A MapReduce-based Big Data Clustering using Swarm-inspired Meta-heuristic Algorithms

Razieh Tekieh[a], Zahra Beheshti [a,b]*
[a]*Faculty of Computer Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran*
[b]*Big Data Research Center, Najafabad branch, Islamic Azad University, Najafabad, Iran*
razieh.tekieh@sco.iaun.ac.ir
**z-beheshti@iaun.ac.ir (Corresponding author)*

Abstract
Clustering is one of the important methods in data analysis. For big data, clustering is difficult due to the volume of data and the complexity of clustering algorithms. Therefore, methods that can handle a large amount of data clustering at the reasonable time are required. MapReduce is a powerful programming model that allows parallel algorithms to run in distributed computing environments. In this study, an improved artificial bee colony algorithm based on a MapReduce clustering model (MR-CWABC) is proposed. The weighted average without greedy selection of the results improves the local and global search of ABC. The improved algorithm is implemented in accordance with the MapReduce model on the Hadoop framework to allocate optimal samples to the clusters such that the compression and separation of the clusters are preserved. The proposed method is compared with some well-known bio-inspired algorithms such as particle swarm optimization (PSO), artificial bee colony (ABC) and gravitational search algorithm (GSA) implemented based on the MapReduce model on the Hadoop framework. The results showed that MR-CWABC is well-suited for big data, while maintaining clustering quality. The MR-CWABC demonstrates an improvement of 7.13%, 7.71% and 6.77% based on the average F-measure compared to MR-CABC, MR-CPSO, and MR-CGSA, respectively.

Keywords: Big data, Clustering, Swarm-inspired Meta-heuristic Algorithm, MapReduce, Data mining, Distributed computing, Hadoop.

1 Introduction

Nowadays, a lot of data is generated by the advancement of technology and widespread application of the Internet, social networks, mobile phones, sensors, and so on. It can be said that the speed of data generation has surpassed our ability to process and extract useful information from it. This massive amount of data is called big data [1], and we are faced, apart from volume, with high production velocity and heterogeneity. Thus, the volume, velocity, and variety comprise three fundamental big data characteristics that should be considered in designing algorithms for analysis [1–3].
Data mining is one of the methods to discover knowledge from a database [4]. Supervised and unsupervised techniques are among data mining methods. Classification is one of the major supervised methods, and clustering and association rule mining constitute unsupervised ones. These latter two measure the similarities among records, patterns and their relationships. The main idea of a clustering algorithm is the groupings of data, so that objects within a group are similar, based on measurement criteria, and there is least similarity between the data of different groups. It can be said that the main function of clustering is the correct group definitions for the data [5]. Different types of clustering algorithms are based on partitioning, hierarchies, density, grids and models [6–10].
Due to the fact that most traditional clustering methods are not designed for high-volume and high-dimension data, and are also complex and time-consuming, they cannot be used in analyzing big data. Most of these methods are not scalable, and neither has this been considered in their design. Also, it has been assumed that all operations will be performed on a single machine and that all data exists in memory [1]. As a result, these methods are problematic, when faced with the high volumes and dimensions of big data.
Clustering is one of the data mining problems that can be transformed into an optimization problem, and which can use metaheuristic algorithms to solve them [11–14]. Metaheuristic algorithms, unlike data mining ones, are not designed to solve specific problems. These algorithms explore the problem space to find an optimal solution, and can find the right answer in terms of time complexity and accuracy, even with high scale and dimensions [15,16]. A combination of metaheuristic algorithms and parallel computing can be a successful means for big data analysis [11,17]. Parallel computing means breaking large problems into smaller pieces, and performing each with a different processor. In addition, processing can be carried out in parallel in a distributed environment [18]. The distribution can be done in a non-automated way, or an

automated way (such as with a MapReduce model). The difference between these two methods relates to the way they manage computing resources; in the MapReduce method, issues of network communication, load balancing, data distribution and fault tolerance are automatically addressed [19].
In this study, an advanced artificial bee colony algorithm called WABC is employed in a MapReduce model to cluster big data. To coordinate WABC with the clustering problem, each member of the population is considered as a solution, and which contains the proposed cluster centers. Also, the Davis-Bouldin (DB) index [20] has been used as an objective function. For this index, the rates of dispersion of objects within a cluster and intra-cluster distance are taken into account. In the calculation of the objective function, the calculation of intra-cluster data dispersion is the most costly part in terms of execution time, which is why the algorithm is implemented in accordance with the MapReduce model in a Hadoop framework. The findings of this study indicate the scalability of the proposed algorithm for clustering large volumes of data, while maintaining clustering quality.
This study is structured as follows: In section 2, the clustering and related works are reviewed. The proposed method for the improvement of artificial bee colony algorithm and mapping WABC with the MapReduce model for clustering is presented in Section 3. Experimental results and discussion are given in section 4. Section 5 provides the conclusions and future works.

2 Background

2.1 Clustering

Clustering is an unsupervised learning method, which objects are grouped based on their similarity inherent according to specified criteria [21]. The DB index is used as the evaluation function of clustering. This index is a function of the ratio of the total dispersion within the clusters to the distances between the clusters. Assume that $X=\{x_1,x_2,...,x_i,...,x_N\}$ is a set of *N* objects, $C=\{c_1,c_2,...,c_j,...,c_k\}$ is a set of *K* cluster centers, and $D=\{d_1,d_2,d_3,...,d_m\}$ represents the vector of values, where *M* denotes the number of dimensions of the problem. That is, each x_i object and each center of the cluster c_j is represented by a vector like *D*. According to the given description, the DB index is calculated as follows [20]:

$$s_{i,q}=\left[\frac{1}{N_i}\sum_{\vec{x}\in c_i}\left\|\vec{x}-\vec{c}_i\right\|\right]^{1/q} \tag{1}$$

$$d_{ij,t}=\left[\sum_{p=1}^{D}\left|c_{i,p}-c_{j,p}\right|^{t}\right]^{1/t}=\left\|\overrightarrow{c_i}-\overrightarrow{c_j}\right\|_t \tag{2}$$

Eq. (1) and Eq. (2) show the distribution of the members of a cluster, and the distance between two cluster centers. The larger that $s_{i,q}$ is, the more space the cluster occupies. The values of *q* and *t* are larger or equal to 1, and can be different values N_i shows the number of members in the cluster c_i Eq. (3) attempts the specification of the cluster that has the most separation from the desired cluster, and finally the DB index is calculated by Eq. (4):

$$R_{i,qt}=\max_{i\in K,i\neq j}\left\{\frac{s_{i,q}+s_{j,q}}{d_{ij,t}}\right\} \tag{3}$$

$$DB = \frac{1}{K}\sum_{i=1}^{K} R_{i,qt} \tag{4}$$

The clustering problem can be transformed into an optimization problem, where the DB index is considered as the evaluation function. Optimum allocation of the objects to clusters is done in several steps. A lower DB value shows better clustering.

2.2 Related works

Big data clustering is one of the main challenges in data mining due to the size, complexity, and variety of data that needs to be processed. Traditional clustering algorithms have been not designed to handle the massive volumes of data generated in today's digital world. Therefore, it is essential to have parallel distributed systems and software platforms, like MapReduce and Hadoop framework to effectively analyze large datasets [22]. In recent years, various methods have been proposed for clustering bulk data with the MapReduce model [23–28]. K-means is one of the most popular clustering algorithms, implemented with the MapReduce model [29–31] for big data clustering. Meta-heuristic algorithms which are used to solve optimization problems [32–35], are very popular in this regards. MR-CPSO [36] was introduced using particle swarm optimization (PSO) algorithm to get the result based on the shortest distance between the data and cluster centers. To make this algorithm scalable, an intra-clustering computation has been implemented using the MapReduce model. Yang and Lee [37] presented an algorithm for clustering big data using an ant colony algorithm. It also uses a scalable method for big data clustering. A clustering method was presented based on the glowworm optimization algorithm (GSO) and the MapReduce model (MR-CGSO) [38]. In the method, each member of the population is a cluster center, and the aggregation of these members represents the final clustering outcome. In another research [39], the authors proposed a MapReduce-based version of the fuzzy C-means algorithm (MR-FCM), which is scalable and can be run in a parallel-processing environment.

A MR-ABC, which uses the artificial bee colony (ABC) algorithm and a MapReduce model, was proposed to cluster the big data [5]. Its goal is to minimize the distance from cluster centers. In this algorithm, every position of the bee food is considered as a solution to the problem of clustering. One MapReduce task procedure is to calculate the distance between the data and cluster centers, which is a costly operation, with a high-execution time. The scalability, speedup and accuracy of the proposed clustering algorithm are suitable for use with big data.

MapReduce Black Hole (MRBH) [40] is another algorithm, which uses the black-hole optimization algorithm and a MapReduce model to cluster big data. A partitioning way [41] was proposed to reduce memory consumption in MapReduce model with a trie-based sampling mechanism. In another research [42], a clustering method based on MapReduce and enhanced grey wolf optimizer (MR-EGWO) was introduced for large-scale datasets. A MapReduce based K-means biogeography based optimizer (MR-KBBO) was suggested to analyze of large scale datasets [43]. Also, a map-reduce-based clustering recommendation system is introduced by improved whale optimization algorithm to cluster large datasets [44].

In another research [45] is explained how provenance can reduce total execution time in MapReduce programs which appending new data to the existing file in each rerun. Meddah and Belkadi [46] managed to minimize execution time by the use of MapReduce for extracting information from event logs. They used process mining techniques, and in map step, they mine patterns from execution traces, after that these small patterns are combined in reduce step.

In these methods, meta-heuristic algorithms or other algorithms like FCM have some advantages and disadvantages. For example, PSO and GWO have good exploitation or ABC has good exploration, but these algorithms suffer from inherent problems. A meta-heuristic based clustering with deep learning was proposed using MapReduce model [47]. The method employed density-based spatial clustering of applications with noise (DBSCAN) to detect random shapes and diverse clusters. In another research, a MapReduce-based model was introduced to automate intrusion detection using machine learning technique [48]. The detection was done by predicting unknown test scenarios and the data in the database was stored to minimize future inconsistencies. A student health monitoring system in IoT [49] was introduced using the MapReduce with deep learning framework was proposed. In the method, a 1-dimension convolution neural network was employed to extract the deep features in the map phase. In the reduce phase, the optimal subset of features was obtained by the adaptive bird rat swarm optimization (ABRSO).

A MapReduce-based fuzzy C-medoids clustering algorithm was proposed to cluster big data repository of documents datasets [24]. A MapReduce-based ensemble hierarchical clustering algorithm was introduced [50]. In the research, the results of different single clustering methods were combined by ensemble clustering algorithm and the MapReduce was applied to implement hierarchical clustering methods. In another study, an enhanced query optimization process of big data

was proposed using ant colony algorithm (ACO) and genetic algorithm (GA) based on MapReduce model. To cluster data, normalized K-means (NKM) was applied and the optimized query was obtained by ACO-GA [51].
A federated artificial neural network (FANN) was introduced to predict the critical path in a MapReduce workflow. A self-organizing map (SOM) neural network was employed to classify the health node [52]. In another research, an adaptive-chicken squirrel search algorithm driven deep belief network was proposed based on MapReduce model to predict the dropout and stress level of students [53]. A MapReduce-based distributed tensor clustering algorithm was suggested [54] to big data processing. Moreover, an enhanced K-means algorithm based on MapReduce and Hadoop framework was proposed [55]. A new distance measure was introduced in this method to calculate the distance between two data samples using sparse reconstruction.
Although different methods based on MapReduce model have been proposed, these methods encounter several disadvantages. For example, MR-CPSO and MRBH apply PSO and BH algorithms which have poor exploration and fall in local optima. MR-ABC employs ABC which shows slow convergence rate and faces the weak exploitation. MR-KBBO uses BBO and K-means algorithms. BBO demonstrates a premature convergence [49] and K-means algorithm depends on the initial values and falls into the local optimal if the initial values are not set well [50]. In addition, MR-FCM is based on FCM algorithm which encounters some disadvantages of K-means. DBSCAN also depends on input parameters and the density of data [56].
Despite the fact that the MapReduce and Hadoop provide a scalable and distributed computing framework to efficiently analyze big data, a new methods based on these tools are necessary to enhance the performance of big data clustering. Hence, an enhanced big data clustering artificial bee colony algorithm designed by the MapReduce model (MR-CWABC) is proposed in this study to improve the search ability of algorithm for finding the promising areas to efficient big data clustering.

3 The proposed method

In this study, three steps are proposed to cluster the big data and assign objects to the clusters more precisely. The first step is to introduce the weighted-average artificial bee colony (WABC) algorithm, so that the final solution of the algorithm is more accurate and closer to the global optimization. The second step is to determine the cluster centers, and the third step is related to the way of the objective function is calculated. The DB function is considered as the objective function.

3.1 Weighted-average artificial bee colony (WABC) algorithm

The WABC algorithm enhances the performance of ABC by first preventing both stagnation in the local optimization and early convergence, and second, increasing the speed of convergence to global optimization. In the WABC algorithm, three groups of bees are considered: employed, onlooker and scout bees in three phases. After initializing the bees' population, in the first phase (employed bees' phase), the new positions are obtained as follows:

$$v_{N_{s+i}}^{d}(t+1) = x_i^d(t) + rand(0,1) \times (mean^d(t) - x_i^d(t)) \tag{5}$$

where $rand(0,1)$ is a random number in the range of [0,1]. The value of i starts from 1 and continues to N_s . N_{s+i} indicates that new positions are created in addition to N_s previous positions. $mean^d$ is the weighted average of the positions in the previous generation in dimension d.
The new and old employed bees' positions are merged, and the best ones are chosen based on the value of the objective function and the number of employed bees. The weighted average of the employed bees' positions in the previous generation, and the position distance of each bee from the mean are used the above equation. This $mean^d$ is obtained in each generation in the employed bee:

$$mean^d(t+1) = \sum_{i=1}^{N_s} w_i(t) \times x_i^d(t) \tag{6}$$

where w_i is a weight assigned to the i^{th} bee position according to its fitness. Employed bees are sorted based on their fitness and the bee with the better fitness has a higher weight. The fitness is calculated [57] as follows:

$$fitness_i = \begin{cases} \dfrac{1}{1+fit_i} & if\,(fit_i \geq 0) \\ 1+abs(fit_i) & otherwise \end{cases} \tag{7}$$

Where fit_i is the value of the objective function for the i^{th} bee.
The w_i is computed as follows:

$$w_i = \frac{fitness_i}{\sum_{j=1}^{N_s} fitness_j} \tag{8}$$

$$\sum_{i=1}^{N_s} w_i = 1, w_1 \geq w_2 \geq ... \geq w_{N_s} \geq 0 \tag{9}$$

To prevent premature convergence and stagnation in the local optimization, after calculation of every new position, one of its dimensions is randomly selected and replaced by a random number in the range of the problem. In the proposed method, the second phase (onlooker bees' phase) is similar to standard ABC [57,58]. The new position of onlooker bees are calculated as follows [39]:

$$x_i^d(t+1) = x_i^d(t) + rand(-1,1) \times (x_r^d(t) - x_i^d(t)) \tag{10}$$

where $rand(-1,1)$ is a random number in the range of [-1,1]. x_r is a random bee and $r \neq i$.
In the third phase (scout bees' phase), the bees have not improved during the sequential iterations, their positions are updated based on three randomly selected bees as follows:

$$\begin{aligned} x_i^d(t+1) = x_i^d(t) &+ rand(0,1) \times (x_{r1}^d(t) - x_i^d(t)) \\ &+ rand(0,1) \times (x_{r2}^d(t) - x_{r3}^d(t)) \end{aligned} \tag{11}$$

where $r1, r2, r3 \in \{1,2,...,N_s\}, r1 \neq r2 \neq r3 \neq i$.

3.2 Mapping WABC with the MapReduce model for clustering

To adapt the WABC algorithm with data clustering, each bee is considered as a center of cluster. The structure for each bee is shown in Fig. 1. In this figure, there are three cluster centers, c_1 , c_2 and c_3 . Each cluster center has four properties. The main objective is to allocate data to the cluster centers so that the intra-cluster data distances' mean is minimized to the center, and the clusters' distances from one another each has the maximum possible value. The fitness value of each bee is determined by the DB index.
The framework of MR-CWABC is shown in Fig. 2. As shown in Fig. 2, the initial allocation of the clusters' centers is carried out and sent to the employed bees phase. These bees find new solutions. Then, the employed bees share their information with the onlookers' bees. The onlooker bees choose the best solutions, based on their fitness functions, and try to generate a more appropriate solution (cluster centers). Any bee that cannot find a better solution through the specified steps is changed by positioning in the scout bees phase using certain relationships.

This procedure continues until the algorithm termination condition is reached. After each solution changes in each phase, an evaluation operation (fitness function) must be performed, which will become more time-consuming with increasing dataset volume. An evaluating fitness function block is applied to calculate the DB index. The block uses a Hadoop Distributed File System (HDFS) for storage and MapReduce for processing. Hadoop [59] is an open-source software framework that enables distributed big data processing on commodity machines. It is responsible for task scheduling, concurrency control, resource management and fault tolerance.
MapReduce [60] is a parallel-programming model that seeks to hide details about parallel execution of programs from the user's point of view. Data is split by MapReduce into independent parts, with each section's size being a function of the data volume and the number of computational nodes. This model uses two separate functions for mapping and reduction. Data in a <key, value> format is entered into the map function, and after performing the specified calculation, output is generated with the <key, value> format, and referred to as intermediate results. These intermediate results are grouped based on the keys, and the reduce function generates the final aggregated results. The function combiner is an optional part in a MapReduce model, which locally performs aggregation on every mapping node to reduce the cost of sending all intermediate data to a reduction node. The architecture of this parallel-processing model is shown in Fig. 3.Fig. 4 shows the pseudo-code of MR-CWABC algorithm for clustering. The number of clusters, the number of food sources (N_S), and the maximum iterations are inputs of the algorithm. After running the Employed bees function, number of the bees in the population is double. In the next step, the population is sorted based on the bees' fitness, and then the top N_S bees are kept and the rest of them are eliminated.

Fig. 5 to Fig. 7 show the employed, the onlooker and the scout bees phases, respectively. In each of these phases, after the generation of a new position, the bee, in addition to the population, is added to the Bee Colony list. This list includes the bees for which the objective function should be calculated. In the employed bees function, first all new solutions are created and then the fitness function is called. In this case, the MapReduce function is called only once, and the overhead of calling this function is decreased. Each bee has a fine variable that shows how many consistently times it could not find a better solution. If the value of the fine is more than a threshold (determined by user), this bee is considered as a scout bee and its position will be changed by Scout Bees Function.

Fig. 8 shows the calculation of the fitness function. The DB index is used as the objective function. It is obtained based on two operations of the calculation of the intra-clusters dispersion value, and the calculation of the inter-cluster distances. The calculation of the intra-clusters dispersion will be time-consuming with respect to the big data volume. To make the algorithm scalable, the intra-clusters dispersion calculation is done through the MapReduce model. First, the dispersion within each cluster is computed through the MapReduce model. Then, the distance between cluster centers is specified for each bee, and the objective function value is computed. Later, the bee that corresponds to this bee is found in the population, and if the objective function value is better than the previous value, the information about its position and objective function is updated; otherwise, the bee will be fined.

4 Experimental results and discussion

In this section, the performance of the MR-CWABC is shown for big data clustering. The MR-CWABC has been studied from two aspects: clustering accuracy and parallelization efficiency. All tests are run on a Hadoop cluster of 10 nodes with each having a configuration of a dual-core Intel Xeon 2.9 GHz CPU and 3.7 GB of RAM. The Hadoop version used was 2.7.3.

4.1 Datasets

In this study, standard datasets were used to evaluate the proposed method [61,62]. To increase the size of the Iris, Magic, Electricity and Cover Type datasets, each record was repeated several times. Table 1 shows the characteristics of each dataset. The following datasets are considered:

- *Iris:* This dataset consists of three hidden patterns of three types of plants. The Iris dataset is identified using four attributes of diameter, length, thickness and width of the plant type.
- *Electricity:* This includes electricity prices for one Australian power company. The data is divided into two groups, low and high, using the clustering process and in accord with the price change level [62].
- *Magic:* This consists of recorded data for high-energy gamma particles in the atmosphere, collected by the Cherenkov telescope.

- *Cover Type:* This includes forest cover information for 30×30 meter areas in one of the US states. The dataset consists of seven tree species.
- *Heterogeneity Human Activity Recognition (HHAR):* The dataset contains the readings of two motion sensors commonly found in smartphones. Reading was recorded while users executed activities scripted in no specific order carrying smartwatches and smartphones.

4.2 Clustering Accuracy

To assess the accuracy of the clustering, the F-measure [63] is applied. This criterion uses precision and recall concepts in data retrieval. The F-measure treats each cluster like a query, and each class as the optimal result of the query. The following relation calculates the value of this index for the cluster *j* found by the algorithm and the class *i* that represents the data label in the standard dataset.

$$F(i,j) = \frac{2 \times recall(i,j) \times precision(i,j)}{recall(i,j) + precision(i,j)} \times 100(\%) \tag{12}$$

The recall and precision values are obtained as follows:

$$recall(i,j) = \frac{n_{ij}}{n_i} \tag{13}$$

$$precision(i,j) = \frac{n_{ij}}{n_j} \tag{14}$$

where n_{ij} is the number of the i^{th} class members in the j^{th} cluster, n_i is the count of the *i* class members, and n_j is the count of the *j* cluster members. The F-measure for the entire dataset is computed as follows:

$$F - measure = \sum_i \frac{n_i}{n} \max_j (F(i,j)) \tag{15}$$

where *n* is the total number of dataset members.

4.3 Algorithm evaluation criteria in parallel mode

Three criteria, called Scaleup, Speedup and Efficiency, are used to determine the performance of the algorithm in a parallel environment [64] Speedup shows how much of the parallel algorithm is faster than its serial mode. For Speedup calculations, the size of the dataset was considered constant, and the number of the computational nodes increased with a certain coefficient.

$$Speedup = \frac{T}{T_N} \tag{16}$$

where *T* is the runtime in the serial mode, and T_N is the runtime in the mode of using computational node *N*. In this experiment, the number of computational nodes increased by a factor of two.
The Scaleup represents the overall system efficiency in an incremental working load, and an increasing computational nodes mode. Speedup shows the ability of the parallel algorithm in optimal application of computational nodes.

$$Scaleup = \frac{T_{SN}}{T_{2SN}} \tag{17}$$

where T_{SN} represents the runtime for a dataset with a size of *S*, and *N* the number of computational nodes. T_{2SN} indicates the runtime when the size of the dataset and the number of the computational nodes are doubled. To show the efficiency of the algorithm when the number of the computational nodes is constant and the size of the dataset increases, an efficiency criterion computed as follows:

$$Efficiency = \frac{S_P}{N}$$

(18)

$$Efficiency = \frac{S_p}{N}$$

S_p represents Speedup, and *N* is the number of computational nodes.

4.4 Results and discussion

To evaluate the results, the proposed algorithm is compared with the MR-CABC [5], MR-CPSO [36] and MR-CGSA [65] by F-measure. Their results are shown in Table . MR-CABC, MR-CPSO, and MR-CGSA are respectively based on ABC, particle swarm optimization (PSO), and gravitational search algorithm (GSA) so that all use MapReduce model and evaluation function as same as MR-CWABC for clustering. The average F-measure of MR-CWABC shows the improvement of 7.13%, 7.71% and 6.77% compared with MR-CABC, MR-CPSO, and MR-CGSA, respectively.

Table 3 shows the efficiency of the MR-CWABC for increasing dataset size, in the range of ten computational nodes. The efficiency of the algorithm is improved on all five datasets by increasing the size, to 0.9 and better. Therefore, the algorithm has become more scalable, as demonstrated by increasing the size of the datasets. Therefore, the proposed method is more scalable as the problem size increased.
As shown in Fig. 9, the running time of the MR-CWABC on the datasets decreased almost linearly with increasing number of nodes in the Hadoop cluster. As seen in Fig. 10, the speedup of the MR-CWABC increases relatively linearly with increasing number of nodes. It is known that linear speedup is difficult to achieve because of the communication cost.
Fig. 11 (a) and Fig. 11 (b) demonstrate the running time and Speedup of MR-CWABC, MR-CABC, MR-CPSO and MR-CGSA on HHAR dataset, respectively. As seen in the figure, MR-CPSO has the best runtime. Also, MR-CWABC has the lower runtime than MR-CABC. The Speedup of MR-CWABC is closer to ideal.
Fig. 12 shows the results of Scaleup index for the MR-CWABC on the five datasets used in this study. To calculate the Scaleup, the number of records in each implementation stage is doubled. This index shows the system's throughput, and is expected to remain constant with simultaneous increases in workload and computing resources. As the results show for all five datasets, by increasing the size of the datasets and computational nodes, the Scaleup value is in the range of 0.82 to 0.91.
The results indicate that the amount of computation needed to find a solution depends on the size of the data. With increasing size, the advantages of using distributed computing with more computational nodes become more evident. All algorithms used in the study are meta-heuristic and have similar structures, but the ABC algorithm is capable of generating better results than the PSO algorithm. This is because the PSO algorithm works more on the basis of exploitation, whereas the ABC algorithm attempts to do a global search benefitting from the scout bees. Although ABC has the potential to stagnate in the local optimization and early convergence, the current study applies WABC which seeks to overcome these problems. The GSA is highly dependent on the primary initialization, with good ability in finding global optimization. However, if the primary initialization of the agents is not appropriate, the algorithm will be in trouble for convergence [66]. On the other hand, using a MapReduce model will save time and cost of computing for bulk data clustering. These results indicate the ability of the MR-CWABC to cluster large volumes of data at the appropriate time, while maintaining clustering accuracy.

5 Conclusion and future works

In this study, a method was presented for clustering big data using an improved artificial bee colony algorithm with a MapReduce model, called MR-CWABC. The MR-CWABC increases the accuracy of the results and prevents its early convergence to local optimization. The WABC algorithm was implemented in accordance with the MapReduce model and a Hadoop framework for optimizing big data clustering. The results of MR-CWABC compared with some well-known meta-heuristic algorithms implemented based on the MapReduce model and a Hadoop framework. The purpose of the experiments, performed on standard datasets, was to determine the quality of the solution and the efficiency of the parallel algorithm. The results obtained from the MR-CWABC on millions of records indicate good clustering accuracy compared to other algorithms. Also, with increasing dataset size, the scalability of the algorithm was demonstrated, and has a speedup close to that for a linear mode. However, the running time of MR-CWABC is more than MR-CPSO and MR-CGSA due to more calls the map-reduce task to evaluate a solution in the different phases.

As future works, the algorithm can run on more computational nodes using datasets with larger sizes. In addition, the efficiency of the algorithm on real datasets can be investigated. Moreover, the quality of clustering outlier data requires further evaluation. Furthermore, an automatic clustering can be performed by changing the structure of each bee position to determine the number of clusters in the algorithm. Finally, other machine learning, federated learning and explainable artificial intelligence (XAI) methods can be adapted for big data clustering to provide a powerful framework for distributed and parallel processing of large-scale datasets.

References

1. Tsai, C. W., Lai, C. F., Chao, H. C., et al. "Big data analytics : a survey", *J. Big Data,* **2**, pp. 1–32 (2015). doi:10.1186/s40537-015-0030-3
2. Naeem, M., Naeem, M., Jamal, T., et al. "Trends and future perspective challenges in big data", *Advances in intelligent data analysis and applications,* **253**, pp. 309–325 (2022).
3. Dafir, Z., Lamari, Y. and Slaoui, S. C. "A survey on parallel clustering algorithms for big data", *Artif. Intell. Rev.*, **54**, pp. 2411–2443 (2021).
4. Chen, M. S., Han, J. and Yu, P. S. "Data mining: an overview from a database perspective", *IEEE Trans. Knowl. Data Eng.*, **8**(6), pp. 866-883 (1996).
5. Banharnsakun, A. A "MapReduce-based artificial bee colony for large-scale data clustering", *Pattern Recognit. Lett.*, **93**, pp. 78–84 (2017).
6. Fahad, A., Alshatri, N., Tari, Z., et al. "A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis", *IEEE Trans. Emerg. Top. Comput.*, **2**, pp. 267-279 (2014).
7. Cano, L., Carello, G. and Ardagna, D. "A framework for joint resource allocation of MapReduce and web service applications in a shared cloud cluster", *J. Parallel Distrib. Comput.*, **120**, pp. 127–147 (2018).
8. Yan, Y., Sun, Z., Mahmood, A., Xu, F., et al. "Achieving Differential Privacy Publishing of Location-Based Statistical Data Using Grid Clustering", *ISPRS Int. J. Geo-Information,* **11**, pp. 404 (2022).
9. Ezugwu, A. E., Ikotun, A. M., Oyelade, O. O., et al. "A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects", *Eng. Appl. Artif. Intell.*, **110**, pp. 104743 (2022).
10. Vaghela, R. D. and Iyer, S. S. "A Comparative Analysis of Clustering Algorithm", *ECS Trans.*, **107**, pp. 2435 (2022).
11. Tripathi, A. K., Sharma, K., Bala, M., Kumar, A., Menon, V. G. and Bashir, A. K. "A Parallel Military Dog based Algorithm for Clustering Big data in Cognitive Industrial Internet of Things", *IEEE Trans. Ind. Informatics,* **17**(3), pp. 2134-2142 (2020).
12. Abualigah, L., Gandomi, A. H., Elaziz, M. A., et al. "Advances in Meta-Heuristic Optimization Algorithms in Big Data Text Clustering", *Electronics,* **10**, pp. 101 (2021).
13. Ghobaei-Arani, M. and Shahidinejad, A. "An efficient resource provisioning approach for analyzing cloud workloads: a metaheuristic-based clustering approach", *J. Supercomput.*, **77**, pp. 711–750 (2021).
14. Maheshwari, P., Sharma, A. K. and Verma, K. "Energy efficient cluster based routing protocol for WSN using butterfly optimization algorithm and ant colony optimization", *Ad Hoc Networks,* **110**, pp. 102317 (2021).
15. Beheshti, Z., Shamsuddin, S. M., Hasan, S. et al. "Improved centripetal accelerated particle swarm optimization", *Int. J. Adv. Soft Comput. its Appl.*, **8**, pp. 1–26 (2016).
16. Ajami-Bakhtiarvand, L. and Beheshti, Z. "A New Data Clustering Method Using 4-Gray Wolf Algorithm", *Nashriyyah-I Muhandisi-I Barq Va Muhandisi-I Kampyutar-I Iran, B-Muhandisi-I Kampyutar,* **19**, pp. 261–274 (2022).
17. Bashabsheh, M. Q., Abualigah, L. and Alshinwan, M. "Big Data Analysis Using Hybrid Meta-Heuristic Optimization Algorithm and MapReduce Framework", *Integrating Meta-Heuristics and Machine Learning for Real-World Optimization Problems,* **1038**, pp. 181–223 (2022).
18. Tsai, C.-F., Lin, W.-C. and Ke, S.-W. "Big Data Mining with Parallel Computing: A Comparison of Distributed and MapReduce Methodologies", *J. Syst. Softw.,* **122**, pp. 83–92 (2016).
19. Shirkhorshidi, A. S., Aghabozorgi, S., Wah, T. Y. et al. "Big data clustering: A review", *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, **8583 LNCS**, pp. 707–720 (2014).

20. Chou, C.-H., Su, M.-C. and Lai, E. "A new cluster validity measure and its application to image compression", *Pattern Anal. Appl.*, **7**, pp. 205–220 (2004).
21. Saxena, A., Prasad, M., Gupta, A., et al. "A review of clustering techniques and developments", *Neurocomputing*, **267**, pp. 664–681 (2017).
22. Ianni, M., Masciari, E., Mazzeo, G. M., et al. "Fast and effective Big Data exploration by clustering", *Futur. Gener. Comput. Syst.*, **102**, pp. 84–94 (2020).
23. Lalitha, R. and Rameshkumar, K. "Cluster-based convolution process on big data in privacy preserving data mining", *Int. J. Bus. Inf. Syst.*, **38**, pp. 17–33 (2021).
24. Sardar, T. H. and Ansari, Z. "Distributed Big Data Clustering using MapReduce-based Fuzzy C-Medoids", *J. Inst. Eng. Ser. B*, **103**, pp. 73–82 (2022).
25. Usha Lawrance, J. and Nayahi Jesudhasan, J. V. "Privacy Preserving Parallel Clustering Based Anonymization for Big Data Using MapReduce Framework", *Appl. Artif. Intell.*, **35**(15), pp. 1–34 (2021).
26. Mbyamm Kiki, M. J., Zhang, J. and Kouassi, B. A. "MapReduce FCM clustering set algorithm", *Cluster Comput.*, **24**, pp. 489–500 (2021).
27. Singh, V. K., Sabharwal, S. and Gabrani, G. "A new fuzzy clustering-based recommendation method using grasshopper optimization algorithm and Map-Reduce", *Int. J. Syst. Assur. Eng. Manag.*, **13**, pp. 2698–2709 (2022).
28. Suki Antely, A., Jegatheeswari, P., Bibin Prasad, M., et al. "Sparse FCM-Based Map-Reduce Framework for Distributed Parallel Data Clustering in E-Khool Learning Platform", *Int. J. Uncertainty, Fuzziness Knowledge-Based Syst.*, **31**, pp. 1–23 (2023).
29. Ma, L., Gu, L., Li, B., et al. "An improved K-means algorithm based on mapreduce and grid", *Int. J. Grid Distrib. Comput.*, **8**(1), pp.189-200, (2015).
30. Cui, X., Zhu, P., Yang, X., et al. "Optimized big data K-means clustering using MapReduce", *J. Supercomput.*, **70**, pp. 1249–1259 (2014).
31. Mao, Y., Gan, D., Mwakapesa, D. S., et al. "A MapReduce-based K-means clustering algorithm", *J. Supercomput.*, **78**, pp. 1–22 (2021).
32. Beheshti, Z., "A fuzzy transfer function based on the behavior of meta-heuristic algorithm and its application for high-dimensional feature selection problems", *Knowledge-Based Syst.*, **284**, pp.111191 (2024).
33. Sharifian, Z., Barekatain, B., Quintana, A. A., et al. "Sin-Cos-bIAVOA: A new feature selection method based on improved African vulture optimization algorithm and a novel transfer function to DDoS attack detection", *Expert Syst. Appl.*, **228**, pp. 120404 (2023).
34. Beheshti, Z. "BMPA-TVSinV: A Binary Marine Predators Algorithm using time-varying sine and V-shaped transfer functions for wrapper-based feature selection", *Knowledge-Based Syst.*, **252**, pp. 109446 (2022).
35. Beheshti, Z., Shamsuddin, S. M. and Sulaiman, S. "Fusion global-local-topology particle swarm optimization for global optimization problems", *Math. Probl. Eng.*, **2014**, pp. 1–19 (2014).
36. Aljarah, I. and Ludwig, S. A. "Parallel particle swarm optimization clustering algorithm based on MapReduce methodology", *4th World Congr. Nat. Biol. Inspired Comput. NaBIC*, Mexico City, Mexico, pp. 104–111 (2012). doi:10.1109/NaBIC.2012.6402247
37. Yang, J. and Li, X. "MapReduce based method for big data semantic clustering", *2013 IEEE Int. Conf. Syst. Man, Cybern. SMC*, Manchester, UK, pp. 2814–2819 (2013). doi:10.1109/SMC.2013.480
38. Al-Madi, N., Aljarah, I. and Ludwig, S. A. "Parallel glowworm swarm optimization clustering algorithm based on MapReduce", *2014 IEEE Symp. Ser. Comput. Intell*, Orlando, FL, USA, pp. 189–196 (2015).
39. Ludwig, S. A. "MapReduce-based fuzzy c-means clustering algorithm: implementation and scalability", *Int. J. Mach. Learn. Cybern*, **6**, pp. 923–934 (2015).
40. Tsai, C. W., Hsieh, C. H. and Chiang, M. C. "Parallel Black Hole Clustering Based on MapReduce", *2015 IEEE Int. Conf. Syst. Man, Cybern. SMC*, Hong Kong, China, pp. 2543–2548 (2015). doi:10.1109/SMC.2015.445
41. Slagter, K., Hsu, C. H. and Chung, Y. C. "An Adaptive and Memory Efficient Sampling Mechanism for Partitioning in MapReduce", *Int. J. Parallel Program*, **43**, pp. 489–507 (2015).
42. Tripathi, A. K., Sharma, K. and Bala, M. "A Novel Clustering Method Using Enhanced Grey Wolf Optimizer and MapReduce", *Big Data Res.*, **14**, pp. 93–100 (2018).
43. Tripathi, A. K., Sharma, K. and Bala, M. "Parallel hybrid bbo search method for twitter sentiment analysis of large scale datasets using mapreduce", *Int. J. Inf. Secur. Priv.*, **13**, pp. 106–122 (2019).
44. Tripathi, A. K., Mittal, H., Saxena, P., et al. "A new recommendation system using map-reduce-based tournament empowered Whale optimization algorithm", *Complex Intell. Syst.*, **7**, 297–309 (2021). doi:10.1007/s40747-020-00200-0
45. Chacko, A. M., Gupta, A., Madhu, S., et al. "Improving execution speed of incremental runs of MapReduce using provenance", *Int. J. Big Data Intell.*, **4**, pp. 186 (2017).
46. Meddah, I. H. A. and Belkadi, K. "Parallel Distributed Patterns Mining Using Hadoop MapReduce Framework", *Int. J. Grid High Perform. Comput.*, **9**, pp. 70–85 (2017).
47. Krishnaswamy, R., Subramaniam, K., Nandini, V., et al. "Metaheuristic Based Clustering with Deep Learning Model for Big Data Classification", *Comput. Syst. Sci. Eng.*, **44**, pp. 391–406 (2023).
48. Asif, M., Abbas, S., Khan, M. A. et al. "MapReduce based intelligent model for intrusion detection using machine learning technique", *J. King Saud Univ. - Comput. Inf. Sci.*, **34**, pp. 9723–9731 (2022).
49. Akhtar, M. M., Shatat, A. S. A., Al-Hashimi, M., et al. "MapReduce with Deep Learning Framework for Student Health Monitoring System using IoT Technology for Big Data", *J. Grid Comput.*, **21**, pp. 67 (2023).
50. Tian, P., Shen, H. and Abolfathi, A. "Towards Efficient Ensemble Hierarchical Clustering with MapReduce-based Clusters

Clustering Technique and the Innovative Similarity Criterion", *J. Grid Comput.*, **20**, pp. 34 (2022).
51. Kumar, D. and Jha, V. K. "An improved query optimization process in big data using ACO-GA algorithm and HDFS map reduce technique", *Distrib. Parallel Databases,* **39**, pp. 79–96 (2021).
52. Demirbaga, U. and Aujla, G. S. "Federated-ANN based Critical Path Analysis and Health Recommendations for MapReduce Workflows in Consumer Electronics Applications", *IEEE Trans. Consum. Electron,* (2023). doi:10.1109/TCE.2023.3318813
53. Kamakshamma, V. and Bharati, K. F. "Adaptive-CSSA: adaptive-chicken squirrel search algorithm driven deep belief network for student stress-level and drop out prediction with MapReduce framework", *Soc. Netw. Anal. Min.*, **13**, pp. 90 (2023).
54. Zhang, H., Li, P., Meng, F., et al. "MapReduce-based distributed tensor clustering algorithm", *Neural Comput. Appl.*, **35**, pp. 24633–24649 (2023).
55. Liu, Y., Du, X. and Ma, S. "Innovative study on clustering center and distance measurement of K-means algorithm: mapreduce efficient parallel algorithm based on user data of JD mall", *Electron. Commer. Res.*, **23**, pp. 43–73 (2023).
56. Hanafi, N. and Saadatfar, H. "A fast DBSCAN algorithm for big data based on efficient density calculation", *Expert Syst. Appl.*, **203**, pp. 117501 (2022).
57. Basturk, B. and Karaboga, D. "An artificial bee colony (ABC) algorithm for numeric function optimization", *IEEE Swarm Intelligence Symposium*, Indianapolis, IN, USA, pp. 12 (2006).
58. Karaboga, D. "An idea based on honey bee swarm for numerical optimization", *Technical Report TR06,* Erciyes University, Engineering Faculty, Computer Engineering Department (2005).
59. Shvachko, K., Kuang, H., Radia, S. and Chansler, R. "The Hadoop Distributed File System", *IEEE 26th symposium on mass storage systems and technologies (MSST). Washington, DC, USA*, pp. 1–10 (2010).
60. Dean, J. and Ghemawat, S. "Simplified data processing on large clusters", *Sixth Symp. Oper. Syst. Des. Implement.*, **51** (1), pp. 107–113 (2004).
61. Dua, D. and Graff, C. {UCI} Machine Learning Repository. (2017).
62. Bifet, A., Holmes, G., Kirkby, R., et al. "MOA: Massive Online Analysis", *J. Mach. Learn. Res.*, **11**, pp. 1601–1604 (2010).
63. Brief, T. "Agreement, the F-Measure , and Reliability in Information Retrieval", *J. Am. Med. Informatics Assoc.*, **12**, pp. 296–298 (2005).
64. Barr, R. S. and Hickman, B. L. "Reporting computational experiments with parallel algorithms: Issues, measures, and experts' opinions", *ORSA J Comput.*, **5**(1), pp. 2-18 (1993).
65. Sekar, K. and Padmavathamma, M. "Privacy preserving-aware over big data in clouds using GSA and MapReduce framework", *Int. J. Bus. Intell. Data Min.*, **16**, pp. 150–176 (2020).
66. Systems, I. J. I. and Sahoo, G. A "Review on Gravitational Search Algorithm and its Applications to Data Clustering and Classification", *Intell. Syst. Appl.*, **6**, pp. 79–93 (2014). doi:10.5815/ijisa.2014.06.09

Biographies

Razieh Tekieh received her BSc and MSc in Computer Engineering from Islamic Azad University Najafabad branch (IAUN), Najafabad, Iran. Her current research interests include Big Data Computing, Bio-Inspired Algorithms, Machine Learning and Data Mining. She can be reached at razieh.tekieh@sco.iaun.ac.ir.

Zahra Beheshti received her BSc and MSc in Computer Engineering from Islamic Azad University Najafabad branch (IAUN), Najafabad, Iran, and PhD and Post-Doc in Artificial Intelligence (Soft Computing) from Universiti Teknologi Malaysia (UTM), Malaysia. She is an assistant professor at Islamic Azad University Najafabad Branch. According to the report published by Stanford University and Scopus, she is one of the 2% influential scholars for three years (2020 to 2022), which depicts the 100,000 top scientists in the world. Her current research interests include Bio-Inspired Algorithms, Machine Learning, Fuzzy Expert System, Data Mining and Big Data Computing. She can be reached at z-behshti@iaun.ac.ir or beheshti_zahra@yahoo.com.

Fig. 1: The structure for one bee

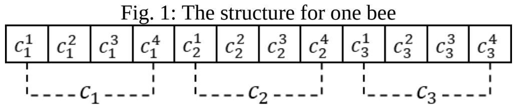

c_1^1 c_1^2 c_1^3 c_1^4 c_2^1 c_2^2 c_2^3 c_2^4 c_3^1 c_3^2 c_3^3 c_3^4
c_1
c_2
c_3

Fig. 2**:** The framework of the MR-CWABC

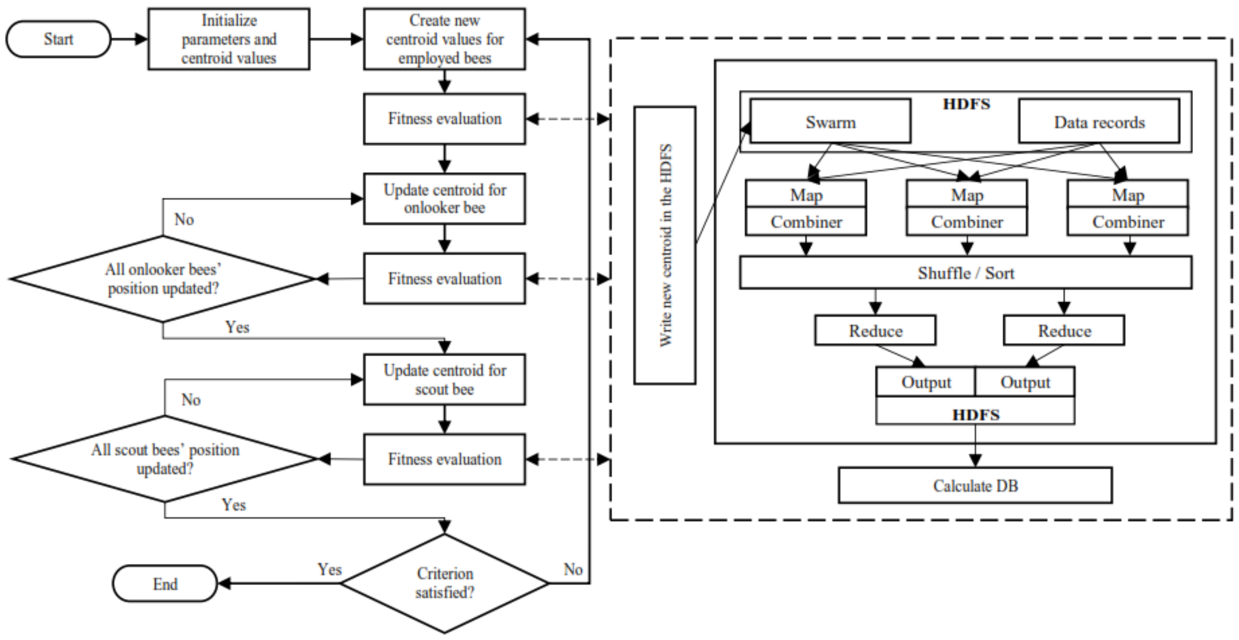

Start
Initialize parameters and centroid values
Create new centroid values for employed bees
Fitness evaluation
Update centroid for onlooker bee
No
All onlooker bees' position updated?
Fitness evaluation
Yes
Update centroid for scout bee
No
All scout bees' position updated?
Fitness evaluation
Yes
End
Yes
Criterion satisfied?
No
Write new centroid in the HDFS
HDFS
Swarm
Data records
Map
Combiner
Map
Combiner
Map
Combiner
Shuffle / Sort
Reduce
Reduce
Output
Output
HDFS
Calculate DB

Fig. 3: The MapReduce model

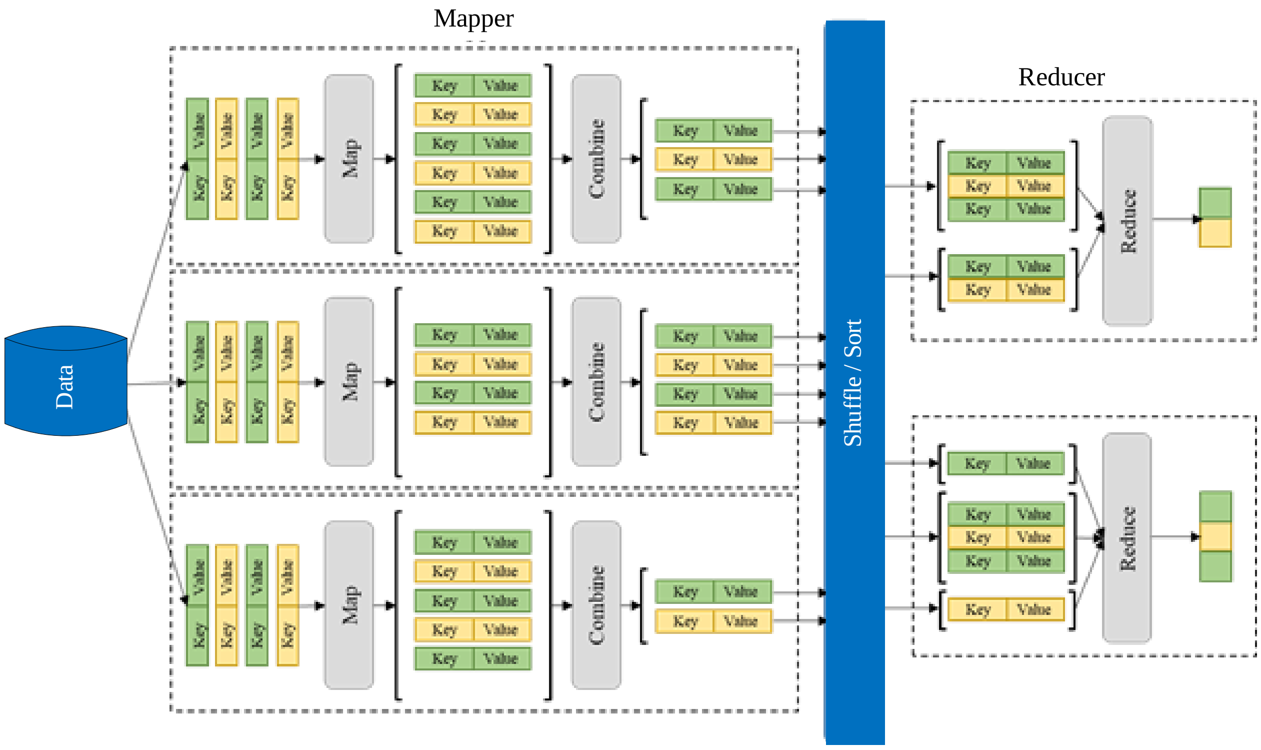


Fig. 4: The pseudo-code of the MR-CWABC Algorithm

```
MR − CWABC Algorithm
Input:
number of clusters, number of population, maxIteration
Output:
Cluster Centers
begin
1:    Generate an initial population and calculate fitness
2:     while iteration < maxIteration
3:          call Employed Bees Function (popualtion)
4:          sort population
5:          remove extra bees from population
6:          call OnLookerBeesFunction(population)
7:          call ScoutBeesFunction(population)
8:    end while
end
```

Fig. 5: The pseudo-code of the first phase

```
Employed Bees Function
Input:
population
begin
1:    foreach bee in population
2:        Calculate the new position using Eq. (5)
3:        add newBee to the population
4:        add newBee to BeeColony list
5:    end foreach
6:    call FitnessFunction(population, BeeColony)
end
```

Fig. 6: The pseudo-code of the second phase

```
Onlooker Bees Function
Input:
population
begin
1:    Calculate the new position using Eq. (10)
2:    add newBee to BeeColony list
3:    call FitnessFunction(population, BeeColony)
end
```

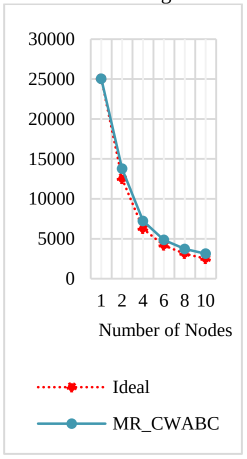
Fig. 7: The pseudo-code of the third phase

```
Scout Bees Function
Input:
population
begin
1:    foreach bee in population
2:        if bee.fine > fineThreshold
3:            Calculate the new position using Eq. (11)
4:            add bee to BeeColony list;
5:            call FitnessFunction(population, BeeColony)
6:        end if
7:    end foreach
end
```

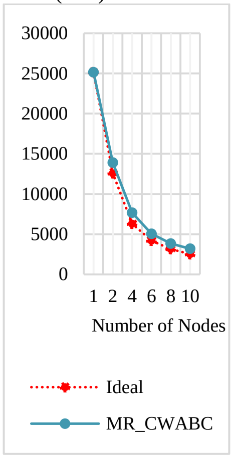
Fig. 8: The pseudo-code of Fitness Function

```
Fitness Function
Input:
population, BeeColony
begin
1:    write BeeColony in the HDFS
2:    run MapReduce job
3:    foreach bee in BeeColony
4:        Calculate DB base on Eq. (4)
5:        if DB < bee_cost
6:            find bee in population and update it
7:        else
8:            find bee in population and add fine
9:        end if
10:   end foreach
end
```

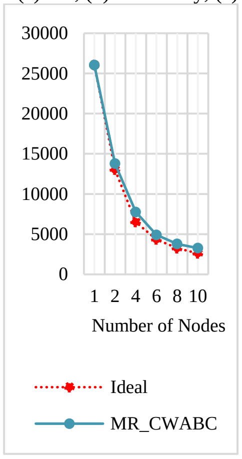
Fig. 9: Running Time (Sec.) of MR-CWABC on (a) Iris, (b) Electricity, (c) Magic, (d) Cover Type and (e) HHAR datasets



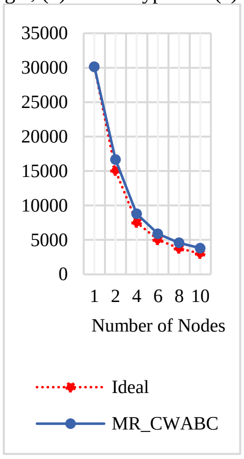
(a)



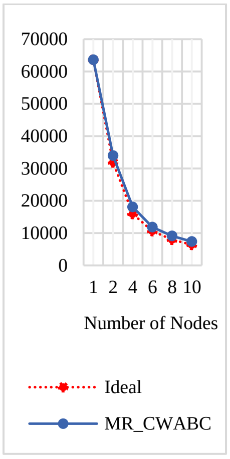
(b)



(c)

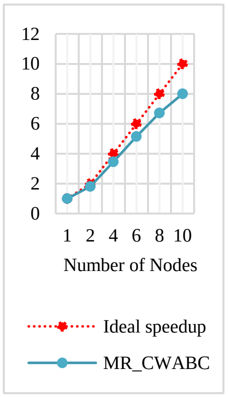


(d)

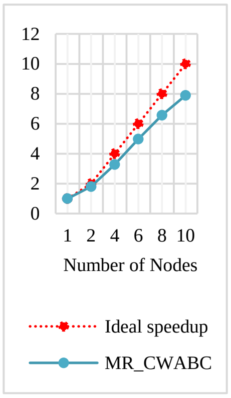


(e)

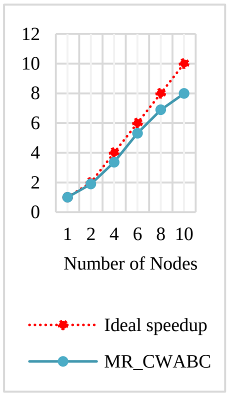
Fig. 10: Speedup of MR-CWABC on (a) Iris, (b) Electricity, (c) Magic, (d) Cover Type and (e) HHAR datasets



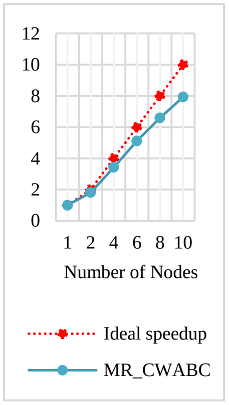
(a)



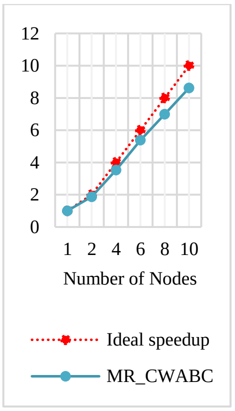
(b)



(c)

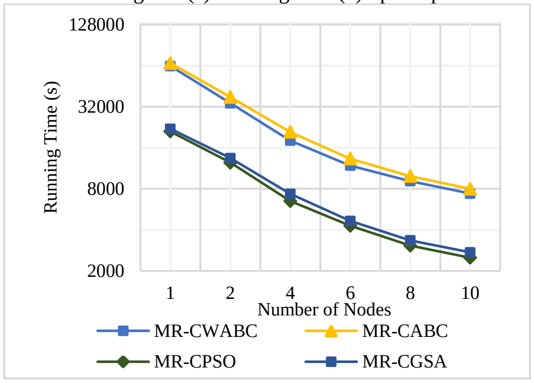


(d)

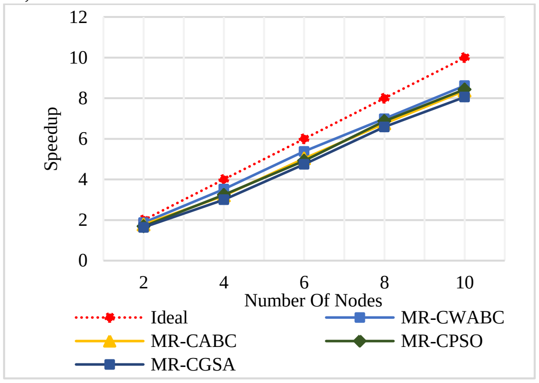


(e)

Fig. 11: (a) Running time (b) Speedup of MR-CWABC, MR-CABC, MR-CPSO and MR-CGSA on HHAR dataset

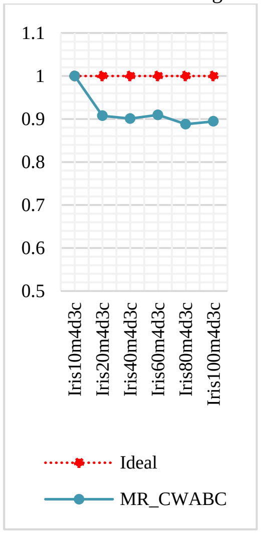


(a)

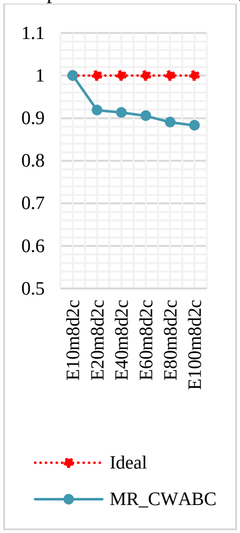


(b)

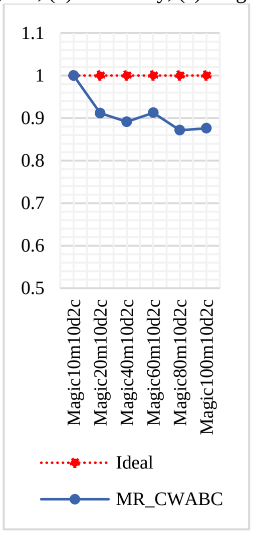
Fig. 12: Scaleup of MR-CWABC on (a) Iris, (b) Electricity, (c) Magic, (d) Cover Type and (e) HHAR datasets



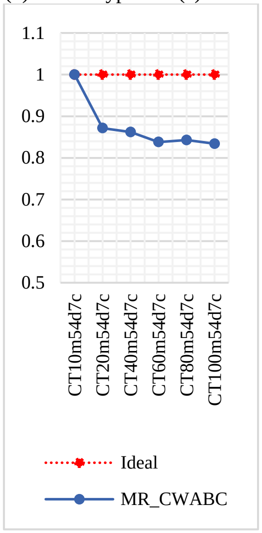
(a)



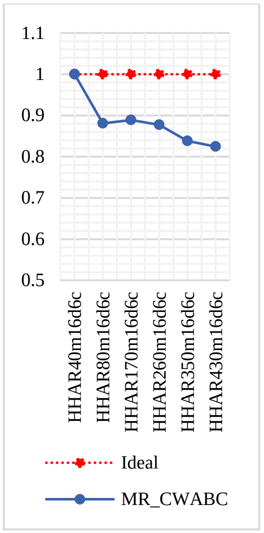
(b)



(c)



(d)



(e)

Table 1: Dataset characteristics

Dataset	**Number of Records**	**Number of Dimensions**	**Number of Clusters**
Iris	10,000,050	4	3
Electricity	10,421,760	8	2
Magic	10,080,600	10	2
Cover type	10,458,216	54	7
HHAR	43,930,257	16	6

Table 2: The F-measure values of compared algorithms

Dataset	***MR-CWABC***	***MR-CABC***	***MR-CPSO***	***MR-CGSA***
Iris	84.8%	74.9%	73.1%	69.5%
Electricity	61.3%	61.3%	61.2%	61.2%
Magic	55.2%	53.3%	52.2%	54.9%
Cover Type	59.1%	53.9%	54.7%	56.0%
HHAR	58.3%	54.1%	54.7%	56.9%

Table 1: The efficiency of MR-CWABC for different sizes of datasets

Dataset	**Number of records × 10^6**				
	Number of nodes				
	2	4	6	8	10
Iris	0.9	0.909	0.914	0.918	0.921
Electricity	0.91	0.918	0.923	0.927	0.93
Magic	0.91	0.914	0.92	0.922	0.924
Cover type	0.923	0.927	0.931	0.936	0.945
HHAR	0.926	0.932	0.938	0.942	0.955