algorithms

MDPI

Article

K-Means Clustering Algorithm Based on Chaotic Adaptive Artificial Bee Colony

Qibing Jin, Nan Lin * and Yuming Zhang

School of Information, Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China; jinqb@mail.buct.edu.cn (Q.J.); 2015400132@mail.buct.edu.cn (Y.Z.)
* Correspondence: nanlin@mail.buct.edu.cn; Tel.: +86-1312-114-1302

Abstract: K-Means Clustering is a popular technique in data analysis and data mining. To remedy the defects of relying on the initialization and converging towards the local minimum in the K-Means Clustering (KMC) algorithm, a chaotic adaptive artificial bee colony algorithm (CAABC) clustering algorithm is presented to optimally partition objects into K clusters in this study. This algorithm adopts the max–min distance product method for initialization. In addition, a new fitness function is adapted to the KMC algorithm. This paper also reports that the iteration abides by the adaptive search strategy, and Fuch chaotic disturbance is added to avoid converging on local optimum. The step length decreases linearly during the iteration. In order to overcome the shortcomings of the classic ABC algorithm, the simulated annealing criterion is introduced to the CAABC. Finally, the confluent algorithm is compared with other stochastic heuristic algorithms on the 20 standard test functions and 11 datasets. The results demonstrate that improvements in CAABA-K-means have an advantage on speed and accuracy of convergence over some conventional algorithms for solving clustering problems.

Keywords: artificial bee colony (ABC) algorithm; K-means clustering (KMC) algorithm; chaos algorithm; Metropolis algorithm; simulated annealing

Citation: Jin, Q.; Lin, N.; Zhang, Y. K-Means Clustering Algorithm Based on Chaotic Adaptive Artificial Bee Colony. *Algorithms* **2021**, *14*, 53. https://doi.org/10.3390/a14020053

Received: 12 January 2021
Accepted: 2 February 2021
Published: 7 February 2021

1. Introduction

Clustering procedure [1–3] is a process that divides a set of objects into clusters according to the predefined criteria such that objects in the same cluster are more parallel to each other than other objects in different clusters. Clustering is often used in solving a part of complicated tasks in pattern recognition [4], image analysis [5], and other fields on data processing [6]. An excellent clustering algorithm still has a higher interference-free capability and lower time complexity than traditional algorithm when processing large amounts of data. The clustering algorithms can be subdivided into two categories: hierarchical clustering and partitional clustering. The hierarchical clustering algorithm divides the pattern into fewer structures continuously, and it is usually described by the tree structure. Partition clustering is the division of a set of objects into K non-intersecting subsets with high internal similarity. The center-based clustering algorithms are the most popular partitional clustering methods.

K-means is simple and efficient, in which case it becomes one of the most popular center-based cluster methods [3]. However, relying on the initialization of K states and convergence towards the local minimum are significant shortcomings of K-means classification. In order to overcome these problems, many other methodologies have been applied to algorithm. A clustering algorithm which based on genetic algorithm was proposed by Mualik and Bandyopadhyay, and its effectiveness is proved on real-life datasets. Simulated Annealing (SA) approach is proposed to solve the clustering puzzle by Selim and Al-Sultan (1991). Beyond that, many heuristic algorithms like Particle Swarm Optimization (PSO) [7], Differential Evolution (DE), and ABC, have also been successively adapted in the clustering algorithm optimization improvements.

ABC is a swarm intelligence algorithm that is derived from the honeybee colony's gathering behavior. After Turkish scholars Karabogac Denis raised this problem in 2005 [8,9], it was widely applied to solve function optimization problems, for the advantages of less parameters, simpler structure, and being easier to implement [10]. Compared with PSO and Genetic Algorithm (GA), ABC's advantage is proved further in Section 4. Similar to other swarm intelligence algorithms, the performance of optimized ABC mainly depends on its search strategy. Due to the randomness of the search mechanism, the algorithm is easy to get stuck at local optimal value and has slow convergence speed. ABC has been optimized gradually after the proposal, and it has been extended to various fields. Inspired by PS0 algorithm, the global optimal solution is used to guide the search formula (GABC) in [11]. Inspired by the DE algorithm, another new improved algorithm MABC was proposed. In [10], comparing with the original ABC algorithm, the MABC modifies the employed bee stage and onlooker bee stage, which improves the efficiency. The proposed algorithm is then applied to solve a loudspeaker design problem using FEM. The CGABC based on the crossover is proposed in [12]. The Crossover operator of genetic algorithm is introduced into the Global optimized Artificial Bee Colony algorithm. Crossover is the transfer of good genes from the parent of a population to its offspring. The brand new ABC_elite is proposed in [13]. To better balance the tradeoff between exploration and exploitation, it proposes a depth-first search (DFS) framework. The article introduces two novel solution search equations which incorporates the information of elite solutions and can be applied to the employed bee phase. Furthermore, many studies increase search efficiency by changing the greedy search mechanism. Sharma T K [14] changes the search path of scout bee. Two new mechanisms for the movements of scout bees are proposed. In the first method, the scout bee follows a non-linear interpolated path while in the second one, scout bee follows Gaussian movement. Yang, Weifeng Gao [15,16] improves the greedy search and adapts to more optimization problems using introduced adaptive methods. In addition, to enhance the global convergence, when producing the initial population and scout bees, both chaotic systems and opposition-based learning method are employed. Xiang W L [8] proposes a depse-first search framework. Gao W [17] increases information sharing among individuals through improvement. In addition, many scholars [18,19] choose to combine the ABC algorithm with other familiar algorithms for optimization. Each bee should select whether adopts greedy strategy or not based on its fitness value on each generation. A great progress in solving complex numerical optimization problems has achieved in [19,20]. With the continuous improvement and optimization, ABC has been applied in more fields, such as workshop scheduling [21–25] software aging prediction [26], machine learning [27], multi-objective optimization [28], dynamic optimization [29,30] and so on.

ABC has unique advantages for data optimization problems. In this work, the improved ABC is extended to clustering procedures. Crossover operation and adaptive threshold are integrated to improved ABC algorithm. The simulated annealing technique and Fuch chaotic perturbation operation are drawn into the algorithm. Furthermore, the initialization equation and the fitness function are reformed according to the shortcomings of the K-means. The CAABA-K-means has been proved to be superior on speed and accuracy of convergence over some conventional algorithms for solving clustering problems.

The remainder of this work is distributed as follows: Section 2 discusses the ABC algorithm and clustering analysis problems. The chaotic adaptive ABC (CABC) algorithm adapted for solving K-means clustering problems is introduced in Section 3. Section 4 shows that our method outperforms some other methods by showing experimental studies. Section 5 is the conclusion, which summarizes our proposed method.

2. Relative Work

2.1. Artificial Bee Colony Algorithm

ABC is a swarm intelligence algorithm, which imitates the division of labor and the search mode of bees to find the maximum amount of nectar [8]. In the classic ABC

algorithm, the artificial bee colony is divided into three categories according to their behaviors: employed bee, onlooker bee, and scout bee. In the beginning, the number of employed bees equals to onlooker bees, and the third kind of bees appear gradually. The employed bee uses information about the initial honey source to find new honey source and shares the information with the onlooker bee. The onlooker bee waits in the hive and chooses a better source according to the greedy selection mechanism. However, if the honey source information has not been updated for a long time, the corresponding employed bee will be transformed into a scout bee. The task of scout bee is to search for the honey source randomly around the hive and find a new valuable honey source eventually. Self-organization, self-adaption, social division, and collaboration are significant features of the entire colony. ABC simulates the foraging behavior as the process of searching for the optimal solution, defines the adaptability to the environment of individuals as the objective function of the problem to be solved, and takes the greedy selection method as the basis for eliminating the different solution. This process is iterated until the optimal solution is reached and the entire function gradually converges. The steps can be described specifically as follows.

2.1.1. Initialization Stage

In ABC algorithm, the fitness value represents the quality of nectar sources and candidates solution corresponding to food sources. It is assumed that the population size is N. The initial solution is obtained through Equation (1).

$$x_{ij} = x_j^{\min} + rand * (x_j^{\max} - x_j^{\min}) \tag{1}$$

where x_{ij} is the j-th component of the i-th vector. $i = 1, 2, \ldots N$, $j = 1, 2, \ldots D$. x_j^{max} and x_j^{min} are the upper bound and lower bound of the j-th component, and $rand \in [0, 1]$ is a random number from 0 to 1. The algorithm executes global searching randomly for food sources and derives the revenue value.

2.1.2. Employed Bee Stage

After initialization, the ABC algorithm starts the stage of employed bees. Employed bees search randomly around the current region according to Equation (2) and shares the information with the onlookers; thus, a new set of honey sources, $V_i = (V_{i1}, V_{i2}, \ldots V_{iD})$, is generated

$$V_{ij} = x_{ij} + \phi_{ij}\Big(x_{ij} - x_{kj}\Big) \tag{2}$$

where $\phi_{ij} \in [-1, 1]$, $i = 1, 2, \ldots N$. In addition, $j = 1, 2, \ldots D$ is the index, which is chosen randomly. In addition, $i \neq k$ is a necessary requirement to reduce duplication of effort. If the fitness value of a new source V_{ij} is ameliorated, the source will be superseded by the new one.

2.1.3. The Probability of Selecting the New Food Source

The fitness value is calculated by Equation (3), which is an evaluation criteria of nectar source quality.

$$fit_i = \begin{cases} \frac{1}{1+f_i}, f_i \geq 0; \\ 1+|f_i|, \ f_i < 0 \end{cases} \tag{3}$$

where $i = 1, 2, \ldots N$. fit_i denotes the fitness value of x_i. The larger fit_i value means the higher quality of honey source. In addition, f_i is the value of the i -th nectar source's objective function.

When the fitness values are calculated, they are applied to calculate the probability of selecting the i-th honey source P_i, which can be used as a basis for onlooker bees to select honey sources.

$$P_i = \frac{fit_i}{\sum_{n=1}^{N} fit_i} \tag{4}$$

where fit_i denotes the fitness value of the x_i.

2.1.4. Scout Bee Stage

If the search steps reach a certain threshold, but no better position is found, the position of this employed bee is re-initialized randomly according to Equation (1).

Each food source is verified by employed bees and/or onlooker bees for its potential inclusion as a candidate position. The ABC algorithm utilizes employed bees, onlooker bees and scout bees to iteratively search the solution space until reach the maximum number of iterations. If a food source is not improved further through limit trials, it is deemed as an exhausted source and abandoned. Under different conditions, the three kinds of bees transform into each other and the result gradually approaches the optimal solution. The transformation diagram is as Figure 1.

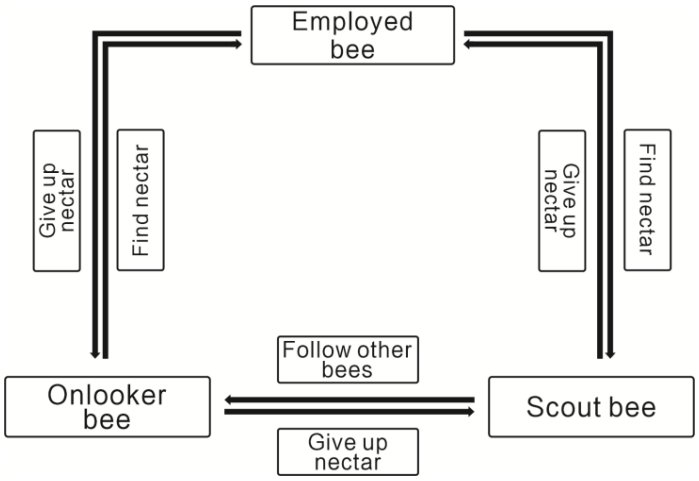


Figure 1. Structure Diagram of Artificial Bee colony Algorithm.

2.2. K-MEANS Cluster Algorithm

Clustering methods are suitable for finding internally homogeneous groups in data. The K-means algorithm is one of the oldest clustering techniques [1], which is constructed based on the iterative hill-climbing process. The main idea is to gather the original data into K clusters according to similar attributes. The main processing procedure is as follows. Firstly, K samples are randomly selected from the original data, and each sample is taken as the center of K clusters. Then the distance between the K center samples and the remained samples is separately calculated. According to the calculation, each sample is classified into the nearest cluster. The iterative process is repeated until the cluster no longer changes. Therefore, the traditional K-means clustering is expressed as follows.

$x = \left\{ x_i \in R^d, i = 1, 2, \cdots, n \right\}$ is the original data sample, where $x_i (i = 1, 2, \cdots, n)$ is d -dimensional vectors. $C = \{C_1, C_2, \cdots, C_k\}$ is the cluster sets, where k is the number of clusters.

K-means criterion function is expressed as follows:

$$J = \sum_{j=1}^{k} \sum_{x_i \in C_j} d(x_i, C_j) \tag{5}$$

where $d(x_i, C_j)$ represents the distance between data x_i and its clustering center C_j, and J represents sum of its internal distances.

3. Chaotic Adaptive Artificial Bee Colony (CAABC) for Clustering

During the iterative optimization process of the classic ABC algorithm, both employed bee and onlooker bee follow a completely random search strategy. As a result, the ABC

algorithm has strong global search capabilities. However, the algorithm selects the honey source blindly at the stage of the employed bee. Only the random number ϕ_{ij}, between –1 and 1], can be used to control the search region of the neighborhoods, and the search direction of onlooker bee is guided by the information of honey source from employed bee. In the entire iteration process of the algorithm, the blindness and randomness make the algorithm more complex and sacrifice the accuracy of algorithm results. For balancing the exploration and exploitation performance, ABC has been optimized from different perspectives. To improve the convergence speed, GABC combines the ABC with PSO algorithm and the global optimal solution is referred to change the random neighborhood searching of classic ABC, which enhances the exploitation performance of algorithm. However, simply adding the global optimal solution not only enhances the convergence speed but also increases the risk of premature convergence. Therefore, inspired by GABC, the information carried by ordinary individuals in this paper is effectively utilized and the search space is adaptively adjusted to avoid premature convergence. In order to overcome the disadvantages of K-means clustering algorithm, such as over dependence on the initial clustering center, easily falling into local optimum, and the premature and slow convergence of the artificial bee colony algorithm due to the limitations of search strategies, a hybrid clustering method combining the improved artificial bee colony algorithm and K-means algorithm is proposed which makes full use of the characteristics of the improved artificial bee colony algorithm and K-means algorithm.

3.1. MAX–MIN Distance Product Algorithm

Initialization affects the global convergence and the performance of the algorithm, so it is particularly important in the evolutionary algorithm. K-means clustering algorithm has high sensitivity to the initial stage. Based on [31,32], we propose a max–min distance product algorithm for initialization. The initialization process not only reduces the randomness of colony initialization but also reduces the sensitivity of Kmeans clustering to initial points. In [33], the max–min distance means is used to search the optimal initial clustering center, in which case convergence speed and accuracy of the algorithm have been significantly improved. However, it may lead to clustering conflict when the initial clustering center is excessive dense. The maximum distance method proposed in [34] reduces the number of iterations effectively, but there would be the problem of initial point deviation. It is possible that the product of two distances is the same, but the density of the points is quite different.

To improve the efficiency, we propose a max–min distance product algorithm. We can get T_m from T according to Equation (6). T_m represents the product of the maximum and minimum values in od.

$$T_m = mix(od) * min(od) \tag{6}$$

where k points are randomly selected as the initial cluster centers from the original data set. Meanwhile, is a data set used to store the distance between other points in the data set among cluster centers. T is an array that stores the product of elements in od. T_m is the product of the maximum center distance and the minimum center distance. The points that corresponding to the T_m are selected as cluster centers instead of initial points. The distribution of current initial points can be dispersed by the max–min distance product. Moreover, it can avoid the situation that two distance products are equal, but the point density of their regions is quite varied, and magnify the difference between points. The enhanced selection method is better.

3.2. New Fitness Function

Fitness function is the crux of population evolution, which determines the solution quality directly. It is the key factor which affects the stability and convergence of the

algorithm. Based on the characteristics of the iteration of ABC and the basic idea of the K-means algorithm, a new fitness function is adopted in this work:

$$fitness_i = CN_i / J_i; i = 1, 2, \cdots N \tag{7}$$

where CN_i represents the number of samples in the i -th cluster.

$$J_i = \sum_{x_j \in C_j} d(x_j, C_i) \tag{8}$$

J_i denotes the sum of the distance between the sample points among the centers. The new fitness value can give rise to equilibrate the influence of the numbers and distance of samples. The phenomenon of inaccurate judgment caused by the same value of CN_i or J_i is avoided, which improves the adaptability of the function and makes the iterative process more accurate.

3.3. New Position Update Rules

3.3.1. Arregate-Dispeise Algorithm

To improve convergence speed and accuracy, aggregate-disperse algorithm is introduced here. On the basis of the simplex method, we propose an "aggregate-disperse operation" as a guiding strategy for the iteration. According to the relationship among the global optimal solution, elite solution and ordinary individual, the search range and step length change adaptively.

- The Simplex Method

The simplex method is a traditional optimization method which uses the iterative transformation of the vertex of the geometric graph to approach the optimal value gradually. Take a dual function as example. Take three points X_1, X_2, X_3, which are not collinear, as the vertices to form a triangle. Calculate the function value $f(X_1), f(X_2), f(X_3)$ and compare them to each other. Calculate the function value $f(X_1), f(X_2), f(X_3)$ and compare them to each other.

(1) $f(X_1) > f(X_2) > f(X_3)$ means X_1 is the worst solution, and X_3 is the best one. The algorithm should search in the opposite direction to find the minimum. $X_4 = (X_2 + X_3)/2$ is the midpoint of X_2X_3, X_5 is on the extension line of X_1X_4, and X_5 is called the reflection point of X_1 with respect to X_4:

$$X_5 = X_4 + \alpha(X_4 - X_1) = -\alpha X_1 + (1 + \alpha)X_4 \tag{9}$$

where α is the reflection coefficient, which equals 1 as usual. The geometric relationship is shown in Figure 2:

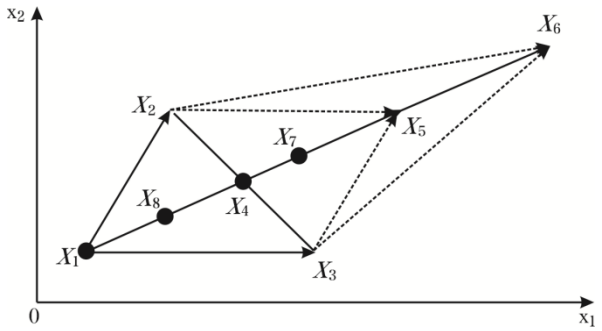


Figure 2. The Simplex Method.

(2) $f(X_5) \leq f(X_3)$ denotes that that direction of searching is correct, the algorithm should keep going in this direction. Let $\alpha = 1.5$. If $f(X_6) \leq f(X_5)$, X_5 is replaced by X_6 to form a new simplex, or X_6 is dropped.

(3) $f(X_3) \leq f(X_5) \leq f(X_2)$ means that the searching is going in the right direction, but doesn't need to expand.
(4) $f(X_2) \leq f(X_5) \leq f(X_1)$ demonstrates that X_5 has gone too far to need to be retracted.
(5) If $f(X_5) > f(X_1)$, X_1, X_2 need to be retracted toward X_3.

- Aggregate and Disperse Operator

The purpose of the aggregate operation is to provide guidance for the population gathering in a potential direction during the iteration. With this operation, the algorithm can increase the convergence speed in the initial stage and strengthen the local search ability in the later stage. x_{global}, x_{ebest}, X_k are three given solutions, the worst individual is moved toward the others. In order to accelerate the convergence speed, the elitist solution x_{ebest} and the global optimal solution x_{global} are used to guide the search process.

The parameter setting is $\alpha \in (0, 2)$ [31] to ensure that the new solution maintains convergence while moving toward a better direction generally. α can be considered as the punishment parameter for the poor individual, and -α is the encouragement parameter.

$\alpha \in (0, 1)$ denotes that the better solution is not found, so the influence of the original strategy should be moderately weakened.

$\alpha \in (1, 2)$ denotes that the original strategy should be strengthen.

The parameters are transformed to form a convex combination to avoid negative weight in the later stage.

$$X_5 = (1 - \beta)X_1 + \beta(\varphi X_2 + (1 - \varphi)X_3) \tag{10}$$

The simplex method and ABC are fused in the two-dimensional coordinate space in Figure 3:

$$X_i = (1 - \beta)X_k + \beta\left(\varphi X_{elite} + (1 - \varphi)X_{global}\right) \tag{11}$$

where $\beta \in (0, 2)$, $\varphi \in (0, 1)$. The vectors OA, OB, and OC represent the global optimal solution, elite solution, and ordinary individual respectively. If $OE = \beta OA + (1 - \beta)OB$, according to convex combination analysis, the point E lies on the line segment AB, and OF ends up in the triangle $\Delta A'B'C$. When the searching area is extended to n-dimensional space, the point F will fall into an area with the line segment AB as the median line, which is the potential space limited by elite solutions and globally optimal solutions. Finally, multiple planes intersect at the global optimum. The results converge to the globally optimal solution with high probability.

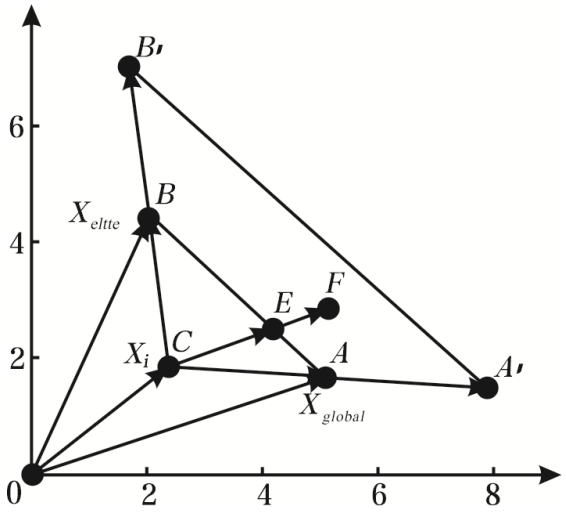


Figure 3. Aggregate Operation.

However, if the three solutions are collinear, it will be trapped in the local optimum because the search space is too narrow. Therefore, the disperse operation is used to expand the search space.

$$X_i = (1 - 2 \times \varsigma) \times X_c + \varsigma \times (\gamma \times X_a + (1 - \gamma) \times X_b) \tag{12}$$

The vectors ζ, γ are random numbers between 0 and 1. After the dispersion operation, the search area is extended to triangle area $\Delta ABC'$. In multidimensional space, multiple planes intersect at the global optimum with high probability eventually. Figure 4. demonstrates the disperse operation in the two-dimensional plane.

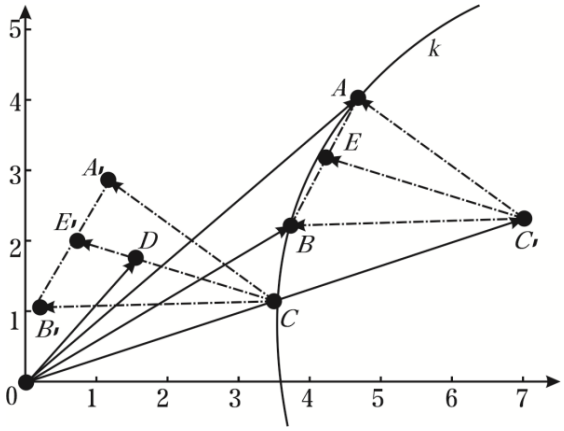


Figure 4. Disperse Operation.

3.3.2. Adaptive Adjustment

In the iterative process of ABC algorithm, the neighborhood search range is controlled by a random parameter, and the neighborhood search is performed randomly and aimlessly. The effectiveness of the algorithm is influenced by the blindness and randomness visibly. In order to remedy the defects, an adaptive parameter is used to adjust the algorithm's search step length. Furthermore, in order to have stronger adaptive performance, we replace the fixed-size parameter with an alterable one, $s(iter)$, during the iteration.

$$s(iter) = -2(exp(-q\hat{}1.9)) + 2; \tag{13}$$

where $q = iter/max\ cycle$. $iter$ and $max\ cycle$ are the number of current iteration and the maximum iteration severally. As is shown in Equation (13), the step length factor $s(iter)$ decreases and adjusts adaptively with the iteration process. In the initial time, the global searching is executed efficiently with a large step length, and the step length is variable in the later process to achieve a detailed local search.

3.3.3. Genetic Crossover

The randomness of the searching method limits the optimization ability and affect the convergent rate of canonical ABC. To balance the performance of the algorithm, the crossover operation is carried out to intersect with the global optimal solution based on unbiased adaptive optimization. The main goal of the GA algorithm is for reference, and the diversity of the population and overall optimization ability are further increased by crossing with the excellent parent generation. Crossover operations are performed to find more valuable individuals in the searchable space. The larger the size of the intersection, the more combinations of the allogeneic genes are exchanged, and the wider the searchable range is. However, with the expanding of the size of the intersection, the increase of searchable scope shrinking. The larger the scope of the crossover operation means the smaller the probability that any individual in the space can be searched. Therefore, the probability of excellent vertices being searched will affected by the scope of intersection.

CR is the local search coefficient, that is used to control the activity of individuals during the local search. The smaller the value means the more active of individual's behavior.

The improved algorithm in this paper has as good local search ability because of the ergodicity of chaotic search. Combined with the characteristics of gene crossover and the ergodicity of the chaotic disturbance, we conduct a comparative test for different CR values from 0 to 1, and finally concludes that the algorithm can achieve better performance when CR = 0.6.

Combining the improvements above, we can get a new position updating formula, the calculation process is shown as Equation (14).

$$x_{i,j} = \begin{cases} (1-s(iter)) * x_{k,j} + s(iter) * \varphi * x_{ebest} \\ \quad + (1-\varphi) * x_{global}), rand < cr \\ x_{global} + s(iter) * (x_{global} - x_{k,j}), others \end{cases} \tag{14}$$

If the location of three solutions is on the same line, the position updating criterion is changed to Equation (15) on the base of disperse operation:

$$x_{i,j} = \begin{cases} (1-s(iter)) * x_{k,j} + \frac{s(iter)}{2} * \varphi * x_{ebest} \\ \quad + (1-\varphi) * x_{global}), rand < cr \\ x_{global} + \frac{s(iter)}{2} * \left(x_{global} - x_{k,j}\right), others \end{cases} \tag{15}$$

In the iteration, the cross factor $cr = 0.6$, x_{global} represents the global optimal solution, $x_{k,j}$ is the ordinary individual selected from $\{1, 2 \ldots N\}$ randomly, and x_{ebest} is the elitist solution. After sorting the solution, x_{ebest} is selected from the top $R * N$ solutions randomly, where N is the population number, and $R = 0.1$.

In order to verify the effectiveness of the improved method in this part, the improved ABC algorithm whose position updating according to the "aggregate-disperse operation" and cross operation is temporarily named CAABC-2. As the components of CAABC, its effectiveness will be proved in the fourth part.

3.4. New Chaotic Disturbance

Chaos is a unique movement pattern of a nonlinear system with particular features of sensitivity to the initial value, randomness, and ergodicity. Chaotic search is generated by iterating chaos sequence through a certain particular format and extending the numerical range of the chaos variables to the value range of the optimization variables through the form of the carrier wave. Fuch chaos [32], as a new type of discrete mapping, has unique advantages over logistic chaotic mapping, with more optimized chaotic performance and fewer iterations. It is proved that the chaotic map has no rational number fixed point, then the mapping relational formula is used to establish a chaotic model that is used to solve the Lyapunov exponent, and the sensitivity of chaotic maps to initial values is investigated under large variation and small variation on initial starting points. The chaotic map is then used to establish chaotic generator to replace the finite-collapse map, and to improve the dynamic performance of chaotic optimization. The method improves the search efficiency by continuously reducing the searching space of variables and enhancing search precision. It is more ergodic and does not fall into local optimum with incorrect initial value setting. The expression is:

$$x_{n+1} = cos(1/x_n^2) \tag{16}$$

Thus, iteration sequence X_{n+1} is obtained. In the formula, $n = \{1, 2, 3 \ldots N\}$;

The Lyapunov exponent of Fuch chaos is solved in [32], and the results shows that Fuch chaos has a stronger chaotic property and a more homogeneous ergodic property than Logistic chaos and Tent chaos.

In this work, the adaptive value of the function is computed based on a novel function and the chaotic disturbance is increased to 15% of individuals with poor performance and the elite solution to update the historical optimal adaptive value. If the new solution is

superior, the new position will be used to replace the original one. The chaotic algorithm can effectively avoid converging on local optimum and gain higher precision.

To verify the effectiveness of the Fuch chaos in the CAABC, on the basis of CAABC-2, chaotic disturbance is added. It is named CAABC-1 temporarily.

3.5. New Probability of Selecting Based on SA

Simulated Annealing (SA) algorithm is a heuristic Monte Carlo inversion method [33]. The temperature attenuation function is used to control the temperature declining process for simulated annealing of solid-state systems. In this work, the Metropolis algorithm is integrated into ABC. When the adaptive value of the new honeybee source is lower than the current one, it might be accepted with a certain probability. The annealing temperature T determines the probability.

For the simulated annealing nonlinear inversion, the cooling function is:

$$T(t+1) = \sigma T(t) \tag{17}$$

where $T(t)$ is the temperature of t times, T_0 is initial temperature value, and σ is the coefficient of cooling, generally between 0.9 and 1. In this work, the single variable experiments were carried out within the standardized threshold for several times, and the algorithm achieved the best performance when the value of σ was determined to be 0.95 eventually. The difference between the new fitness value F' and the current fitness value F is:

$$\Delta F = F' - F \tag{18}$$

If $\Delta F < 0$, the new food source is selected, or the selection is conducted according to the Metropolis algorithm.

$$exp\left(\frac{-\Delta F}{T}\right) \geq rand(0,1) \tag{19}$$

The inferior solution with poor performance is accepted possibly according to the metropolis rule, therefore, the points are easier to escape from the local optimum, and the prematurity of ABC algorithm has been largely curbed.

3.6. The Procedures of CAABC-K-means

The novel clustering algorithm is integrated with chaotic adaptive artificial bee colony (CAABC) and K-means cluster (KMC) algorithm. The new location obtained by CAABC is used as the initial point of KMC for iteration process, and then the new center point obtained after calculation is applied to update the swarm. In order to match up to KMC, the max–min distance product algorithm and a novel fitness function are proposed based upon ABC algorithm. In the search space, the step length is reduced adaptively when the search approaches the optimum solution. Moreover, the cross-operation increases population diversity in the position updating process. Furthermore, the ergodicity of Fuch chaotic perturbation is carried out on the elite solution and infeasible solutions, meanwhile, the inferior solution is accepted with a certain probability according to the metropolis rule. Hence, the points are easier to jump out of the local optimum, and the prematurity of ABC algorithm has been largely curbed. The employed bee is translated into a scout when the food source of which has been exhausted. If a scout discovered a valuable food source, it would be employed.

In this way, CAABC algorithm and K-mean clustering are alternately performed until the end of the algorithm. The flow chart of algorithm execution is shown in Figure 5. The main steps of the algorithm can be described as follows:

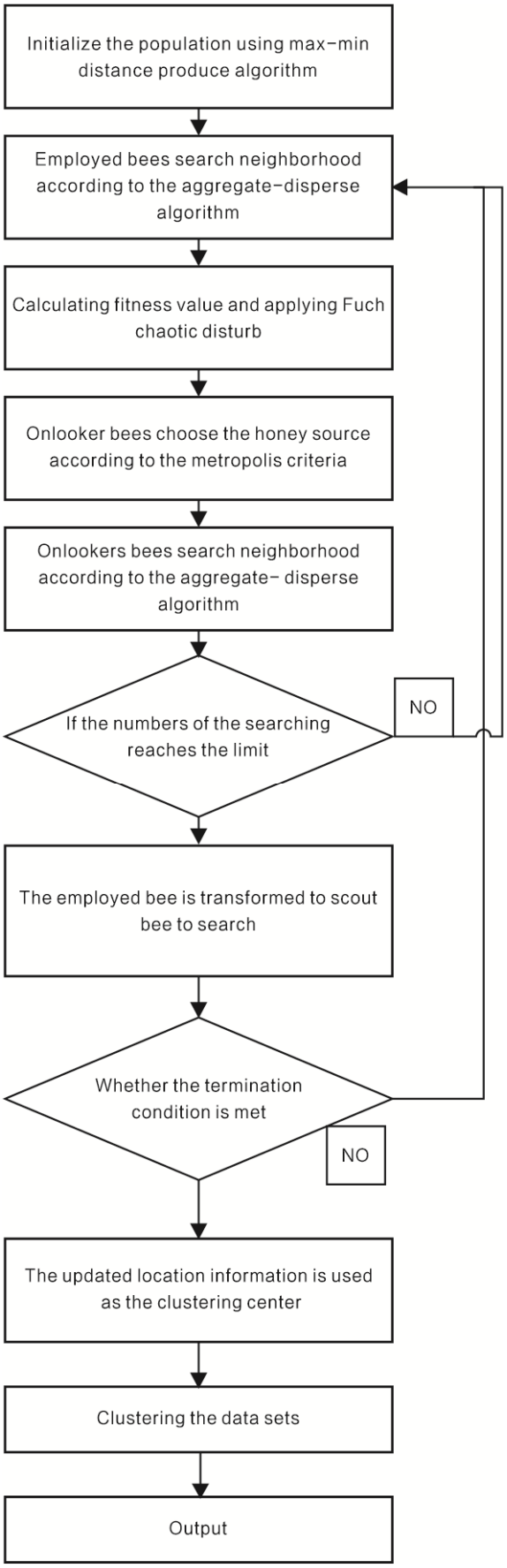


Figure 5. Flow chart of Chaotic Adaptive Artificial Bee Colony-Kmeans.

1. Initial parameters are set as follows: N represents the number of population, D denotes the space vector dimension, *max cycle* is the maximum iteration times, and cross parameter $cr = 0.6$. *limit* is the threshold of maximum optimization times, and the annealing coefficient $\sigma = 0.95$. The initial population is obtained according to the max–min distance product algorithm.
2. The fitness value can be obtained according to Equation (3), and then solution approaches to the global optimal solution. At the same time, chaotic perturbations are added into the elite solution, which is selected from the preponderant solution set randomly and the infeasible solution in the bottom 15% according to Equation (16). The position is updated according to Equation (14) or Equation (15). Eventually, the location of the honey source is extended to the D-dimensional space. Whether the new solution is accepted depends on the Metropolis criteria.
3. Onlooker bee executes the employed bee option and neighborhood searching performs under the same criteria.
4. The updated location information, which is obtained after all the onlooker bees have completed the search, is used as the clustering center, the data set is performed a K-means iterative clustering, and the clustering center of each class is refreshed with the clustering division.
5. If *limit* for abandonment is reached, the employed bee determines whether the number of updates reaches the limit. If the limit is reached, the employed bee is translated into a scout when the food source of which has been exhausted. A new round of honey source searching begins.
6. If the number of iterations has reached the maximum "*max cycle*", the optimal solution is output, otherwise, the algorithm goes back to step 2.
7. K-means algorithm is executed to get results.

4. Numerical Experiments

In order to verify the effectiveness of CAABC, we design an optimization performance test on 20 benchmark functions. In order to make fair comparison, the parameter settings are referred as [35]. The algorithm is compared with the classic ABC, the Hybrid Artificial Bee Colony which proposed memory mechanisms (HABC) [34], the Improved Artificial Bee Colony which charges permutation as employed to represent the solutions (IABC) [36] and the DFSABC algorithm respectively. In order to verify the effectiveness of each component of the algorithm, CAABC-1 and CAABC-2 are also compared. The details of benchmark functions are listed in Table 1. In addition, we also use three standard evaluation index to evaluate the clustering performance of CAABC-K-means algorithm and other algorithms in this part.

Table 1. Benchmark Function.

Number	Equation	Name	Domain
1	$f_{01}(x) = \sum_{i=1}^{D} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	Alpine	$[-10, 10]^D [-10, 10]^D$
2	$f_{02}(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	Schwefel2.22	$[-10, 10]^D$
3	$f_{03}(x) = \max\{\lvert x_i \rvert, 1 \le x_i \le D\}$	Schwefel2.21	$[-10, 10]^D$
4	$f_{04}(x) = \sum_{i=1}^{D} i{x_i}^4 + random[0, 1)$	QuarticWN	$[-1.28, 1.28]^D$
5	$f_{05}(x) = \sum_{i=1}^{D} i{x_i}^4$	Quartic	$[-1.28, 1.28]^D$
6	$f_{06}(x) = \sum_{i=1}^{D} \lvert i x_i \rvert^{(i+1)}$	SumPower	$[-10, 10]^D$

Table 1. *Cont.*

Number	Equation	Name	Domain
7	$f_{07}(x) = \sum_{i=1}^{D} z_i{}^2 z = x-$	ShiftedSphere	$[-100, 100]^D$
8	$f_{08}(x) = \sum_{i=1}^{D} (x_i + 0.5)^2$	Step	$[-100, 100]^D$
9	$f_9 = \sum_{i=1}^{D} x_i{}^2 + \left(\sum_{i=1}^{D} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{D} 0.5 i x_i\right)^4$	Zakharow	$[-5, 5]^D$
10	$f_{10}(x) = \sum_{i=1}^{D} i x_i{}^2$	SumQuares	$[-10, 10]^D$
11	$f_{11}(x) = \sum_{i=1}^{D} \lvert x_i \rvert^{(i+1)}$	SumDifference	$[-10, 10]^D$
12	$f_{12}(x) = -418.98288727243369 \times D + \sum_{i=1}^{D} \left[-x_i \sin(x_i) \sqrt{\lvert x_i \rvert}\right]$	Schwefel2.26	$[-500, 500]^D$
13	$f_{13}(x) = \sum_{i=1}^{D} [100(z_{i+1} - x_i{}^2)^2 + (z_i - 1)^2]$, $z = x-$	ShiftedRosenbrock	$[-10, 10]^D$
14	$f_{14}(x) = \sum_{i=1}^{D} \left(\sum_{J+1}^{i} x_j\right)^2$	Schwfel1.2	$[-100, 100]^D$
15	$f_{15} = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)$	Ackley	$[-32, 32]^D$
16	$f_{16}(x) = \sum_{i=1}^{D} (x_i{}^2 - 10cos(2\pi x) + 10)$	Griewank	$[-600, 600]^D$
17	$f_{17}(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i{}^2 - \prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}}) + 1$	Rastrigin	$[-5.12, 5.12]^D$
18	$f_{18}(x) = 0.5 + \frac{sin^2\sqrt{\sum_{i=1}^{D} x_i{}^2} - 0.5}{\left(1 + 0.001\sum_{i=1}^{D} x_i{}^2\right)^2}$	Schaffer	$[-100, 100]^D$
19	$f_{19}(x) = \sum_{i=1}^{D} [100(x_{i+1} - x_i{}^2)^2 + (x_i - 1)^2]$	Rosenbrock	$[-10, 10]^D$
20	$f_{20}(x) = \sum_{i=1}^{D} x_i{}^2$	Sphere	$[-100, 100]^D$

The simulation experiment is coded using MATLAB® R2019a, running on a system with 2.5 GHz Core-i5 CPU, 4 GB RAM, and Windows 10 operating system.

4.1. Test Environment and Parameter Settings

The experimental parameters are set as follows: dimensions D are 30 and 60 respectively, and the maximum number of iterations *max cycle* is set to 15e4 and 30e4 respectively. In addition, the population size N is set to 20 and the limit is set to $D * N/2$. Under different dimension conditions, we run each benchmark function for 20 times independently.

4.2. CAABC Performance Analysis

To demonstrate the superiority and effectiveness of the CAABC, the CAABC algorithm is compared with other well-known algorithms on twenty benchmark problems. The population parameter settings are same as the setting mentioned in [36]: $N = 25$, the maximum number of evaluations max$cycle = 10,000 * D$, and other function parameter settings are shown in Table 2. Tables 3 and 4 demonstrate the comparison under the 30-dimensional and 60-dimensional parameter settings respectively. The best results are shown in bold. All algorithms are executed in the same machine environment. Each

result was recorded after separate trials for 20 times. The results are listed in Tables 3 and 4; it can be clearly seen that most results of CAABC are remarkable in accuracy of convergence. Other algorithms are run with longer CPU time, which proves the superiority of CAABC algorithm. We select five representative test function for comparison, namely, Sphere (unimodal separable function US), Rosenbrock (unimodal nonseparable function UN), Rastrigin, Alpine (multimodal separable function MS) and Ackley (multimodal nonseparable function MN). The result is given in Figures 6 and 7, which can make it visualized clearly from different views. In addition, the abscissa of the Figure 6 and 7 represents the number of iterations of the algorithm, the ordinate represents the value of the optimization function.

Table 2. Parameter Setting.

Algorithm	Parameter
DFSABC	$N = 20, max\ cycle = 10,000 * D, limit = N * D$
IABC	$N = 30, max\ cycle = 10,000 * D, limit = 100, MI = 10, r = 3$
CAABC	$N = 20, max\ cycle = 10,000 * D, limit = N * D/2cr = 0.6$
ABC	$N = 20, max\ cycle = 10,000 * D, limit = N * D$
HABC	M = 3, $N = 20, maxcycle = 10,000 * D, limit = N * D\ l_{max}$
PSO+K-means	$N = 20, limit = N * D, max\ cycle = 10,000 * D, C = 5, \omega min_{max}$

Table 3. Comparison with other improved Artificial Bee Colony in 30 dimensions.

No.	Mean/Std.	ABC	IABC	HABC	CAABC	DFSABCelite	CAABC1	CAABC2
f_1	Mean	1.37×10^{-16}	2.10×10^{-16}	1.15×10^{-15}	$\mathbf{3.17 \times 10^{-29}}$	8.91×10^{-25}	4.55×10^{-25}	6.90×10^{-16}
	Std.	1.14×10^{-16}	5.39×10^{-16}	3.04×10^{-15}	$\mathbf{2.39 \times 10^{-145}}$	6.24×10^{-25}	5.19×10^{-105}	1.82×10^{-15}
	CPUtime	25.23	8.45	6.25	**4.03**	6.49	5.36	7.02
f_2	Mean	8.94×10^{-186}	1.04×10^{-30}	3.76×10^{-183}	$\mathbf{3.73 \times 10^{-195}}$	7.98×10^{-193}	3.73×10^{-195}	2.26×10^{-183}
	Std.	0.00	0.00	0.00	**0.00**	0.00	0.00	0.00
	CPUtime	16.44	5.26	7.68	**3.60**	4.02	5.92	6.74
f_3	Mean	2.30×10^{-180}	1.99×10^{-9}	3.04×10^{-179}	$\mathbf{1.45 \times 10^{-179}}$	8.94×10^{-175}	6.25×10^{-177}	5.37×10^{-175}
	Std.	**0.00**	4.84×10^{-3}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	CPUtime	15.50	6.64	7.65	**4.67**	5.63	7.45	8.34
f_4	Mean	7.52×10^{-4}	1.95×10^{-4}	2.42×10^{-4}	$\mathbf{2.95 \times 10^{-4}}$	6.92×10^{-4}	2.92×10^{-4}	5.61×10^{-4}
	Std.	4.28×10^{-6}	4.33×10^{-6}	4.80×10^{-6}	$\mathbf{2.30 \times 10^{-6}}$	3.07×10^{-6}	3.33×10^{-6}	4.72×10^{-6}
	CPUtime	52.42	25.21	24.43	**11.53**	16.43	15.92	42.53
f_5	Mean	4.76×10^{-228}	6.01×10^{-4}	9.95×10^{-217}	$\mathbf{1.49 \times 10^{-237}}$	1.95×10^{-230}	1.82×10^{-230}	5.97×10^{-217}
	Std.	0.00	0.00	6.67×10^{127}	**0.00**	3.01×10^{-197}	1.01×10^{-197}	4.00×10^{127}
	CPUtime	40.05	30.21	25.34	**18.77**	23.43	19.52	20.32
f_6	Mean	1.86×10^{-189}	7.00×10^{-32}	4.43×10^{-198}	$\mathbf{4.83 \times 10^{-218}}$	1.41×10^{-199}	4.00×10^{-200}	2.74×10^{-198}
	Std.	**0.00**	8.85×10^{-32}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	CPUtime	59.32	31.46	29.56	**10.42**	28.45	25.64	35.28
f_7	Mean	3.47×10^{-118}	3.03×10^{-6}	3.06×10^{-120}	$\mathbf{2.67 \times 10^{-124}}$	8.68×10^{-123}	1.02×10^{-124}	1.84×10^{-120}
	Std.	7.77×10^{-218}	6.21×10^{-7}	9.18×10^{-120}	$\mathbf{3.53 \times 10^{-124}}$	8.33×10^{-122}	1.03×10^{-122}	5.56×10^{-120}
	CPUtime	17.04	10.26	12.46	**7.53**	12.64	12.02	13.31
f_8	Mean	4.72×10^{-121}	8.83×10^{-6}	1.12×10^{-123}	$\mathbf{2.04 \times 10^{-124}}$	1.06×10^{-123}	2.04×10^{-124}	1.31×10^{-123}
	Std.	1.05×10^{-124}	2.19×10^{-7}	2.39×10^{-123}	$\mathbf{2.93 \times 10^{-124}}$	2.06×10^{-123}	2.83×10^{-123}	2.67×10^{-123}
	CPUtime	5.66	3.61	4.73	**2.63**	3.76	3.32	4.47
f_9	Mean	$\mathbf{3.81 \times 10^{-69}}$	5.35×10^{-26}	$\mathbf{3.73 \times 10^{-69}}$	3.87×10^{-69}	3.20×10^{-68}	3.97×10^{-69}	2.14×10^{-68}
	Std.	$\mathbf{9.36 \times 10^{-67}}$	1.54×10^{-27}	$\mathbf{9.69 \times 10^{-68}}$	9.74×10^{-69}	9.74×10^{-69}	9.74×10^{-67}	1.17×10^{-68}
	CPUtime	47.32	35.22	33.52	**23.38**	26.71	28.41	32.25

Table 3. *Cont.*

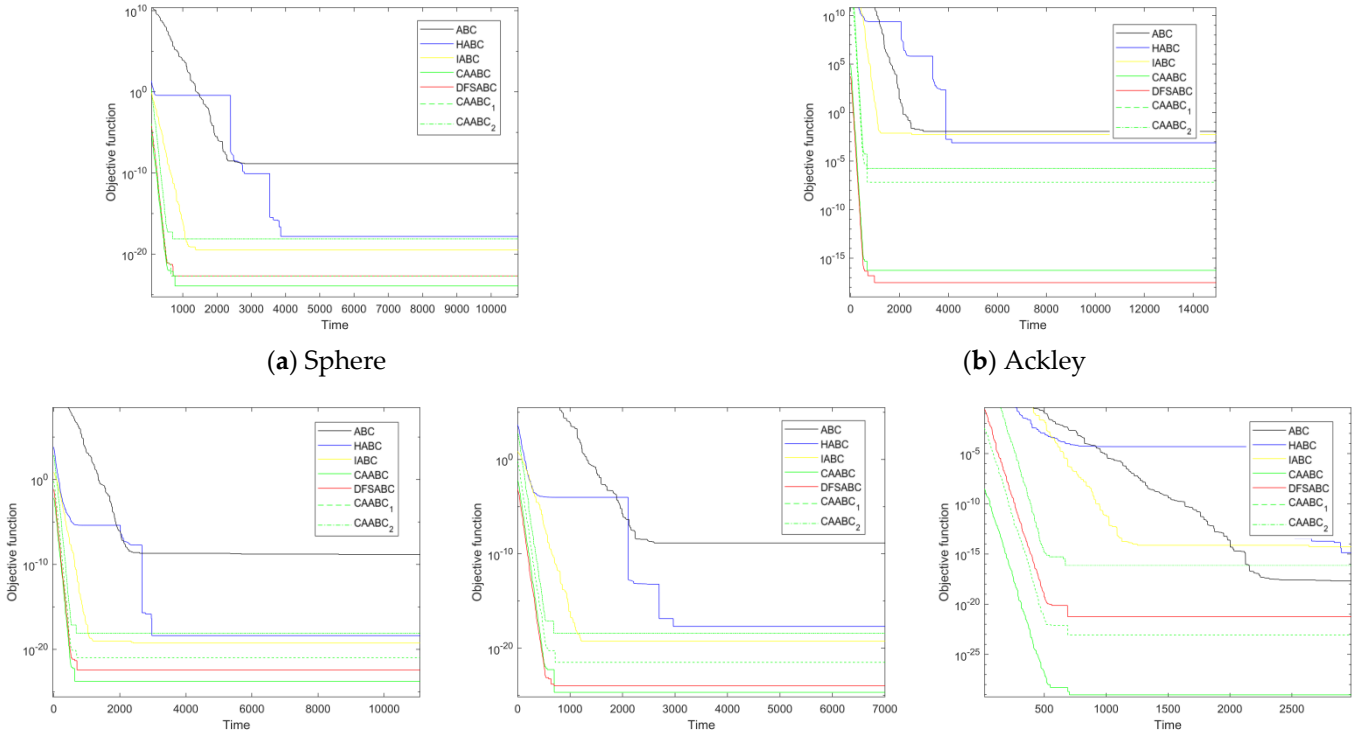
No.	Mean/Std.	ABC	IABC	HABC	CAABC	DFSABCelite	CAABC1	CAABC2
f_{10}	Mean	2.21×10^{-5}	2.73×10^{-5}	3.36×10^{-258}	$\mathbf{1.09 \times 10^{-237}}$	3.39×10^{-236}	1.22×10^{-238}	2.22×10^{-235}
	Std.	1.32×10^{-6}	1.90×10^{-6}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	CPUtime	34.75	15.75	14.75	**12.82**	16.43	15.76	17.34
f_{11}	Mean	5.56×10^{-184}	7.83×10^{-32}	1.19×10^{-190}	$\mathbf{1.00 \times 10^{-192}}$	1.39×10^{-192}	1.99×10^{-190}	9.06×10^{-181}
	Std.	**0.00**	2.73×10^{-33}	**0.00**	2.97×10^{-161}	6.97×10^{-159}	3.57×10^{-159}	6.64×10^{-144}
	CPUtime	24.69	15.67	14.39	**11.71**	14.04	13.52	15.43
f_{12}	Mean	4.32×10^{-5}	3.05×10^{-5}	**0.00**	**0.00**	**0.00**	0.00	0.00
	Std.	2.98×10^{-6}	3.63×10^{-7}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	CPUtime	23.53	13.24	12.88	**7.54**	12.67	13.85	27.48
f_{13}	Mean	1.94×10^{-238}	2.61×10^{-8}	8.74×10^{-286}	$\mathbf{1.65 \times 10^{-294}}$	8.24×10^{-286}	1.44×10^{-288}	5.74×10^{-285}
	Std.	**0.00**	2.21×10^{-9}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	CPUtime	25.65	19.56	17.55	**10.59**	17.66	18.95	19.05
f_{14}	Mean	1.36×10^{-56}	1.46	5.95×10^{-102}	$\mathbf{3.50 \times 10^{-106}}$	5.95×10^{-103}	4.90×10^{-105}	3.93×10^{-102}
	Std.	3.00×10^{-56}	1.29×10^{-2}	4.75×10^{-102}	6.99×10^{-106}	$\mathbf{6.75 \times 10^{-106}}$	7.99×10^{-106}	2.85×10^{-102}
	CPUtime	18.64	15.45	15.04	**6.24**	8.75	9.77	13.94
f_{15}	Mean	8.86×10^{-17}	4.49×10^{-16}	2.21×10^{-17}	$\mathbf{1.54 \times 10^{-20}}$	9.81×10^{-19}	7.91×10^{-20}	1.39×10^{-17}
	Std.	1.15×10^{-16}	5.02×10^{-17}	3.12×10^{-17}	$\mathbf{1.76 \times 10^{-20}}$	3.45×10^{-19}	1.40×10^{-19}	1.89×10^{-17}
	CPUtime	9.86	9.07	8.52	**6.87**	8.05	7.92	8.09
f_{16}	Mean	3.09×10^{-16}	2.83×10^{-17}	3.44×10^{-17}	$\mathbf{3.30 \times 10^{-20}}$	3.54×10^{-19}	3.58×10^{-19}	2.09×10^{-17}
	Std.	2.03×10^{-17}	3.53×10^{-17}	3.64×10^{-17}	$\mathbf{4.16 \times 10^{-20}}$	6.67×10^{-18}	6.63×10^{-19}	2.59×10^{-17}
	CPUtime	20.34	14.35	17.45	**12.86**	15.99	17.63	18.63
f_{17}	Mean	1.81×10^{-19}	3.32×10^{-17}	7.86×10^{-17}	$\mathbf{2.32 \times 10^{-20}}$	$\mathbf{2.09 \times 10^{-20}}$	3.00×10^{-18}	4.72×10^{-17}
	Std.	2.59×10^{-17}	2.53×10^{-17}	3.34×10^{-17}	$\mathbf{3.85 \times 10^{-20}}$	3.97×10^{-18}	3.97×10^{-18}	2.24×10^{-17}
	CPUtime	12.96	10.32	8.44	**7.56**	8.94	8.07	9.30
f_{18}	Mean	3.39×10^{-12}	1.47×10^{-242}	2.21×10^{-242}	$\mathbf{6.81 \times 10^{-251}}$	4.38×10^{-247}	4.38×10^{-247}	1.33×10^{-242}
	Std.	2.22×10^{-13}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	CPUtime	22.37	20.73	19.55	**18.83**	20.08	20.44	21.56
f_{19}	Mean	1.36×10^{-15}	4.21×10^{-17}	2.09×10^{-17}	$\mathbf{7.68 \times 10^{-21}}$	6.59×10^{-19}	8.51×10^{-20}	1.29×10^{-17}
	Std.	8.02×10^{-16}	6.04×10^{-17}	3.42×10^{-17}	$\mathbf{2.62 \times 10^{-25}}$	3.75×10^{-19}	2.25×10^{-19}	2.08×10^{-17}
	CPUtime	16.32	13.44	14.35	**13.01**	15.33	15.53	16.22
f_{20}	Mean	1.36×10^{-15}	6.25×10^{-17}	3.09×10^{-17}	$\mathbf{7.68 \times 10^{-21}}$	2.53×10^{-18}	1.51×10^{-18}	2.01×10^{-17}
	Std.	8.02×10^{-16}	8.84×10^{-17}	3.02×1017	$\mathbf{7.70 \times 10^{-21}}$	3.94×10^{-19}	3.75×10^{-19}	1.84×10^{-17}
	CPUtime	4.78	4.14	4.02	**3.63**	3.98	4.02	4.64

Table 4. Comparison with other improved Artificial Bee Colony in 60-dimensions.

No.	Mean/Std.	ABC	IABC	HABC	CAABC	DFSABC_elite	CAABC1	CAABC2
f_1	Mean	1.52×10^{-14}	4.93×10^{-17}	4.53×10^{-19}	$\mathbf{2.78 \times 10^{-22}}$	1.04×10^{-19}	5.36×10^{-20}	3.61×10^{-19}
	Std.	2.99×10^{-14}	2.06×10^{-17}	6.22×10^{-19}	$\mathbf{2.16 \times 10^{-22}}$	6.17×10^{-19}	3.17×10^{-19}	8.03×10^{-19}
	CPUtime(s)	34.36	11.55	12.25	**8.09**	9.99	7.56	11.03
f_2	Mean	3.92×10^{-210}	6.88×10^{-61}	7.14×10^{-210}	$\mathbf{9.17 \times 10^{-253}}$	5.45×10^{-240}	2.80×10^{-240}	4.63×10^{-210}
	Std.	**0.00**	1.70×10^{-61}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	CPUtime	25.40	11.23	13.60	**5.69**	7.32	8.92	11.04
f_3	Mean	4.54×10^{-179}	4.45×10^{-2}	5.96×10^{-178}	$\mathbf{2.26 \times 10^{-181}}$	8.32×10^{-180}	4.40×10^{-180}	3.92×10^{-178}
	Std.	**0.00**	5.29×10^{-3}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	CPUtime	27.50	13.64	15.63	**8.97**	10.43	12.55	15.32
f_4	Mean	2.03×10^{-4}	1.22×10^{-4}	6.04×10^{-4}	$\mathbf{2.02 \times 10^{-4}}$	2.04×10^{-4}	2.09×10^{-4}	2.09×10^{-4}
	Std.	3.36×10^{-7}	9.67×10^{-7}	9.54×10^{-6}	$\mathbf{6.08 \times 10^{-7}}$	2.11×10^{-7}	4.21×10^{-7}	6.32×10^{-6}
	CPUtime	80.47	39.21	37.43	**20.33**	26.42	30.02	34.53

Table 4. *Cont.*

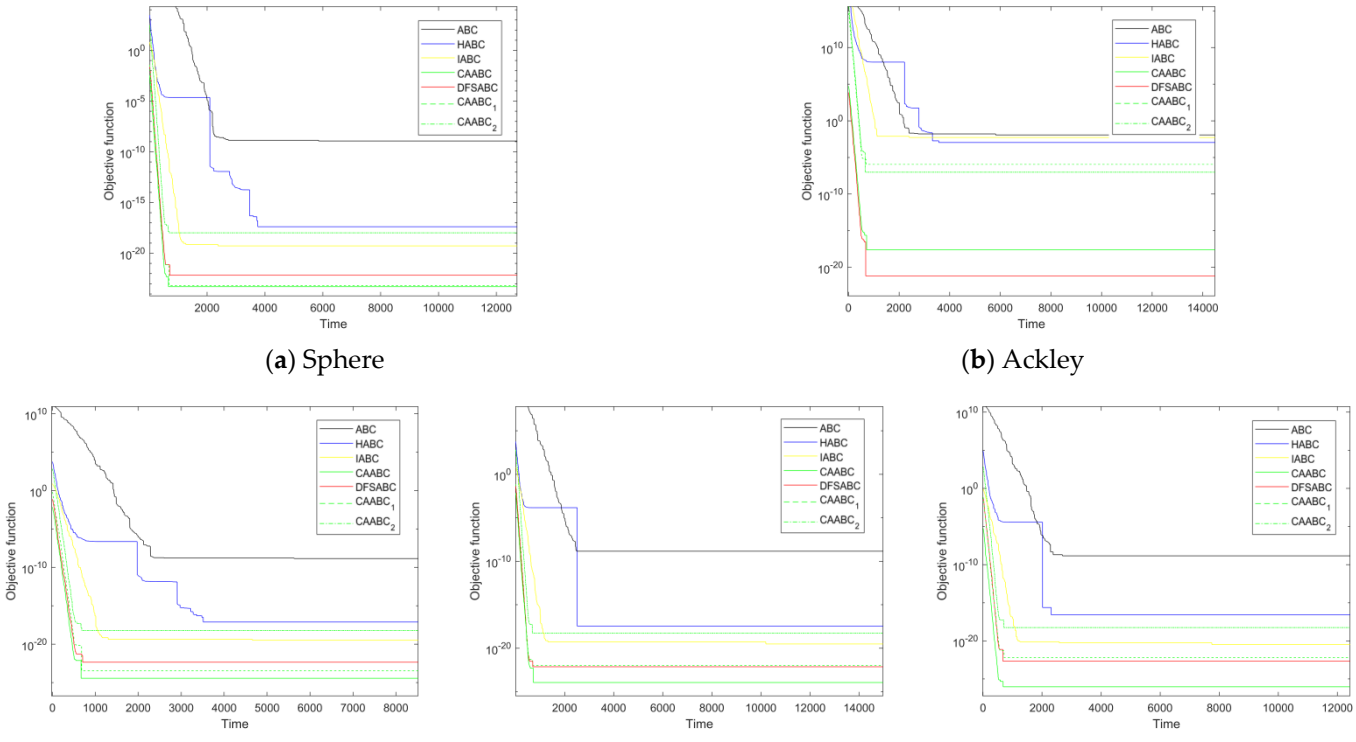
No.	Mean/Std.	ABC	IABC	HABC	CAABC	DFSABC_elite	CAABC1	CAABC2
f_5	Mean	2.52×10^{-229}	2.02×10^{-4}	4.08×10^{-237}	**0.00**	8.15×10^{-242}	4.19×10^{-242}	2.64×10^{-237}
	Std.	**0.00**	3.07×10^{-5}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	CPUtime	79.05	59.34	60.32	**37.08**	40.02	43.22	69.02
f_6	Mean	4.05×10^{-195}	3.05×10^{-62}	6.66×10^{-205}	$\mathbf{1.47 \times 10^{-249}}$	4.56×10^{-224}	2.35×10^{-224}	4.32×10^{-205}
	Std.	**0.00**	7.49×10^{-63}	6.02×10^{-157}	**0.00**	3.32×10^{-189}	1.71×10^{-189}	3.90×10^{-157}
	CPUtime	106.99	43.42	39.53	**18.02**	36.43	37.66	49.58
f_7	Mean	3.70×10^{-121}	4.25×10^{-6}	7.08×10^{-120}	$\mathbf{1.49 \times 10^{-122}}$	1.33×10^{-120}	6.92×10^{-121}	5.45×10^{-120}
	Std.	7.66×10^{-121}	2.20×10^{-8}	4.10×10^{-119}	$\mathbf{3.08 \times 10^{-122}}$	2.10×10^{-119}	1.08×10^{-119}	4.02×10^{-119}
	CPUtime	23.44	15.42	15.33	**10.93**	13.41	13.02	16.33
f_8	Mean	1.08×10^{-120}	3.56×10^{-6}	9.05×10^{-119}	$\mathbf{2.19 \times 10^{-121}}$	5.65×10^{-120}	3.02×10^{-120}	6.23×10^{-119}
	Std.	2.23×10^{-120}	6.24×10^{-7}	2.85×10^{-118}	$\mathbf{4.54 \times 10^{-121}}$	1.27×10^{-120}	8.87×10^{-121}	1.85×10^{-118}
	CPUtime	11.73	7.61	7.52	**3.32**	4.66	5.32	6.45
f_9	Mean	9.34×10^{-254}	1.37×10^{-28}	2.66×10^{-258}	$\mathbf{2.54 \times 10^{-265}}$	9.06×10^{-258}	4.66×10^{-258}	7.59×10^{-258}
	Std.	**0.00**	1.24×10^{-29}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	CPUtime	87.77	55.22	53.88	**27.31**	29.75	30.41	58.22
f_{10}	Mean	6.63×10^{-178}	5.40×10^{-6}	9.12×10^{-123}	$\mathbf{1.29 \times 10^{-123}}$	9.01×10^{-123}	5.30×10^{-123}	1.17×10^{-122}
	Std.	1.65×10^{-128}	1.38×10^{-7}	1.84×10^{-122}	$\mathbf{2.33 \times 10^{-132}}$	9.84×10^{-123}	6.26×10^{-123}	1.83×10^{-122}
	CPUtime	46.64	19.75	24.56	**18.62**	19.04	19.35	20.53
f_{11}	Mean	6.63×10^{-178}	8.67×10^{-62}	1.86×10^{-204}	$\mathbf{4.01 \times 10^{-220}}$	9.86×10^{-214}	5.07×10^{-214}	1.21×10^{-204}
	Std.	1.65×10^{-128}	6.90×10^{-63}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	CPUtime	31.72	23.42	19.34	**18.21**	20.35	20.42	23.22
f_{12}	Mean	3.98×10^{-5}	1.83×10^{-5}	2.04×10^{-5}	$\mathbf{2.07 \times 10^{-5}}$	2.31×10^{-5}	2.25×10^{-5}	2.82×10^{-5}
	Std.	8.19×10^{-6}	3.31×10^{-6}	3.87×10^{-7}	$\mathbf{3.84 \times 10^{-5}}$	5.99×10^{-7}	5.06×10^{-7}	9.00×10^{-8}
	CPUtime	34.13	18.34	17.15	**12.66**	17.68	18.97	20.88
f_{13}	Mean	6.10×10^{-235}	1.32×10^{-8}	6.29×10^{-236}	$\mathbf{1.08 \times 10^{-241}}$	5.96×10^{-238}	3.07×10^{-238}	4.11×10^{-236}
	Std.	0.00	1.12×10^{-9}	0.00	**0.00**	0.00	0.00	0.00
	CPUtime	35.95	30.66	27.64	**20.59**	28.43	27.05	28.43
f_{14}	Mean	8.43×10^{-136}	2.00	4.13×10^{-103}	$\mathbf{1.74 \times 10^{-145}}$	5.13×10^{-131}	2.64×10^{-131}	2.68×10^{-103}
	Std.	8.00×10^{-9}	5.62×10^{-2}	8.54×10^{-103}	$\mathbf{3.61 \times 10^{-145}}$	8.04×10^{-113}	4.14×10^{-113}	5.53×10^{-103}
	CPUtime	28.14	16.85	19.44	**10.32**	12.75	12.42	15.64
f_{15}	Mean	9.96×10^{-18}	2.24×10^{-18}	2.57×10^{-1}	$\mathbf{1.54 \times 10^{-20}}$	$\mathbf{8.96 \times 10^{-21}}$	4.92×10^{-4}	1.67×10^{-1}
	Std.	1.15×10^{-16}	2.24×10^{-2}	5.76×10^{-1}	$\mathbf{1.76 \times 10^{-20}}$	9.96×10^{-9}	5.12×10^{-9}	3.73×10^{-1}
	CPUtime	12.43	10.77	10.42	**7.04**	9.65	7.32	10.04
f_{16}	Mean	3.09×10^{-16}	5.83×10^{-17}	1.68×10^{-17}	$\mathbf{2.17 \times 10^{-22}}$	4.97×10^{-18}	2.56×10^{-18}	1.41×10^{-17}
	Std.	2.03×10^{-17}	3.93×10^{-17}	1.55×10^{-17}	$\mathbf{2.24 \times 10^{-22}}$	1.35×10^{-19}	6.96×10^{-20}	1.01×10^{-17}
	CPUtime	30.03	27.44	26.42	**24.60**	25.56	25.60	29.33
f_{17}	Mean	3.06×10^{-16}	3.33×10^{-17}	6.94×10^{-17}	$\mathbf{3.84 \times 10^{-24}}$	6.44×10^{-20}	3.31×10^{-20}	4.50×10^{-17}
	Std.	2.59×10^{-17}	2.53×10^{17}	1.41×10^{-16}	$\mathbf{4.91 \times 10^{-24}}$	3.43×10^{-22}	1.79×10^{-22}	9.14×10^{-17}
	CPUtime	18.06	17.32	15.84	**15.01**	17.33	16.87	18.05
f_{18}	Mean	1.56×10^{-12}	2.00×10^{-216}	5.33×10^{-242}	$\mathbf{2.09 \times 10^{-248}}$	5.03×10^{-245}	2.59×10^{-245}	3.46×10^{-242}
	Std.	2.05×10^{-13}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	CPUtime	27.56	23.77	23.05	**22.85**	22.98	23.05	24.63
f_{19}	Mean	3.76×10^{-15}	7.03×10^{-17}	6.20×10^{-18}	$\mathbf{6.00 \times 10^{-24}}$	3.42×10^{-19}	1.76×10^{-19}	4.24×10^{-18}
	Std.	6.25×10^{-15}	6.66×10^{-17}	8.07×10^{-18}	$\mathbf{6.91 \times 10^{-24}}$	3.07×10^{-21}	1.58×10^{-21}	5.23×10^{-18}
	CPUtime	21.93	17.46	17.55	**17.02**	17.39	17.53	18.27
f_{20}	Mean	3.82×10^{-16}	3.21×10^{-17}	1.24×10^{-17}	$\mathbf{4.21 \times 10^{-25}}$	3.77×10^{-21}	1.94×10^{-21}	8.04×10^{-18}
	Std.	1.63×10^{-16}	2.36×10^{-17}	5.71×10^{-17}	$\mathbf{4.39 \times 10^{-25}}$	3.01×10^{-19}	1.55×10^{-19}	3.71×10^{-17}
	CPUtime(s)	9.36	4.55	4.23	**3.04**	3.99	3.54	5.03



(**a**) Sphere (**b**) Ackley

(**c**) Rosenbrock (**d**) Rastrigin (**e**) Alphine

Figure 6. Comparison with other improved Artificial Bee Colony algorithms in 30 dimensions.



(**a**) Sphere (**b**) Ackley

(**c**) Rosenbrock (**d**) Rastrigin (**e**) Alphine

Figure 7. Comparison with other improved Artificial Bee Colony algorithms in 60 dimensions.

According to the results in Tables 3 and 4, it shows that CAABC is superior to or at least equal to other algorithms in the rest of benchmark functions except for the several

benchmark functions. The case shown in Table 3 happens only on f_9, f_{11} and f_{14}. In addition, in these functions, the difference between the improved algorithm and the others is less than 5%. However, in 60 dimensional comparison experiment, in Table 4, the improved algorithm achieves good results in f_{15}. At the same time, it can be clearly seen in Figure 5 that CAABC also has a better convergence rate. Based on the above experimental sresults, the superiority of this algorithm is proved.

With the increase of dimension, the results of CAABC are even closer to ideal results. It indicates that the optimization effect of the CAABC is better than canonical ABC and other mentioned algorithms. The best value, the worst value, the average value, or the standard deviation, might be more ideal when the running time is basically the same. By comparing with CAABC-1 and CAABC-2, the effectiveness of the improved algorithm components has also been verified. Overall, compared with classic ABC and other improved ABC algorithms, the accuracy and efficiency of convergence have been enhanced. The exploration and exploitation performance are productively balanced at the same time.

4.3. CAABC-K-means Performance Analysis

The CAABC clustering algorithm on four standard evaluation indices are tested and compared with other well-known algorithms to evaluate the clustering performance of the proposed algorithm. To prove the clustering performance of the improved algorithm, in addition to the comparison algorithm mentioned above, experimental comparison with PA [35] and GPAM [37] clustering algorithm are also added. The general parameter setting is shown in Table 2. In addition, the maximum number of iterations *max cycle* is set to 100. The eleven datasets are Iris (7 January 1988), Balance-scale (22 April 1994), Wine (7 January 1991), E.coli (1 September 1996), Glass (9 January 1987), Abalone (12 January 1995), Musk (9 December 1994), Pendigits (7 January 1998), Skin Seg (17 July 2012), CMC (7 July 1997), and Cancer datasets (3 March 2017)(http://archive.ics.uci.edu/ml/). They have been considered to study and evaluate the performance of algorithms by many authors. The details of Iris, Balance-scale, Wine, E.coli, Glass, Abalone, Musk, Pendigits, Skin Seg, CMC, and Cancer datasets are summarized in Table 5. The optimal result is shown in bold in Tables 6–9.

Table 5. The datasets downloaded from UCI Machine Learning Repository.

Datasets	Samples	Dimensions	Classes
Iris	150	4	3
Balance-scale	625	4	3
Glass	214	10	6
Wine	178	13	3
ECOLI	336	7	8
Abalone	4177	8	28
Musk	6598	166	2
Pendigits	10,992	16	10
Skin Seg.	245,057	3	2
CMC	1473	9	3
Cancer	683	9	2

Table 6. The Normalized Mutual Information for classifying the eleven training datasets.

Datasets	K-Means	ABC+K-Means	PSO+K-Means	CAABC-K-Means	PAM	GPAM
Iris	70.22	71.32	73.32	**79.08**	77.39	79.06
Balance-scale	52.33	53.21	56.33	**59.03**	53.49	57.83
Glass	39.44	40.32	40.36	**49.32**	48.23	49.03
Wine	38.09	40.30	53.20	**57.20**	56.32	57.04
ECOLI	57.00	57.30	57.40	**60.32**	56.03	58.33
Abalone	49.00	47.30	49.50	**68.09**	48.22	57.34
Musk	50.02	53.40	59.32	**59.98**	47.56	54.99
Pendigits	40.05	40.78	49.03	**58.93**	40.00	56.09
Skin Seg.	78.00	79.03	83.01	**88.7**	63.04	80.37
CMC	68.04	79.03	83.07	**89.00**	69.34	74.95
Cancer	53.99	58.34	56.83	**59.99**	57.09	59.03

Table 7. The Accuracy for classifying the eleven training datasets.

Datasets	K-Means	ABC+K-Means	PSO+K-Means	CAABC-K-Means	PAM	GPAM
Iris	50.28	54.32	53.02	**59.06**	58.99	59.00
Balance-scale	50.33	52.29	51.03	**60.01**	54.04	58.32
Glass	60.44	59.32	58.96	**69.32**	68.33	69.08
Wine	89.10	89.19	90.43	**93.11**	92.84	92.94
ECOLI	83.76	84.30	85.29	**89.04**	86.00	87.04
Abalone	70.99	72.02	74.91	**84.11**	84.01	84.05
Musk	60.73	69.93	68.34	**70.00**	68.35	69.68
Pendigits	50.82	50.01	59.11	**63.47**	53.06	62.44
Skin Seg.	70.93	72.38	80.93	**81.02**	60.99	75.64
CMC	59.02	59.24	59.15	**62.03**	58.37	60.75
Cancer	49.03	53.04	53.75	**59.23**	57.98	59.00

Table 8. The average running time (sec.) for classifying the eleven training datasets.

Datasets	K-Means	ABC+K-Means	PSO+K-Means	CAABC-K-Means	PAM	GPAM
Iris	0.35	0.49	0.27	**0.23**	0.33	0.25
Balance-scale	0.78	0.79	0.7	**0.49**	0.79	0.52
Glass	0.98	1.03	0.9	**0.79**	1.03	0.96
Wine	1.06	1.79	0.99	**0.61**	1.11	0.85
ECOLI	0.73	0.95	0.7	**0.57**	0.95	0.60
Abalone	3.90	3.07	2.33	**0.93**	7.99	0.95

Table 8. *Cont.*

Datasets	K-Means	ABC+K-Means	PSO+K-Means	CAABC-K-Means	PAM	GPAM
Musk	2.02	1.68	1.42	**0.61**	10.04	3.02
Pendigits	2.97	2.03	1.93	**0.38**	9.73	1.04
Skin Seg.	3.01	2.93	4.09	**0.46**	6.83	1.97
CMC	1.92	1.31	1.77	**0.58**	2.98	0.82
Cancer	0.34	0.32	0.28	**0.15**	0.44	0.19

Table 9. The f-score of the training datasets.

	K-Means			ABC+K-Means			PSO+K-Means			CAABC-K-Means			PAM			GPAM		
Datasets	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
Iris	0.90	0.88	0.90	0.90	0.90	0.90	0.91	0.92	0.92	**0.99**	**0.97**	**0.98**	0.94	0.96	0.94	0.97	0.97	0.97
Balance-scale	0.93	0.92	0.93	0.91	0.92	0.93	0.94	0.95	0.95	**1.00**	**1.00**	**1.00**	0.98	0.96	0.94	0.98	0.99	0.94
Glass	0.84	0.83	0.85	0.84	0.83	0.80	0.82	0.83	0.8	**0.90**	**0.91**	**0.95**	0.82	0.83	0.8	0.9	0.88	0.82
Wine	0.93	0.94	0.94	0.93	0.93	0.94	0.97	0.98	0.96	**1.00**	**1.00**	**1.00**	0.96	0.96	0.92	0.98	0.96	0.94
ECOLI	0.76	0.77	0.79	0.80	0.84	0.83	0.82	0.84	0.83	**0.89**	**0.89**	**0.89**	0.80	0.84	0.83	0.83	0.85	0.84
Abalone	0.29	0.34	0.32	0.23	0.24	0.22	0.19	0.24	0.22	**0.49**	**0.39**	**0.42**	0.22	0.24	0.22	0.29	0.34	0.32
Musk	0.73	0.72	0.70	0.63	0.60	0.67	0.57	0.52	0.60	**0.83**	**0.82**	**0.80**	0.53	0.50	0.50	0.63	0.72	0.70
Pendigits	0.70	0.72	0.73	0.77	0.72	0.75	0.78	0.77	0.78	**0.83**	**0.83**	**0.83**	0.70	0.70	0.73	0.79	0.79	0.73
Skin Seg.	0.66	0.63	0.64	0.76	0.72	0.74	0.60	0.63	0.64	**0.88**	**0.85**	**0.85**	0.66	0.63	0.64	0.71	0.73	0.71
CMC	0.79	0.74	0.78	0.82	0.82	0.82	0.79	0.79	0.79	**0.94**	**0.94**	**0.94**	0.79	0.74	0.76	0.83	0.84	0.83
Cancer	0.69	0.69	0.69	0.73	0.73	0.73	0.79	0.76	0.77	**0.90**	**0.89**	**0.89**	0.69	0.69	0.59	0.83	0.79	0.89

We use standard evaluation index, Normalized Mutual Information (*NMI*), Accuracy (*ACC*), and F-score [38] to evaluate the clustering performance of CAABC-K-means algorithm and other algorithms. The corresponding results and the running time of the algorithm are analyzed in Tables 6–9.

The *NMI* is defined as follows:

$$NMI = 2\frac{I(X,Y)}{H(X)+H(Y)} \tag{20}$$

In the function, the I is mutual information between the sample and the label and H is the entropy.

In addition, the Accuracy (*ACC*) can be described as follow:

$$ACC = \frac{N_C}{N_S} \tag{21}$$

where N_S is the number of samples, and N_C is the correct number of samples.

In this paper, F-score is used to measure the accuracy of the clustering results. The performance comparisons among all the models are reported before and visualized in Table 9.

Precision (P), Recall (R), and F-measure (F) are often used to describe the accuracy of the clustering results. They are defined as follow:

$$P = \frac{TP}{TP + FP} \tag{22}$$

$$R = \frac{TP}{TP + FN} \tag{23}$$

$$F = \frac{2TP}{2TP + FP + FN} \tag{24}$$

where TP is True Positive, which is the number of the data that is classified into the cluster correctly. FN means False Negative, which is the number of the data that is not classified into the correct cluster. FP means False Positive, the number of the data that is classified into the cluster which is not belong. F-measure is the weighted harmonic average of Precision and Recall.

By analyzing the data of *NMI*, *ACC*, and running time from Tables 6–8, it is obvious that our proposed algorithm achieves excellent results than comparison algorithms. Specifically, Tables 7 and 8 show that CAABC-K-means obtains a better accuracy of classification than most compared algorithms, and Tables 6 and 8 show the proposed algorithm is more efficient and far superior to others. It is encouraging to find that CAABC-K-means achieves the highest F-core on ten datasets out of eleven datasets. The accuracy of the algorithm is better among the other clustering algorithms we compared with. Table 9 shows that the CAABC-K-means algorithm can offer a much better accuracy of classification than and similar efficiency to the traditional and some improved clustering algorithm.

From the above results, we can obtain that the CAABC algorithm performs better in terms of accuracy and efficiency. KMC algorithm is sensitive to initialization, so the computing time is longer, and it needs multiple iterations to reach the optimal value. ABC+K-means algorithm is easy to remain local optima, and it is difficult to achieve local optimization due to the stagnation of convergence in the later stage. The PSO+K-means and ABC_elite+K-means algorithms have improved the global optimization ability, however, the sensitivity of initialization of the clustering algorithm still exists, and the optimization effect of the algorithm is inconspicuous. Although the standard deviation has been reduced, the effect is inapparent. The improvement measures proposed in CAABC-K-means algorithm reduces the randomness in initializing, impairs the effect of an initial point, adaptive search mechanism to accelerate the algorithm convergence, and the disturbance and simulated annealing algorithm provide more possibility for solutions to escape from local optimum. Thus, the superiority of CAABC is obvious and the proposed algorithm can be considered as a feasible and efficient heuristic to find optimal solutions to clustering problems of allocating objects to K clusters.

5. Conclusions

Modeling the behavior of honey bees to search and solve problems has been the emerging area of swarm intelligence. In this paper, an improved artificial bee colony algorithm (CAABC) is developed to solve clustering problems. The method is inspired by the forage behavior in nature. The CAABC-K-means algorithm can be adapted to the process that the number of clusters known as a priori. In the CAABC algorithm, for the purpose of stabilizing the disturbance caused by the variety of the initial value of the clustering algorithm, we adapt the max–min distance product method in the initialization stage of the algorithm, which weakened the randomness of the initialization process to some extent. In the iterative process, chaotic disturbance and adaptive adjustment are added to obtain a better performance. After the comparison, chaos disturbance was added to the elite solution and unsatisfactory solutions in a certain percentage. The introduce of the "aggregate-disperse operation" speeds up the convergence of the algorithm and provides favorable conditions for escaping from the local optimal. Furthermore, on this basis, the global optimal solution is cross-operated to retain dominant individuals and

improve population diversity. Moreover, the simulated annealing criterion is integrated into the probability selection to achieve a better precision. By selecting the appropriate functions, the characteristics of ABC for group optimization are retained, and the local optimal solution can be avoided effectively. In addition, CAABC-K-means has the global search ability of CAABC, which reduces the number of iterations of K-means. The problem of poor global search ability of Kmeans algorithm is solved by the combination of two algorithms. Furthermore, according to the characteristics of the clustering algorithm, the impacts of the sample numbers and the distance between the sample centers are took into account in fitness selection, which reduces the possibility that the distribution of samples is excessive clustered. To evaluate the performance of the confluent algorithm, it is compared with other stochastic heuristic algorithms on several benchmark functions and real datasets. It can be concluded from the primary results of experience, which are very promising in terms of the accuracy of the solution found and the processing efficiency, that the CAABC-K-means clustering algorithm achieves better results.

The efficiency and accuracy of the algorithm have been improved, but the time complexity cannot be reduced effectively because of the location update formula which is still guided by global optimal solution. How to ensure the advantages of the existing algorithm while reducing the time complexity will be our next research direction. Applying the proposed algorithm to solve other optimization problems and improving the performance of the clustering algorithm will be considered in our future work.

Author Contributions: All authors contributed extensively to the work presented in this paper. Conceptualization, Q.J. and N.L.; methodology, Q.J.; software, Q.J. and N.L.; validation, Q.J. and N.L.; formal analysis, N.L. and Y.Z.; investigation, Q.J., N.L. and Y.Z.; data curation, N.L.; writing—original draft preparation, N.L.; writing—review and editing, N.L. and Y.Z.; supervision, Q.J.; funding acquisition, Q.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China, grant numbers 61673004 and the Fundamental Research Funds for the Central Universities of China (XK1802-4).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hartigan, J.A.; Wong, M.A. A K-Means Clustering Algorithm. *Appl. Stat.* **1979**, *28*, 100–108. [CrossRef]
2. Punit, R.; Dheeraj, K.; Bezdek, J.C.; Sutharshan, R.; Palaniswami, M.S. A rapid hybrid clustering algorithm for large volumes of high dimensional data. *IEEE Trans. Knowl. Data Eng.* **2018**, *31*, 641–654.
3. Ramadhani, F.; Zarlis, M.; Suwilo, S. Improve birch algorithm for big data clustering. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *725*, 012090. [CrossRef]
4. Cerreto, F.; Nielsen, B.F.; Nielsen, O.A.; Harrod, S.S. Application of data clustering to railway delay pattern recognition. *J. Adv. Transp.* **2018**, 377–394. [CrossRef]
5. Kouta, T.; Cullell-Dalmau, M.; Zanacchi, F.C.; Manzo, C. Bayesian analysis of data from segmented super-resolution images for quantifying protein clustering. *Phys. Chem. Chem. Phys.* **2020**, *22*, 1107–1114. [CrossRef] [PubMed]
6. Dhifli, W.; Karabadji NE, I.; Elati, M. Evolutionary mining of skyline clusters of attributed graph data. *Inf. Sci.* **2020**, *509*, 501–514. [CrossRef]
7. Eberhart, S.Y. Particle swarm optimization: Developments, applications and resources. In Proceedings of the Congress on Evolutionary Computation, Seoul, Korea, 27–30 May 2001.
8. Xiang, W.L.; Li, Y.Z.; He, R.C.; Gao, M.X.; An, M.Q. A novel artificial bee colony algorithm based on the cosine similarity. *Comput. Ind. Eng.* **2018**, *115*, 54–68. [CrossRef]
9. Karaboga, D. *An Idea Based on Honey bee Swarm for Numerical Optimization*; Technical Report-tr06; Erciyes University: Kayseri, Turkiye, 2005.
10. Zhang, X.; Zhang, X.; Ho, S.L.; Fu, W.N. A modification of artificial bee colony algorithm applied to loudspeaker design problem. *IEEE Trans. Magn.* **2014**, *50*, 737–740. [CrossRef]
11. Zhu, G.; Kwong, S. Gbest-guided artificial bee colony algorithm for numerical function optimization. *Appl. Math. Comput.* **2010**, *217*, 3166–3173. [CrossRef]

12. Zhang, P.H.; Jing-Ming, L.I.; Xian-De, H.U.; Jun, H.U. Research on global artificial bee colony algorithm based on crossover. *J. Shandong Univ. Technol.* **2017**, *11*, 1672–6197.
13. Cui, L.; Li, G.; Lin, Q.; Du, Z.; Gao, W.; Chen, J. A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation. *Inf. Sci.* **2016**, *367–368*, 1012–1044. [CrossRef]
14. Sharma, T.K.; Pant, M. Enhancing Scout Bee Movements in Artificial Bee Colony Algorithm. In Proceedings of the International Conference on Soft Computing for Problem Solving, Roorkee, India, 20–22 December 2011.
15. Yang, Z.; Gao, H.; Hu, H. Artificial bee colony algorithm based on self-adaptive greedy strategy. In Proceedings of the IEEE International Conference on Advanced Computational Intelligence, Xiamen, China, 29–31 March 2018; pp. 385–390.
16. Gao, W.; Liu, S.; Huang, L. A global best artificial bee colony algorithm for global optimization. *J. Comput. Appl. Math.* **2012**, *236*, 2741–2753. [CrossRef]
17. Gao, W.; Liu, S.; Huang, L. A novel artificial bee colony algorithm with Powell's method. *Appl. Soft Comput.* **2013**, *13*, 3763–3775. [CrossRef]
18. Wu, B.; Qian, C.; Ni, W.; Fan, S. Hybrid harmony search and artificial bee colony algorithm for global optimization problems. *Comput. Math. Appl.* **2012**, *64*, 2621–2634. [CrossRef]
19. Kang, F.; Li, J.; Li, H. Artificial bee colony algorithm and pattern search hybridized for global optimization. *Appl. Soft Comput.* **2013**, *13*, 1781–1791. [CrossRef]
20. Tasgetiren, M.F.; Pan, Q.K.; Suganthan, P.N.; Chen, H.L. A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. *Inf. Sci.* **2011**, *181*, 3459–3475. [CrossRef]
21. Guanlong, D.; Zhenhao, X.; Xingsheng, G. A Discrete Artificial Bee Colony Algorithm for Minimizing the Total Flow Time in the Blocking Flow Shop Scheduling. *Chin. J. Chem. Eng.* **2012**, *20*, 1067–1073.
22. Zhang, R.; Song, S.; Wu, C. A hybrid artificial bee colony algorithm for the job shop scheduling problem. *Int. J. Prod. Econ.* **2013**, *141*, 167–178. [CrossRef]
23. Pan, Q.K.; Wang, L.; Li, J.Q.; Duan, J.H. A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation. *Omega* **2014**, *45*, 42–56. [CrossRef]
24. Gao, K.Z.; Suganthan, P.N.; Pan, Q.K.; Tasgetiren, M.F.; Sadollah, A. Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion. *Knowl. Based Syst.* **2016**, *109*, 1–16. [CrossRef]
25. Li, J.Q.; Pan, Q.K. *Solving the Large-Scale Hybrid Flow Shop Scheduling Problem with Limited Buffers by a Hybrid Artificial Bee Colony Algorithm*; Elsevier Science Inc.: Beijing, China, 2015.
26. Liu, J.; Meng, L.-Z. Integrating Artificial Bee Colony Algorithm and BP Neural Network for Software Aging Prediction in IoT Environment. *IEEE Access* **2019**, *7*, 32941–32948. [CrossRef]
27. Karaboga, D.; Gorkemli, B. A quick artificial bee colony -qABC- algorithm for optimization problems. In Proceedings of the 2012 International Symposium on Innovations in Intelligent Systems and Applications, Trabzon, Turkey, 2–4 July 2012.
28. Zaragoza, J.C.; Sucar, E.; Morales, E.; Bielza, C.; Larranaga, P. Bayesian Chain Classifiers for Multidimensional Classification. In Proceedings of the International Joint Conference on Ijcai, Catalonia, Spain, 16–22 July 2011.
29. Park, J.Y.; Han, S.Y. Application of artificial bee colony algorithm to topology optimization for dynamic stiffness problems. *Comput. Math. Appl.* **2013**, *66*, 1879–1891. [CrossRef]
30. Xiang, Y.; Zhou, Y. A dynamic multi-colony artificial bee colony algorithm for multi-objective optimization. *Appl. Soft Comput.* **2015**, *35*, 766–785. [CrossRef]
31. Borchani, H.; Bielza, C.; Larranaga, P. Learning CB-decomposable multi-dimensional bayesian network classifiers. In Proceedings of the 5th European Workshop on Probabilistic Graphical Models, Helsinki, Finland, 13–15 September 2010; pp. 25–32.
32. Lou, A. A Fusion Clustering Algorithm Based on Global Gravitation Search and Partitioning Around Medoid. In Proceedings of the CSSE 2019: Proceedings of the 2nd International Conference on Computer Science and Software Engineering, Rome, Italy, 22–23 June 2019; p. 6.
33. Wu, M. Heuristic parallel selective ensemble algorithm based on clustering and improved simulated annealing. *J. Supercomput.* **2018**, *76*, 3702–3712. [CrossRef]
34. Fan, C. Hybrid artificial bee colony algorithm with variable neighborhood search and memory mechanism. *J. Syst. Eng. Electron.* **2018**, *29*, 405–414. [CrossRef]
35. Tellaroli, P. CrossClustering: A Partial Clustering Algorithm. *PLoS ONE* **2018**, *11*, e0152333.
36. Peng, K.; Pan, Q.K.; Zhang, B. An Improved Artificial Bee Colony Algorithm for Steelmaking-refining-Continuous Casting Scheduling Problem. *Chin. J. Chem. Eng.* **2018**, *26*, 1727–1735. [CrossRef]
37. Wenyuan, F.U.; Chaodong, L. An Adaptive Iterative Chaos Optimization Method. *J. Xian Jiaotong Univ.* **2013**, *47*, 33–38.
38. Yu, S.-S.; Chu, S.-W.; Wang, C.-M.; Chan, Y.-K.; Chang, T.-C. Two improved k-means algorithms. *Appl. Soft Comput.* **2018**, *68*, 747–755. [CrossRef]