A hybrid approach based on Artificial Bee Colony and Differential Evolution for Data Clustering

Mira Ait Saada
Department of Computer Science
University M'hamed Bougara of Boumerdes, Algeria
Email: aitsaadamira@gmail.com

Selma Djebili
Department of Computer Science
University M'hamed Bougara of Boumerdes, Algeria
Email: djebili.selma@gmail.com

Ali Berrichi
Department of Computer Science
LIMOSE Laboratory
University M'hamed Bougara of Boumerdes, Algeria
Email: ali.berrichi@univ-boumerdes.dz

***Abstract*—Clustering is one of the most popular data mining techniques. Among the several methods that have been developed to solve clustering problems, there are some approaches that formalize clustering as an optimization problem and use metaheuristics such as PSO (Particle Swarm Optimization) and GA (Genetic Algorithm) to solve it. In this paper, an improved Artificial Bee Colony (ABC) algorithm called K-ABC-DE is proposed. The improvement of ABC is done by hybridizing it with Differential Evolution (DE) metaheuristic and the popular k-means algorithm. The proposed approach is compared with five state-of-the-art algorithms. Each algorithm is run on seven known Benchmark datasets from the UCI Machine Learning Repository. Experimental tests have shown that K-ABC-DE outperforms ABC algorithm and the four other tested algorithms.**

***Index Terms*—Artificial Bee Colony, Differential Evolution, Mutation, K-means, Clustering.**

I. Introduction

Clustering is a mathematical operation that consists of separating a group of objects into classes so that they are as dissimilar as possible and that the objects of each class are as similar as possible. The most popular clustering algorithm is k-means. It is widely used but is very sensitive to initial values and falls easily into local optima. To work around this issue, we can formalize clustering problem as an optimization problem. Some methods called metaheuristics are able to solve such problems without getting stuck into local optima. Among the most known metaheuristics that have been applied to the clustering problem, we have GA [1], PSO [2], ACO [3], DE [4], ABC [5]. ABC is a Swarm Intelligence based algorithm inspired by bees' behvior. It was created by D. Karaboga [6] and has the advantage of having few parameters to set and being flexible enough to be hybridized with other algorithms. In addition, it has shown quite good results in many problems such as clustering.

Nevertheless, ABC has one major disadvantage which is its weak exploitation of solutions. This is why several variants of ABC [7]–[9] have emerged since its creation, with the aim of overcoming its drawbacks. Several of ABC variants have been applied to clustering problem such as [5], [8]–[11]. We propose in this article a hybridization of ABC with DE algorithm, which has a good exploitation of solutions.

II. Clustering Problem

Let $\mathcal{P} = \{P_1, ..., P_n\}$ be a set of n patterns each with d features. These patterns can also be represented by a matrix $X_{n\times d}$ having n vectors X_i of size d. The vector X_i corresponds to the pattern P_i of the set $\mathcal{P}$ and each element $e_{i,j}$ of X_i corresponds to the value of the j^{th} feature of the pattern P_i.

Given a $X_{n\times d}$ matrix, a clustering algorithm will try to find a partition $\mathcal{C} = \{C_1, ..., C_k\}$ so that the similarity of patterns in the same C_i cluster is maximal and patterns belonging to distinct clusters differ as much as possible. Partitions must maintain the following properties [12] :

1) Each cluster must have at least one affected pattern, *i.e.* $C_i \neq \emptyset \ \forall i \in \{1, 2, ..., k\}$.
2) Two distinct clusters must not have patterns in common, *i.e* $C_i \cap C_j = \emptyset \ \forall i \neq j$ and $i, j \in \{1, 2, ..., k\}$.
3) Each pattern must definitely be affected to a cluster, *i.e.* $\bigcup_{i=1}^{k} C_i = \mathcal{P}$.

III. K-means Algorithm

K-means [13] is one of the most popular clustering algorithms. Its aim is to classify patterns into k classes by aggregating them around the centers. The algorithm starts by selecting k initial centers (randomly or using a certain heuristic) of the classes, then it places each pattern in the nearest class, then recalculates the center of each class and so on until patterns no longer change class. The k-means process is described in Algorithm 1.

Algorithm 1 K-means algorithm

```
Input: A set of $n$ patterns $X_1, ..., X_n$ ; The number of classes $k$ ;
1: Select $k$ initial centroids $c_1, ..., c_k$;
2: repeat
3:     affect each pattern to the $i^{th}$ cluster having the closest centroid $c_i$;
4:     for $i$ from $1$ to $k$ do
5:         $c_i \leftarrow$ the mean of the $i^{th}$ cluster elements;
6:     end for
7: until centroids does not change
```

978-1-5386-3269-7/17/$31.00 2017 IEEE

The weakness of k-means is that it converges to a local minimum that depends on the initial criteria.

IV. Artificial Bee Colony Algorithm

ABC algorithm [6] is based on behavior of bees and particularly on food search mechanism. Each solution of the optimisation problem represents a food source. There are three types of bees : employed bees, onlookers and scouts. Each employed bee is affected to one food source and is responsible of exploiting it and sharing information about it with the onlookers. Hense, the number of employed bees is equal to the number of solutions of the population. Each onlooker bee chooses one food source among the population and exploits its neighborhood. Each employed bee whose solution is exhausted becomes a scout whose role is to look for a new food source in the search space. ABC algorithm starts by initializing the population of solutions using Eq. 1 then calculates the fitness using Eq. 2.

$$x_{ij} = x^j_{min} + rand(0,1)(x^j_{max} - x^j_{min}) \tag{1}$$

where x^j_{min} and x^j_{max} are the lower and upper bounds of the j^{th} parameter of the i^{th} solution.

$$fit(x_i) = \begin{cases} \frac{1}{1+f_i(x_i)} & \text{if } f_i(x_i) \geq 0 \\ 1 + abs(f_i(x_i)) & \text{if } f_i(x_i) < 0 \end{cases} \tag{2}$$

where $f_i(x_i)$ is the objective function value of the i^{th} solution x_i.

Then the rest of the algorithm consists of three main phases :

1) Employed phase : Which consists of sending emplyed bees to food sources in order to explore the neighborhood looking for better sources using Eq. 3.

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{3}$$

where ϕ_{ij} is a random number belonging to the interval $[-1, 1]$, j is a randomly selected dimension, x_k is a food source position with $k \in 1, 2, , SN$ chosen randomly such that $k \neq i$.

2) Onlooker phase : Where each onlooker bee performs a fitness-based selection between solutions of the population using Eq. 4 and explores its neighborhood using Eq. 3.

$$P_i = \frac{fit_i}{\sum\limits_{n=1}^{SN} fit_n} \tag{4}$$

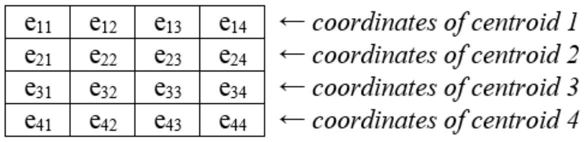
3) Scout phase : Which consists of identifying exhausted solutions and replacing them with new randomly generated solutions using Eq. 1.

Lastly, the best solution is memorized and the phases are repeated until the stop condition is satisfied.

V. Proposed Approach

A. Solution representation

In this study, a solution is a set of centroids and is represented by a $k \times d$ matrix of real numbers, where k is the number of clusters and d the number of features of the patterns to be classified. The i^{th} line of the solution represents the position of the i^{th} centroid. Figure 1 shows an example of a solution with four centroids ($k = 4$) and four features ($d = 4$).



e_{11}	e_{12}	e_{13}	e_{14}	← *coordinates of centroid 1*
e_{21}	e_{22}	e_{23}	e_{24}	← *coordinates of centroid 2*
e_{31}	e_{32}	e_{33}	e_{34}	← *coordinates of centroid 3*
e_{41}	e_{42}	e_{43}	e_{44}	← *coordinates of centroid 4*

Fig. 1: Exemple of a candidate solution

B. Fitness calculation

To evaluate solutions quality, we use Eq. 2. Where the objective function f is the Sum of Squared Errors (SSE), which is defined as follows :

$$SSE(X,C) = \sum_{i=1}^{N} Min \ \{ \ \|X_i, C_l\|^2 \ \mid \ l = 1, ..., k \ \} \tag{5}$$

where C is a solution, X the set of patterns X_i and C_l the l^{th} centroid of the solution C. The distance used is the Euclidean distance which is defined as :

$$d(X_u, X_v) = \sqrt{\sum_{i=1}^{d} (e_{u,i} - e_{v,i})^2} \tag{6}$$

where $e_{u,i}$ and $e_{v,i}$ are the i^{th} elements of X_u and X_v vectors respectively.

In order to improve ABC algorithm, several research studies have emerged. According to previous studies that focused on ABC, the main weakness of ABC algorithm is the lack of exploitation of solutions. This is mainly due to the randomness of the search process of neighbor solutions (Eq. 3). To overcome this problem, many authors have made improvements that affect the neighborhood function. Among the most popular, there is GABC algorithm [7], where the author proposes a new equation to calculate neighborhood for both employed and onlooker phases. The equation uses the information of the best solution *gbest* and is defined as follows :

$$v_{ij} = x_{ij} + \phi_{ij} \times (x_{ij} - x_{kj}) + \psi_{ij} \times (gbest_j - x_{ij}) \tag{7}$$

where x_{ij} is the current solution, $\phi_{ij} \in [-1, 1]$ and $\psi_{ij} \in [0, 1.5]$ are uniformly distributed random numbers. $gbest_j$ is the j^{th} element of the best solution $gbest$ found so far and j as defined in [7], is a random number which represents one of the solution elements indices. In addition, we have found that calculating the neighborhood with the previous formula for all j (for all elements of the solution) yielded better results without significantly increasing algorithm time consumption.

Moreover, according to the author of [14], the lack of exploitation of the ABC algorithm is partly due to the fact that the neighborhood function in the original algorithm is used by both employed and onlooker bees. Thus, no variety search behavior can be anticipated. Two different equations are proposed in [14] to calculate the neighborhood. The employed bees use, instead of Eq. 3, the mutation *current-to-rand/1* of DE algorithm, which is defined as :

$$v_i = x_i(g)+k_i\times(x_{i_1}(g)-x_i(g))+F'\times(x_{i_2}(g)-x_{i_3}(g)) \quad (8)$$

where i_1, i_2 and i_3 are indexes of food sources randomly chosen so that $i_1 \neq i_2 \neq i_3 \neq i$. k_{ij} is a random value from a uniform distribution between 0 and 1. $F' \in [0,1]$ a random value from a uniform distribution created once for each iteration.

As for onlooker bees, the neighborhood equation proposed in [14] is defined as :

$$v_{ij} = gbest_j + F \times (x_{dest,j} - x_{src,j}) \quad (9)$$

where $gbest$ denotes the bee with the currently best fitness value. $x_{dest,j}$ and $x_{src,j}$ are the bees chosen randomly such that $f(x_{dest,j}) < f(x_{src,j})$, f being the objective function to minimize. F here is a randomly chosen value from a uniform distribution between 0 and 1.

Assuming that the research strategies of the two phases (employed and onlooker) had to be different and after several tests, we found that the best combination was to use the *current-to-rand/1* (Eq. 8) in the employed phase and the GABC formula (Eq. 7) for the onlooker phase.

Again with the aim of improving the ABC algorithm, and based on a part of the study of *D. C. Tran* [10], we have added a phase in each iteration which is the mutation phase. For that, we used a formula proposed in [10] and inspired by the mutation of the DE algorithm and the arithmetic crossing of the *HABC* [9] algorithm, it is defined as :

$$u_{ij} = F'_{ij} \times (x_{ij} - x_{k_1 j}) + F_{ij} \times (gbest_j - x_{k_2 j}) \quad (10)$$

where x_{k_1} et x_{k_2} are two food sources which are randomly selected from food source population such that i, k_1 and k_2 are mutually different. F_{ij} and F'_{ij} are in this case randomly chosen between 0 and 1.

One of the advantages of this approach is that it significantly improves the ABC algorithm without increasing the computation time or adding parameters to be adjusted, since the mutation operators used do not depend on the parameters F and F'.

Lastly, since the best solution *gbest* is used to calculate the neighborhood (for both employed and onlooker phases) as well as in the mutation phase, it seemed interesting to initialize *gbest* with a significant value to accelerate the convergence of the algorithm and give it a head start. For that, we chose k-means algorithm to yield the initial value of *gbest* because of its simplicity and low time consumption.

Algorithm 2 K-ABC-DE algorithm

```
1: Initialize the population solution $x_i$ , $i = 1, 2, ..., SN$ ;
2: Initialize $gbest$ using k-means algorithm (Alg. 1);
3: Assign employed bees to solutions;
4: Evaluate the fitness value $fit(x_i)$ of solutions $x_i$
   using Eq. 2;
5: Set $trial_i$ to 0 , $i = 1, 2, ..., SN$;
6: repeat
   /* Employed Phase */
7:     for $i = 1$ to $SN$ do
8:         Produce a new solution $v_i$ from $x_i$ using Eq. 8;
9:         Evaluate the fitness value of the new solution
           using Eq. 2;
10:        Update $trial_i$;
11:    end for
12:    Calculate the probabilities $p_i$ , $i = 1, 2, ..., SN$
       using Eq. 4;
   /* Onlooker Phase */
13:    for each onlooker bee do
14:        Select a solution $x_i$ depending on $p_i$;
15:        Produce a new solution $v_i$ from $x_i$ using Eq. 7;
16:        Evaluate the fitness value of the new solution
           using Eq. 2;
17:        Update $trial_i$;
18:    end for
   /* Mutation Phase */
19:    for chaque solution $x_i$ do
20:        Produce a mutant solution $u_i$ using Eq. 10;
21:        Evaluate the fitness value of the new solution
           using Eq. 2;
22:        Update $trial_i$;
23:    end for
   /* Scout Phase */
24:    for each solution $x_i$ do
25:        if $trial_i > limit$ then
26:            Replace $x_i$ with a new solution generated
               using Eq. 1;
27:        end if
28:    end for
29:    Memorize the best solution found so far;
30: until stop condition is satisfied
31: return $gbest$;
```

VI. Experimental Results and Discussion

The proposed algorithm K-ABC-DE is compared with five other algorithms : ABC, GABC, DE/rand/1/bin (which uses Eq. 11 as mutation operator and binomial crossover), DE/current-to-rand/1 (which uses Eq. 8 for mutation and no crossover operator) and K-means.

$$u_i = x_{i_1} + F \times (x_{i_2} - x_{i_3}) \tag{11}$$

where F is the mutation factor and $x_{i_1}, x_{i_2}, x_{i_3}$ are randomly chosen solutions from the population so that indices are mutually different.

All algorithms are implemented in Python 3.6 using Spyder environnement and executed on a computer running Windows 7 with an intel Core(TM) i7-3537U CPU @ 2.00 GHz, 4.00 GB RAM.

A. Termination condition

The number of iterations is no longer a reasonable measure because a different calculation time can be taken by each iteration according to the algorithm. In order to compare the different algorithms, a fair measurement method should be selected as the number of function evaluations (FE) [9]. It corresponds to the number of fitness evaluations. This is justified because in data clustering, the task that consumes the most time is the calculation of the fitness function (SSE). Since all the algorithms tested stop after the same number of evaluations, the calculation time is close when they are applied to the same data. Hense, we compare performances obtained by the algorithms within a nearly equal amout of time. Concerning k-means, it is an approximate algorithm and can get results in seconds, but its results are of less good quality. In our case, the number of evaluations was set to 10,000 FE.

B. Parameter tuning

For the ABC algorithm, the population size was set to 100 solutions and the limit parameter was set to 50. Similarly for GABC. Concerning DE algorithm, the population size was set to 50 solutions, mutation factor F to 0.6 and crossover rate CR to 0.9. Lastly, the algorithm K-ABC-DE has the same parameters as ABC, since the mutation factors F and F' are chosen randomly.

C. Testing sets

All algorithms were tested on seven Benchmark Datasets available on the UCI Machine Learning Repository and listed in Table I. The used datasets are : Iris, Wisconsin Diagnostic Breast Cancer (WDBC), Glass, Contraceptive Method Choice (CMC), Wine, Balance and Liver Disorder.

Dataset	Patterns	Features	Classes
Iris	150	4	3
WDBC	569	30	2
Glass	214	9	6
CMC	1473	9	3
Wine	178	12	3
Balance	625	4	3
Liver	345	6	2

TABLE I: Benchmark Datasets

D. Comparative analysis

Table II summarizes the results obtained by running each algorithm 30 times for each dataset. The first line indicates the average quality of the 30 runs and the second line refers to the standard deviation.

Concerning the Iris dataset, K-ABC-DE and GABC algorithms have a very low standard deviation, which means that they converge most of the time towards the optimum with a slight advantage for the GABC algorithm. As for the WDBC, Glass, CMC and Wine datasets, the standard deviation obtained by K-ABC-DE is the lowest by far, and the value of the objective function is the best. For the Balance and Liver Disorder datasets, the K-ABC-DE, GABC and DE/rand/1 algorithms have roughly equivalent results, with a slight advantage for DE/rand/1. The other algorithms generally obtained poorer results. We also note that the interest of the proposed algorithm compared to the other approaches is even greater when the dimensions of the data (the number of features) increases. In addition, it can be seen that the improved versions of the ABC algorithm give better results than the basic ABC for the seven datasets. Indeed, the basic version of ABC gives unsatisfactory results, especially in comparison with GABC and K-ABC-DE. It also obtained a very high standard deviation, which reflects its weak capacities to converge. This explains why the basic version of ABC (which uses Eq. 3 as neighborhood function) is no longer widely used in the literature. It is often replaced by a variant like GABC, which, on the other hand, gives quite good results. As for the DE algorithm, the classical version *DE/rand/1/bin* outperforms the *DE/current-to-rand/1* variant for almost all datasets. Moreover, we note that k-means gives correct results but only on two datasets Glass and CMC, but nevertheless obtained the lowest score in terms of standard deviation with the dataset Glass. For the Iris, WDBC, Wine, and Liver Disorder datasets, k-means obtained the poorest results in terms of quality.

The figure 2 shows the evolution of the fitness (calculated using Eq. 2) over the iterations for 5 algorithms and 7 datasets. The abscissae of the curves represent the number of evaluations (FE) performed and the ordinates the average fitness of the population. The points thus represent the quality of the population as a function of the progress of the algorithm. The figure confirms the slowness of convergence of ABC, in comparison with the other algorithms. Also, it can be noted that the K-ABC-DE algorithm converges rapidly with all the datasets, and in particular with Glass and CMC, which are the algorithms for which k-means has obtained good results. This shows the interest of k-means in the K-ABC-DE algorithm. Finally, we note that GABC shows a fast convergence on several datasets. Indeed, most of the time, it begins to converge faster than the other algorithms but is subsequently caught up and overtaken by K-ABC-DE in terms of convergence and quality.

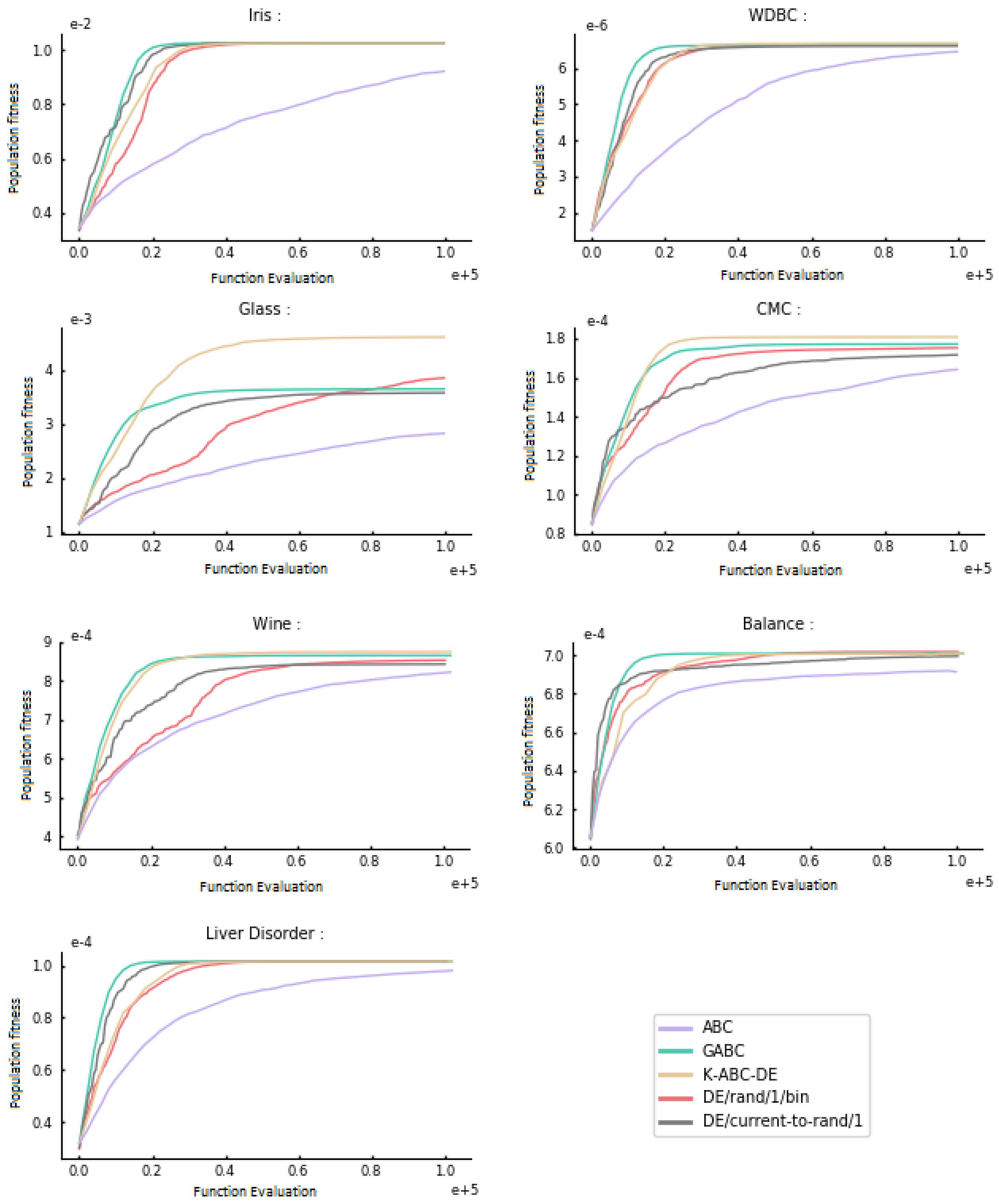


Fig. 2: Convergence of 5 algorithms on 7 datasets

Dataset		K-ABC-DE	GABC	ABC	DE/rand/1/bin	DE/cur-to-rand/1	K-mens
Iris	Quality	96.6554843	96.6554825	98.8456326	96.6556557	97.0462708	103.264402
	Std	3.63241e-06	2.05303e-13	1.46079202	0.00039876	0.72913165	10.7584489
WDBC	Quality	149477.002	150239.326	151658.634	150196.251	151796.930	152647.252
	Std	2.76911506	397.061289	1384.34748	438.338841	1539.55515	2.9104e-11
Glass	Quality	215.756369	269.707471	337.571711	264.789762	274.698917	229.036086
	Std	6.30752800	14.5898115	15.0787818	12.1778873	13.8982214	15.1034412
CMC	Quality	5532.49555	5585.07576	5861.92821	5624.77208	5735.05912	5543.86329
	Std	0.14786244	26.0300713	56.9836642	42.3278499	91.1632828	1.50245126
Wine	Quality	1143.01713	1156.10288	1185.37651	1167.75777	1166.31729	1187.59855
	Std	0.39760860	5.72840844	8.96908007	14.4582320	7.63520494	35.7043236
Balance	Quality	1424.57604	1424.90707	1432.22104	1424.02239	1425.93158	1428.14986
	Std	1.48461709	1.77468802	2.2007602	0.43557487	1.31711510	4.22416141
Liver	Quality	9851.79512	9851.78110	9933.71536	9851.75136	9947.59546	10213.8083
	Std	0.14241793	0.13341057	69.4916237	0.09627148	136.269379	4.71117750

TABLE II: Comparison results of 6 algorithms on 7 datasets

Dataset		K-ABC-DE	ABC-DE
Iris	Quality	96.6554843	96.6554864
	Std	3.63241e-06	6.3071e-06
WDBC	Quality	149477.002	149670.833
	Std	2.76911506	139.280061
Glass	Quality	215.756369	264.851334
	Std	6.30752800	8.49276951
CMC	Quality	5532.49555	5578.57091
	Std	0.14786244	21.8572901
Wine	Quality	1143.01713	1150.62828
	Std	0.3976086	3.39760860
Balance	Quality	1424.57604	1424.68154
	Std	1.48461709	0.89276324
Liver	Quality	9851.79512	9851.97062
	Std	0.14241793	0.94149589

TABLE III: Comparison between K-ABC-DE and ABC-DE

It is estimated from the results that the approach proposed in this study has improved the results achieved by ABC and obtained satisfactory results in comparison with the other algorithms tested on the seven datasets.

E. Evaluation of the impact of k-means

In Table III, K-ABC-DE is compared to ABC-DE in order to measure the impact of k-means initialization of *gbest*.

It is observed from Table III that with the use of k-means, the standard deviation has decreased for most datasets and the quality of the partitions has augmented especially when the dimensions of the data increase. Thus, K-ABC-DE benefits not only from the advantages of ABC and DE but also from those of k-means. Indeed, if we take as an example the WDBC dataset, we note from Table II that k-means gives poor results but with a very low standard deviation, unlike ABC-DE which gives fairly good solutions but much more dispersed. K-ABC-DE gives much better results and much lower standard deviation than ABC-DE for the WDBC dataset and also for the rest of the datasets, except for Balance and Liver Disorder, where the results are almost equivalent. We can conclude that k-means contributes to considerably improve the convergence of the algorithm towards the optimal solution. Furthermore, we note that ABC-DE surpasses the GABC algorithm for most datasets, both by quality and standard deviation.

Figure 3 represents the evolution of the fitness of the population over the iterations for the ABC-DE and K-ABC-DE versions. We note that for all datasets, K-ABC-DE converges significantly faster than the ABC-DE version. This confirms that k-means has a considerable impact on the convergence of the algorithm.

VII. Conclusion

In this study, we proposed the K-ABC-DE approach inspired by several previous research studies. The aim was to combine the advantages of ABC, DE and k-means algorithms while minimizing the disadvantages of each. The main disadvantage of ABC is its poor exploitation of solutions. We have seeked out, through the proposed approach, to find a better balance between exploitation and exploration. Hybrization was car-

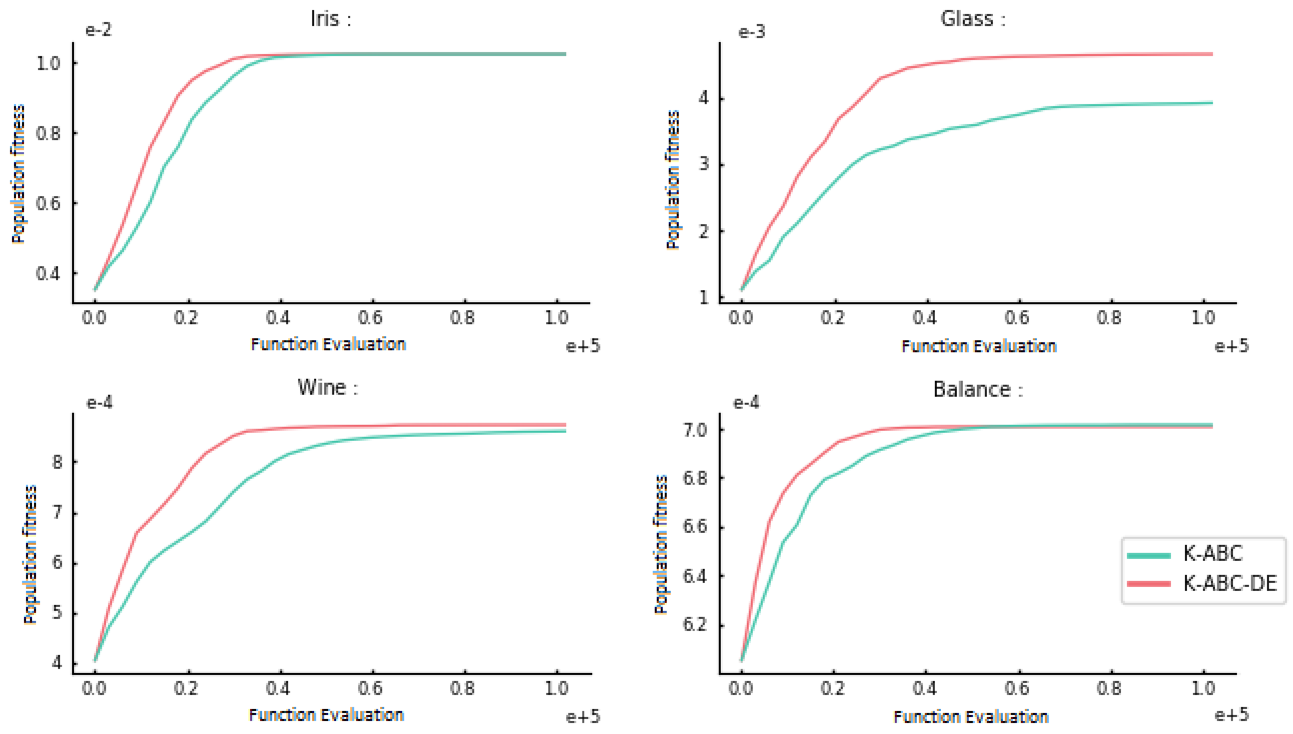


Fig. 3: Convergence of K-ABC-DE and ABC-DE

ried out on several phases of the algorithm ABC, namely: initialization, employed bee phase, onlooker phase and the addition of a mutation phase inspired by DE. The results show not only that the quality of the partitions is higher, but also that the convergence is faster compared with the basic ABC algorithm and other state-of-the-art algorithms. Lastly, one of the advantages of the K-ABC-DE approach is that it significantly improves the ABC algorithm without increasing the computation time or adding parameters to be adjusted, since the DE mutation operators used do not depend on the parameters F and F'. For future works, attention will be given to high dimensional data, as well as adaptation to other data mining problems such as feature selection.

References

[1] Bezdek, J. C., Boggavarapu, S., Hall, L. O., & Bensaid, A. (1994, June). *Genetic algorithm guided clustering*. In Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on (pp. 34-39). IEEE.

[2] Van der Merwe, D. W., & Engelbrecht, A. P. (2003, December). *Data clustering using particle swarm optimization*. In Evolutionary Computation, 2003. CEC'03. The 2003 Congress on (Vol. 1, pp. 215-220). IEEE.

[3] Shelokar, P. S., Jayaraman, V. K., & Kulkarni, B. D. (2004). *An ant colony approach for clustering*. Analytica Chimica Acta, 509(2), 187-195.

[4] Paterlini, S., & Krink, T. (2004, June). *High performance clustering with differential evolution. In Evolutionary Computation*, 2004. CEC2004. Congress on (Vol. 2). IEEE.

[5] Zhang, C., Ouyang, D., & Ning, J. (2010). *An artificial bee colony approach for clustering*. Expert Systems with Applications, 37(7), 4761-4767.

[6] Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization* (Vol. 200). Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

[7] Zhu, G., & Kwong, S. (2010). *Gbest-guided artificial bee colony algorithm for numerical function optimization*. Applied mathematics and computation, 217(7), 3166-3173.

[8] Zou, W., Zhu, Y., Chen, H., & Sui, X. (2010). *A clustering approach using cooperative artificial bee colony algorithm*. Discrete dynamics in nature and society, 2010.

[9] Yan, X., Zhu, Y., Zou, W., & Wang, L. (2012). *A new approach for data clustering using hybrid artificial bee colony algorithm*. Neurocomputing, 97, 241-250.

[10] Tran, D. C., Wu, Z., Wang, Z., & Deng, C. (2015). *A Novel Hybrid Data Clustering Algorithm Based on Artificial Bee Colony Algorithm and K-Means*. Chinese Journal of Electronics, 24(4), 694-701.

[11] Armano, G., & Farmani, M. R. (2014). *Clustering analysis with combination of artificial bee colony algorithm and k-means technique*. International Journal of Computer Theory and Engineering, 6(2), 141.

[12] Das, S., Abraham, A., & Konar, A. (2008). Automatic clustering using an improved differential evolution algorithm. IEEE Transactions on systems, man, and cybernetics-Part A, 38(1), 218-237.

[13] Hartigan, J. A., & Hartigan, J. A. (1975). *Clustering algorithms* (Vol. 209). New York: Wiley.

[14] Worasucheep, C. (2015). *A Hybrid Artificial Bee Colony with Differential Evolution*. International Journal of Machine Learning and Computing, 5(3), 179.