concerning different applications. The benchmark datasets from the UCI machine learning repository used for analysis purposes are shown in Table II. Validation of the performance is done through an analysis of $SICD$ values for different applications by varying the number of bees B and the number of moves M, followed by a comparative study of $SICD$ values with some existing optimization techniques for clustering. The results presented here are the average of 25 random instances of executions.

TABLE II. SPECIFICATION OF THE DATASETS USED

Datasets	Number of clusters	Number of attributes	Number of data objects
Iris	3	4	150 (50, 50, 50)
Glass	6	9	214 (70, 76, 17, 13, 9, 29)
Cancer	2	9	683 (444, 239)
CMC	3	9	1473 (629, 334, 510)

TABLE III. SICD VALUES BASED ON DIFFERENT NUMBERS OF BEES B0

Application	Number of bees	SICD
Iris	4	97.10
	10	96.90
	14	96.77
	18	96.80
Glass	4	226.19
	10	224.09
	14	224.67
	18	224.34
Cancer	4	2985.65
	10	2980.32
	14	2977.09
	18	2983.93
CMC	4	5682.67
	10	5676.97
	14	5678.07
	18	5684.89

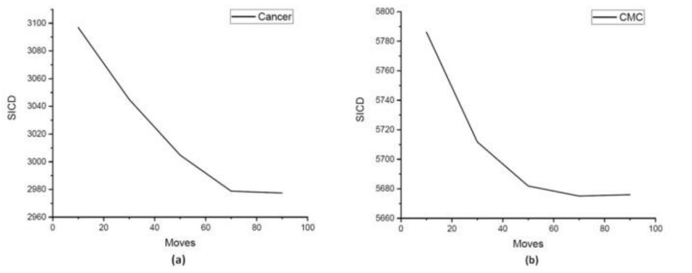

Fig. 3. SICD analysis of (a) Cancer dataset, (b) CMC dataset by varying number of moves.

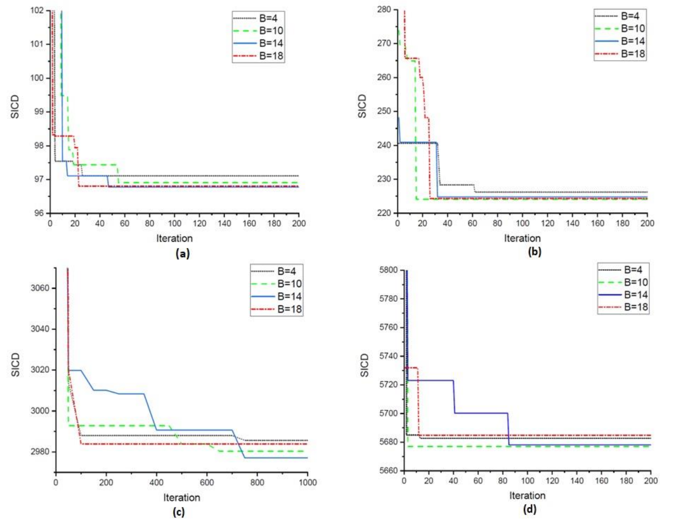

Fig. 4. Convergence graph of (a) Iris, (b) Glass, (c) Cancer, (d) CMC datasets.

Table III shows the *SICD* analysis for different applications on varying *B*. We found that the selection of the number of bees has a large impact on the *SICD* value of the clusters. A small number of bees indicate less exploration. That means there is a chance that some areas of solution space will remain unexplored, which may give poor results. This fact can be observed from the table when *B=4*. On the other hand, a large number of bees indicate too much exploration. Though the increased number of bees provides diversity, it comes with increased exploration time. Again it may not improve the results drastically (Table III, *B=18*). Therefore, compromising time complexity is not a good idea.

The steady-state selection procedure provides exploration power to the bees. Here in every move, the bee tries to improve its generated solution strength by placing a weaker data object with another stronger data object. Thus, the number of moves *M* (that a bee does) has also a huge impact on the algorithm's performance. Less number of bees has to make a large number of moves to explore the solution space whereas a large number of bees can handle it with comparatively less number of moves. Fig. 3 presents the *SICD* values of the generated clustering solution of the Cancer dataset by varying number of moves and keeping the number of bees (*B*) and initial population size (*x*) fixed. From the implementation results, we observed that when *M* is small the generated *SICD* value is high. With increased *M* value the *SICD* decreases and after a certain number of moves, it does not change much. The larger the initial population size *x*, the number of moves required is more. Depending on the dataset used the value of *x* can vary.

In optimization algorithms, the convergence rate is an important factor to analyze the performance. Fig. 4 displays the convergence analysis of the algorithm made for datasets used by varying the number of bees. Fig. 4 also shows that after the initial faster convergence the graph remains constant. The reason is that it is trapped in local optima (for the current centroid, it is the best result). The cross over and mutation operations are used in the algorithm to check this scenario and exploit the result if there is any possibility of improvement. This fact can be observed in B=10, and B=14 of Fig. 4(a), 4(c), B=4 of Fig. 4(b) and B=18 of Fig. 4(d).

Finally, we have made a comparison of the SICD value of the clustering solution for some existing algorithms in Table IV. The OLIBCO algorithm out performs *k*-Means and Classical PSO algorithm for Iris, Glass, Cancer, and CMC datasets except for the best cases of classical PSO of Iris dataset and *k*-Means of Glass dataset. The comparison is also made with IBCOCLUST [17], KIBCLUST [17], Hybrid I [17], Hybrid II [17], MBCO [16], MKCLUST [16], and KMCLUST[16]. It is found that for Iris dataset OLIBCO performs better than IBCOCLUST and the results are near to others. For Glass dataset, the average case SICD is better than IBCOCLUST, KIBCLUST, Hybrid I and MBCO, whereas on best case, it is better than MBCO, MKCLUST and KMCLUST. For Cancer dataset, the OLIBCO gives better results compared to others except average case of IBCLUST and Hybrid II and worst case of MBCO and MKCLUST. Again, for CMC dataset, OLIBCO outperforms MBCO, MKCLUST and KMCLUST.

Here we have observed that for all the applications, the algorithm converges to a certain level quickly, then it keep on exploiting the solution before final convergence result. Clearly, the OLIBCO algorithm has a faster convergence rate. The results show that OLIBCO outperforms some of the existing algorithms in terms of *SICD*. Hence, gives better clustering result.

TABLE IV. A COMPARATIVE ANALYSIS OF SICD VALUES WITH SOME EXISTING ALGORITHMS

Applications	**Measure**	**k-Means [16,17]**	**Classical PSO [16,17]**	**IBCOCLUST [17]**	**KIBCLUST [17]**	**Hybrid I [17]**	**Hybrid II [17]**	**MBCO [16]**	**MKCLUST [16]**	**KMCLUST [16]**	**OLIBCO**
Iris	Best	97.33	96.01	97.22	96.40	96.33	95.10	94.14	95.01	95.19	96.77
	Avg	106.05	100.01	97.27	96.40	96.38	95.14	96.03	96.01	95.98	96.77
	Worst	120.45	117.81	-	-	-	-	104.22	201.00	200.10	97.88
Glass	Best	215.68	270.12	214.85	217.97	214.78	214.71	215.00	215.00	215.23	220.14
	Avg	260.40	289.31	225.19	226.34	226.59	221.50	225.00	220.00	221.00	224.30
	Worst	-	332.00	-	-	-	-	230.00	333.00	332.00	281.17
Cancer	Best	2987.00	2974.14	2976.22	2980.15	2976.24	2976.11	2965.25	2969.01	2971.01	2962.31
	Avg	2988.30	3329.22	2976.89	2980.15	2977.59	2976.24	2990.25	2985.23	2995.43	2977.43
	Worst	3521.50	-	-	-	-	-	3001.01	3076.10	3180.01	3083.83
CMC	Best	5842.20	5694.07	-	-	-	-	5680.12	5678.20	5678.40	5649.76
	Avg	5893.60	5729.11	-	-	-	-	5685.21	5684.80	5684.60	5678.42
	Worst	5934.40	5880.02	-	-	-	-	5798.20	5790.21	5689.70	5750.66

V. CONCLUSION AND FUTURE WORK

In this paper, a new OLIBCO algorithm is proposed as a potential solution for biomedical data clustering. It uses the opposite bees to shield the other directions of mainstream bees. This enhances the quality of the exploration result, thereby improving the convergence rate. The crossover and mutation operations along with the switching operation allow further exploitation of the solution and avoid being stuck in local optima. For validation of the clustering result, we have applied it for different benchmark applications from the UCI machine learning repository. The simulation results show that the algorithm has a faster convergence rate and possible exploitations. It is also observed that the algorithm converges to a certain level within *50* to *100* iterations for all the datasets used. After an initial faster convergence it gives enough chance for exploitations. From the analysis of results, it is clear that by adopting an optimal number of bees B and the number of moves M for exploration the algorithm give better performances. Further from the algorithm it is observed that there is better performance with a similar level of time complexity. The comparison made with different existing algorithms proves the proposed OLIBCO algorithm's efficacy. Further investigations for higher dimensional datasets need to be explored in the future. The algorithm should be tested in different real-life applications of science and technology.

REFERENCES

[1] J. Han, J. Pei, and M. Kamber, Data mining: concepts and techniques. Elsevier, 2011.

[2] A. K. Jain, "Data clustering: 50 years beyond k-means," Pattern recognition letters, vol. 31, no. 8, pp. 651–666, 2010.

[3] Li, Xiaorong, and Zhinian Shu. "Research on Big Data Text Clustering Algorithm Based on Swarm Intelligence." *Wireless Communications and Mobile Computing* 2022 (2022).

[4] R. Xu and D. Wunsch, "Survey of clustering algorithms," IEEE Transactions on neural networks, vol. 16, no. 3, pp. 645–678, 2005.

[5] X. Xiao and H. Huang, "A clustering routing algorithm based on improved ant colony optimization algorithms for underwater wireless sensor networks," Algorithms, vol. 13, no. 10, p. 250, 2020.

[6] O. Engin and A. G¨u¸cl¨u, "A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems," Applied Soft Computing, vol. 72, pp. 166–176, 2018.

[7] D. L. Reddy, C. Puttamadappa, and H. Suresh, "Merged glowworm swarm with ant colony optimization for energy efficient clustering and routing in wireless sensor network," Pervasive and Mobile Computing, vol. 71, p. 101338, 2021.

[8] J. S. Raj, "Machine learning based resourceful clustering with load optimization for wireless sensor networks," Journal of Ubiquitous Computing and Communication Technologies (UCCT), vol. 2, no. 01, pp. 29–38, 2020.

[9] S. Choudhary, S. Sugumaran, A. Belazi, and A. A. A. El-Latif, "Linearly decreasing inertia weight pso and improved weight factor-based clustering algorithm for wireless sensor networks," Journal of Ambient Intelligence and Humanized Computing, pp. 1– 19, 2021.

[10] M. Alswaitti, M. Albughdadi, and N. A. M. Isa, "Density-based particle swarm optimization algorithm for data clustering," Expert Systems with Applications, vol. 91, pp. 170–186, 2018.

[11] Bao, Zhanbiao. "Secure Clustering Strategy Based on Improved Particle Swarm Optimization Algorithm in Internet of Things." *Computational Intelligence and Neuroscience* 2022 (2022).

[12] B. M. Sahoo, H. M. Pandey, and T. Amgoth, "Gapso-h: A hybrid approach towards optimizing the cluster based routing in wireless sensor network," Swarm and Evolutionary Computation, vol. 60, p. 100772, 2021.

[13] E. El-Shafeiy, K. M. Sallam, R. K. Chakrabortty, and A. A. Abohany, "A clustering based swarm intelligence optimization technique for the internet of medical things," Expert Systems with Applications, vol. 173, p. 114648, 2021.

[14] F. Zabihi and B. Nasiri, "A novel history-driven artificial bee colony algorithm for data clustering," Applied Soft Computing, vol. 71, pp. 226–241, 2018.

[15] J. Revathi, V. Eswaramurthy, and P. Padmavathi, "Hybrid data clustering approaches using bacterial colony optimization and k-means," in IOP Conference Series: Materials Science and Engineering, vol. 1070, no. 1. IOP Publishing, 2021, p. 012064.

[16] P. Das, D. K. Das, and S. Dey, "A modified bee colony optimization (mbco) and its hybridization with k-means for an application to data clustering," Applied Soft Computing, vol. 70, pp. 590–603, 2018.

[17] R. Forsati, A. Keikha, and M. Shamsfard, "An improved bee colony optimization algorithm with an application to document clustering," Neurocomputing, vol. 159, pp. 9–26, 2015.

[18] B. N. Chebouba, M. A. Mellal, and S. Adjerid, "Fuzzy multiobjective system reliability optimization by genetic algorithms and clustering analysis," Quality and Reliability Engineering International, vol. 37, no. 4, pp. 1484–1503, 2021.

[19] S. Verma, N. Sood, and A. K. Sharma, "Genetic algorithm-based optimized cluster head selection for single and multiple data sinks in heterogeneous wireless sensor network," Applied Soft Computing, vol. 85, p. 105788, 2019.

[20] M. Alswaitti, M. Albughdadi, and N. A. M. Isa, "Variance-based differential evolution algorithm with an optional crossover for data clustering," Applied Soft Computing, vol. 80, pp. 1–17, 2019.

[21] A. M. Jabbar, K. R. Ku-Mahamud, and R. Sagban, "Ant-based sorting and aco-based clustering approaches: A review," in 2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE). IEEE, 2018, pp. 217–223.

[22] W. Liu, Z. Wang, X. Liu, N. Zeng, and D. Bell, "A novel particle swarm optimization approach for patient clustering from emergency departments," IEEE Transactions on Evolutionary Computation, vol. 23, no. 4, pp. 632–644, 2018.

[23] H. Hakli and M. S. Kiran, "An improved artificial bee colony algorithm for balancing local and global search behaviors in continuous optimization," International Journal of Machine Learning and Cybernetics, pp. 1–26, 2020.

[24] Dorigo, Marco, Mauro Birattari, and Thomas Stutzle. "Ant colony optimization." *IEEE computational intelligence magazine* 1.4 pp. 28-39, 2006.

[25] Kennedy, J., and R. Eberhart. "Particle swarm optimization In: Proceedings of ICNN 95-International Conference on Neural Networks, 1942–1948." IEEE, Perth. https://doi. org/10.1109/icnn (1995).

[26] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," Journal of global optimization, vol. 39, no. 3, pp. 459–471, 2007.

[27] Chebouba, Billal Nazim, Mohamed Arezki Mellal, and Smail Adjerid. "Fuzzy multiobjective system reliability optimization by genetic algorithms and clustering analysis." *Quality and Reliability Engineering International* 37.4, pp. 1484-1503, 2021.

[28] S. M. Bozorgi, M. R. Hajiabadi, A. A. R. Hosseinabadi, and A. K. Sangaiah, "Clustering based on whale optimization algorithm for iot over wireless nodes," Soft Computing, vol. 25, no. 7, pp. 5663–5682, 2021.

[29] Singh, Hakam, et al. "An enhanced whale optimization algorithm for clustering." *Multimedia Tools and Applications* 82.3, pp. 4599-4618, 2023.

[30] S. I. Boushaki, N. Kamel, and O. Bendjeghaba, "A new quantum chaotic cuckoo search algorithm for data clustering," Expert Systems with Applications, vol. 96, pp. 358–372, 2018.

[31] N. Mittal, S. Singh, U. Singh, and R. Salgotra, "Trust-aware energy-efficient stable clustering approach using fuzzy type-2 cuckoo search optimization algorithm for wireless sensor networks," Wireless Networks, vol. 27, no. 1, pp. 151–174, 2021.

[32] A. Srivastava and D. K. Das, "A new kho-kho optimization algorithm: An application to solve combined emission economic dispatch and combined heat and power economic dispatch problem," Engineering Applications of Artificial Intelligence, vol. 94, p. 103763, 2020.

[33] C. Iorio, G. Frasso, A. D'Ambrosio, and R. Siciliano, "A p-spline based clustering approach for portfolio selection," Expert Systems with Applications, vol. 95, pp. 88–103, 2018.

[34] Y. Xu, H. Chen, A. A. Heidari, J. Luo, Q. Zhang, X. Zhao, and C. Li, "An efficient chaotic mutative moth-flame-inspired optimizer for global optimization tasks," Expert Systems with Applications, vol. 129, pp. 135–155, 2019.

[35] F. Pourpanah, Y. Shi, C. P. Lim, Q. Hao, and C. J. Tan, "Feature selection based on brain storm optimization for data classification," Applied Soft Computing, vol. 80, pp. 761–775, 2019.

[36] P. Das, D. K. Das, and S. Dey, "A new class topper optimization algorithm with an application to data clustering," IEEE Transactions on Emerging Topics in Computing, 2018.

[37] Ahandani, Morteza Alinia, and Hosein Alavi-Rad. "Opposition-based learning in the shuffled differential evolution algorithm."*Soft computing* 16.8, pp. 1303-1337, 2012.

[38] Gao, X. Z., et al. "A hybrid optimization method of harmony search and opposition-based learning." *Engineering Optimization* 44.8, pp. 895-914, 2012.

[39] Ma, Xiaoliang, et al. "MOEA/D with opposition-based learning for multiobjective optimization problem." *Neurocomputing* 146, pp. 48-64, 2014.

[40] Wang, Hui, et al. "Enhancing particle swarm optimization using generalized opposition-based learning." *Information sciences* 181.20, pp. 4699-4714, 2011.

[41] Boroujeni, Sayed Pedram Haeri, and Elnaz Pashaei. "A Hybrid Chimp Optimization Algorithm and Generalized Normal Distribution Algorithm with Opposition-Based Learning Strategy for Solving Data Clustering Problems." *arXiv preprint arXiv:2302.08623* (2023).

[42] Abualigah, Laith, et al. "Augmented arithmetic optimization algorithm using opposite-based learning and lévy flight distribution for global optimization and data clustering." *Journal of Intelligent Manufacturing* (2022): 1-39.

[43] Bharti, Kusum Kumari, and Pramod Kumar Singh. "Opposition chaotic fitness mutation based adaptive inertia weight BPSO for feature selection in text clustering." *Applied Soft Computing* 43 (2016): 20-34.

[44] Tizhoosh, Hamid R. "Opposition-based learning: a new scheme for machine intelligence." International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06). Vol. 1. IEEE, 2005.

[45] S. Dhargupta, M. Ghosh, S. Mirjalili, and R. Sarkar, "Selective opposition based grey wolf optimization," Expert Systems with Applications, p. 113389, 2020.

[46] A. Aibinu, H. B. Salau, N. A. Rahman, M. Nwohu, and C. Akachukwu, "A novel clustering based genetic algorithm for route optimization," Engineering Science and Technology, an International Journal, vol. 19, no. 4, pp. 2022–2034, 2016.

[47] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," Pattern recognition, vol. 36, no. 2, pp. 451–461, 2003.

[48] D. E. Goldenberg, "Genetic algorithms in search, optimization and machine learning," 1989.