Big Data Clustering Algorithm Based on Improved Artificial Bee Colony Algorithm and MapReduce (Postprint)

Authors: Sun Qian, Chen Hao, Li Chao

Date: 2019-05-10T00:00:00+00:00

Abstract

To address the low computational efficiency and clustering performance issues in big data clustering algorithms, a big data clustering algorithm based on an improved Artificial Bee Colony (ABC) algorithm and MapReduce is proposed. The Grey Wolf Optimizer (GWO) algorithm is integrated with the Artificial Bee Colony algorithm to simultaneously enhance the exploration and exploitation capabilities of the ABC algorithm. This strategy can effectively improve the performance of clustering processing. Chaotic mapping and opposition-based learning are employed as the initialization strategy for the ABC population to improve the quality of search solutions. The clustering algorithm is implemented based on Hadoop' s MapReduce programming model, achieving clustering processing for big data by minimizing the sum of squared intra-cluster distances. Experimental results demonstrate that the proposed algorithm effectively improves the clustering quality of large datasets while accelerating clustering speed.

Full Text

Abstract

Aiming at the problems of low computational efficiency and clustering performance in big data clustering algorithms, this paper proposes a big data clustering algorithm based on an improved artificial bee colony algorithm and MapReduce. The grey wolf optimizer algorithm is combined with the artificial bee colony algorithm to simultaneously improve the exploration and exploitation capabilities of the artificial bee colony algorithm. This strategy effectively enhances clustering performance. Chaotic mapping and opposition-based learning are employed as the initialization strategy for the ABC population to improve solution quality. The clustering algorithm is implemented based on Hadoop' s

MapReduce programming model, achieving big data clustering by minimizing the sum of squared intra-class distances.

Experimental results demonstrate that the proposed algorithm effectively improves clustering quality for big data sets while accelerating the clustering speed.

Keywords: data analysis; clustering algorithm; artificial bee colony algorithm; grey wolf algorithm; cloud computing; distributed computing

0 Introduction

Clustering is a crucial step in the field of data analysis, where high-quality data analysis is a key factor determining the performance of decision-making systems [**?**]. Many researchers have proposed effective clustering algorithms, but the computational time of these algorithms generally increases with the scale of the dataset, making time efficiency a critical performance metric for big data clustering [**?**]. MapReduce is a powerful programming model that supports parallel and distributed processing of large datasets, effectively improving algorithmic computational efficiency [**?**].

Numerous recent research results demonstrate that metaheuristic algorithms have significant potential for solving clustering problems. The metaheuristic algorithms that have attracted researchers' attention mainly include genetic algorithm (GA) [**?**], differential evolution algorithm (DE) [**?**], ant colony optimization (ACO) [**?**], and particle swarm optimization (PSO) [**?**], all of which support solving unsupervised clustering problems. Metaheuristic algorithms mostly adopt iterative optimization strategies, resulting in high time costs for finding global optimal solutions. Many researchers have combined the MapReduce programming model with metaheuristic algorithms, leveraging MapReduce' s parallel computing capabilities to reduce the overall processing time of metaheuristic algorithms. Reference [**?**] successfully implemented the k-nearest neighbor algorithm in the MapReduce model, significantly improving the processing efficiency of the K-nearest neighbor algorithm for large datasets.

Currently, a few researchers have implemented metaheuristic algorithms on the MapReduce programming model [**?**], including some effective optimization schemes for clustering problems. Reference [**?**] proposed a big data clustering algorithm based on MapReduce and K-means, which effectively clusters big data in high-dimensional space and achieves distributed processing through the MapReduce framework, reducing memory and computational costs. Reference [**?**] successfully combined artificial intelligence algorithms with distributed computing frameworks, achieving significant improvements in both computational efficiency and clustering effectiveness. However, due to the inherent limitations of genetic algorithms, this scheme' s clustering accuracy for big data did not reach ideal levels.

The artificial bee colony (ABC) algorithm is an artificial intelligence algorithm with extremely strong global search capabilities, and many researchers have

successfully applied ABC to solve clustering problems [**?**]. Although ABC has strong global search ability, its local exploitation capability for large-scale problems is weak. To address this weakness, the grey wolf optimizer (GWO) algorithm [**?**] is used to enhance ABC' s local exploitation capability, resulting in the GWOABC (grey wolf optimizer artificial bee colony) algorithm. ABC maintains global search capability through information sharing strategies, while GWO' s hunting strategy enhances local exploitation capability, preventing premature convergence. Additionally, a population initialization mechanism based on chaotic mapping and opposition-based learning is designed to effectively solve the problem of unstable search performance caused by random initialization. The GWOABC algorithm is implemented based on Hadoop's MapReduce model to improve processing speed for big data through distributed computing.

1.1 Grey Wolf Optimizer (GWO)

The GWO algorithm [**?**] is a swarm intelligence optimization algorithm that simulates grey wolf hunting behavior, considering three stages: encircling, hunting, and attacking prey. Algorithm 1 shows the pseudocode of the GWO algorithm.

Algorithm 1: Pseudocode of GWO Algorithm

Input: Population size of search agents N, dimension of solutions n, upper and lower bounds of solutions $[Ub_1, \ldots, Ub_n, Lb_1, \ldots, Lb_n]$, maximum iterations max_i.

Output: Optimal agent.

1. Initialize the grey wolf pack, where $(i \in [N])$, $x_j \in [Ub_j, Lb_j] | j \in [n]$.
2. Initialize α, β, δ, $t = 1$.
3. Calculate the fitness $f(X_i)$ for each search agent, where $i \in [N]$.
4. X_α = global best agent; X_β = second-best agent; X_δ = third-best agent.
5. While $(t < max_i)$ {
6. Foreach agent {
7. `Update current agent position using equations (1)-(5).`
8. }
9. Update X_α, X_β, X_δ.
10. Update parameters A, a, C.
11. $t = t + 1$.
12. }

The GWO model mainly includes the following elements:

a) **Social hierarchy.** The three best solutions are set as α, β, and δ wolves, with the remaining wolves designated as ω wolves.

b) **Encircling prey.** The mathematical model of the encircling strategy is represented as:

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \quad (1) \tag{1}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \quad (2) \tag{2}$$

where $\vec{A}$ and $\vec{C}$ are coefficient vectors, $\vec{X}_p$ is the prey position vector, $\vec{X}$ is the grey wolf position vector, t is the current iteration, and $\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a}$, $\vec{C} = 2\vec{r}_2$, where $\vec{r}_1, \vec{r}_2 \in [0, 1]$ are random vectors. Generally, a_1 is set to 2, and a decreases linearly from 2 to 0 over iterations: $a = 2 - 2 \cdot t/t_{max}$, where t_{max} is the maximum number of iterations.

c) **Hunting phase.** The three optimal solutions α, β, and δ wolves guide the GWO hunting process. The ω wolves update their positions based on α, β, and δ wolves using:

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}| \tag{3}$$

$$\vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}| \tag{4}$$

$$\vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}| \quad (3) \tag{5}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha \tag{6}$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta \tag{7}$$

$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\delta \quad (4) \tag{8}$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (5)$$

d) **Attacking prey.** Parameter a controls the attacking phase of GWO, with its value gradually decreasing during iterations. The two parameters $\vec{A}$ and $\vec{C}$ in GWO control the prey search process, where $\vec{A}$ takes values from $-2a$ to $2a$. If $|\vec{A}| < 1$, the grey wolves attack the prey. Parameter $\vec{A}$ controls the exploration speed of GWO; if $|\vec{A}| > 1$, the search diversity is high.

Reference [**?**] tested traditional GWO on different standard benchmark functions. According to their analysis, GWO easily falls into local optima for multimodal problems because most wolves in the population merely act as followers

guided by the three best solutions, resulting in limited search capability for the entire population. The values of parameters $\vec{A}$ and $\vec{C}$ in GWO determine the weight between global search and local exploitation. If $|\vec{C}| > 1$, candidate solutions diverge from dependent values; if $|\vec{C}| < 1$, candidate solutions converge to dependent values. Reference [**?**] found that GWO has strong search capability in early iterations and strong exploitation capability in later iterations. GWO' s search process guided by three optimal agents leads to insufficient diversity. This paper introduces GWO into the ABC algorithm, using ABC' s information sharing mechanism to improve information sharing among candidate solutions, thereby enhancing GWO' s global search capability.

1.2 ABC Algorithm

The ABC algorithm [**?**] mainly consists of three phases: employed bee phase, onlooker bee phase, and scout bee phase. The search strategies of employed bees and onlooker bees directly update one element of the solution vector randomly, as shown in equation (6):

$$v_{ij} = x_{ij} + \phi_{ij} \cdot (x_{ij} - x_{kj}) \quad (6)$$

where v_{ij} is the new solution obtained by mutating the j-th dimension variable of two different solutions, and ϕ_{ij} is a random number in the interval $[-1, 1]$.

ABC has strong global search capability, but the algorithm does not utilize the optimal solution to guide the search process, which may slow down convergence. Since optimal solution information can improve algorithmic convergence performance and GWO has strong ability to utilize optimal solution information, GWO is introduced into the ABC algorithm to enhance ABC' s performance.

1.3 MapReduce Programming Framework

To improve ABC' s processing efficiency for big data sets, this paper designs an implementation scheme for ABC in the MapReduce model, abbreviated as MR-GWOABC. MR-GWOABC' s mechanism for processing big data clustering tasks contains two main operations: updating cluster centroids and fitness evaluation. Centroid updating is based on ABC implementation. Fitness evaluation calculates the sum of Euclidean distances between each object and the centroids and finds the global optimal value. The clustering procedure divides data objects into clusters, minimizing the sum of Euclidean distances between all objects and centroids, which serves as the fitness function for ABC. The data clustering flow based on MR-GWOABC is shown in Figure 1 [Figure 1: see original paper].

2 Hybrid GWOABC Algorithm

The flow diagram of the proposed GWOABC algorithm is shown in Figure 2 [Figure 2: see original paper]. As can be seen, all steps of GWOABC are the

same as traditional GWO, with additional strategies added in the population initialization and information sharing modules. First, initialization parameters are defined, including population size N, solution space dimension dim, and maximum iterations. Then other parameters are calculated, including a, A, and C. The algorithm mainly consists of three phases: population initialization phase, GWO phase, and ABC phase.

1) Population Initialization Phase. High-quality initial candidate solutions are generated through chaotic mapping and opposition-based learning algorithms. Algorithm 2 describes the population initialization strategy. Observing lines 1 and 2 of Algorithm 1, Logistic chaotic mapping is used to obtain random variables $ch(k)$, which serve as the initial population X defined within variable space boundaries. The Logistic chaotic mapping function is defined as:

$$ch(k+1) = 4 \cdot ch(k) \cdot (1 - ch(k)) \quad (7)$$

where k is the number of iterations, set to 300, and the initial value $ch(0)$ is random.

Algorithm 2: Population Initialization Based on Opposition-Based Learning and Chaotic Mapping

Input: Population size of search agents $|N|$, solution dimension $j \in [n]$, upper and lower bounds of solutions $x_j \in [Ub_1, \ldots, Ub_n, Lb_1, \ldots, Lb_n]$, $k = 3$.

Output: Initialized population $|N|$.

1. Initialize population $|N|$ using chaotic mapping in n-dimensional space, obtaining $ch(k)$ using equation (7).
2. Elements of $X_i = (x_1, x_2, \ldots, x_n)$ are defined as $x_j = Lb_j + ch(k) \cdot (Ub_j - Lb_j)$, where $i \in [N]$, $x_j \in [Ub_j, Lb_j]$, $j \in [n]$.
3. Use OBL to obtain a set of population size N, where $i \in [N]$, with each new opposite solution defined as $x_j^* = Lb_j + Ub_j - x_j$.
4. Combine the two solution sets: $X = (X_i \cup X_i^*) \in |2N|$.
5. Calculate fitness $f(X)$.
6. Select top-N solutions and sort by fitness.

Line 4 derives the opposite population X^* set through the opposition-based learning algorithm, then combines the two sets into one solution $X = (X_i \cup X_i^*) \in |2N|$ and calculates its fitness $f(X)$.

2) GWO Phase. After generating the initial population, traditional GWO updates its parameters and current agent positions using equations (1)-(5).

3) ABC Phase. The ABC swarm shares information among candidate solutions and updates old solutions according to equation (6). To improve sequence randomness and reduce repetition, the Logistic chaotic mapping from equation (7) is used to generate the ϕ_{ij} term in equation (6).

The GWO procedure and ABC procedure are run iteratively until the maximum number of iterations is reached, with the optimal solution serving as the final result. ABC's global search improves GWO's global search capability, giving each wolf pack member the opportunity to share information while preserving the algorithm' s global search and local exploitation capabilities, effectively solving diversity issues and preventing premature convergence.

3 MR-GWOABC Clustering Algorithm

MR-GWOABC mainly has two modules when processing large-scale data clustering tasks: centroid updating processing and fitness evaluation. Centroid updating is completed based on MR-GWOABC, with the objective of calculating the sum of squared Euclidean distances between each object and the centroid to obtain the global optimal value. The clustering objective is to find the optimal assignment of data instances to minimize the sum of squared Euclidean distances, which serves as the fitness function for ABC. Figure 3 [Figure 3: see original paper] shows the data clustering flow based on MR-GWOABC.

According to the flow diagram shown in Figure 3, initial solutions are first generated as employed bee solutions. The clustering quality is calculated by computing the sum of squared Euclidean distances to evaluate solution fitness. Employed bees update and share their newly discovered solutions with onlooker bees, which then select solutions with better fitness. This procedure is repeated until the maximum number of iterations is reached. If a solution' s fitness cannot be improved within a certain time, scout bees regenerate new solutions. However, fitness evaluation for big data sets requires substantial computational time, so the MapReduce programming model is adopted to calculate fitness in a distributed manner.

Algorithms 3 and 4 show the Map and Reduce functions of the MapReduce model. The Map function first retrieves cluster centroids through the GWOABC algorithm, with data records stored in the Hadoop Distributed File System (HDFS). The Map function then extracts the centroids for each bee swarm, calculates the distance between each data point and the centroids, and returns the minimum distance for each centroid ID. The minimum distance for a centroid ID is modeled as the key of the Map function, with a new value obtained from the minimum distance. The Map function passes the new key-value pairs to the Reduce function, which aggregates all identical keys, calculates the overall average distance, and finally triggers the average distance key by calling the Map function' s emit(key, value), computing the fitness value for each bee in the swarm.

Algorithm 3: Map Function

Input: (key: record_{id}, value: record).

/* Initialization */ record_{id} = key; record = value; read(bee swarm);

foreach bee in bee swarm { bee_{id} = extract_{bee}*{id}(bee); CV* =

extract{centroids}(bee); min_{dist} = return_{min}*{distance}(record, CV); centroid*{id} = i; new_{key} = (bee_{id}, centroid_{id}); new_{value} = (min_{dist}); emit(new_{key}, new_{value}); }

Algorithm 4: Reduce Function

Input: key: (bee_{id}, centroid_{id}), value_{list}: (min_{dist}, 1).

Initialization: count = 0; sum_{dist} = 0; avg_{dist} = 0;

foreach value in value_{list} { min_{dist} = extract_{min}*{dist}(value); count = count + 1; sum*{dist} = sum_{dist} + min_{dist}; }

avg_{dist} = sum_{dist} / count; emit(key, avg_{dist});

4 Experimental Results and Analysis

This algorithm' s main contribution is using the MapReduce model to improve clustering processing speed for big data sets. Therefore, experiments evaluate the algorithm' s acceleration effect. The speedup metric is defined as S_p, with the calculation formula:

$$S_p = \frac{T_1}{T_N} \quad (8)$$

where T represents processing time on one Hadoop node, and T_N represents processing time using N Hadoop nodes.

The MR-GWOABC algorithm was simulated to evaluate clustering quality and parallel algorithm effectiveness. The algorithm was implemented in Perl script language. The experimental Hadoop platform consisted of 10 nodes, each configured with a 2.26 GHz CPU, 4 GB memory, and 180 GB disk, running Ubuntu 14.04 with Apache Hadoop 2.6.2.

To evaluate MR-GWOABC algorithm performance on big data sets, synthetic datasets shown in Table 1 were used. Four public datasets from the UCI Machine Learning Repository with different attributes were combined into a large dataset by randomly copying several backups to form a big dataset of 107 records.

Table 1 Attributes of Experimental Datasets

4.1 Clustering Algorithm Performance

F-measure is used as the evaluation metric for clustering quality, calculated from precision and recall metrics as:

$$F(i,j) = \frac{2 \cdot r(i,j) \cdot p(i,j)}{r(i,j) + p(i,j)} \quad (9)$$

where j represents the class generated by the clustering algorithm, i represents the original class label of the dataset, and r and p represent recall and precision, respectively.

Recall is defined as $r(i,j) = \frac{n_{ij}}{n_i}$, and precision as $p(i,j) = \frac{n_{ij}}{n_j}$, where n_{ij} represents the number of class i data points classified as class j, and n_i and n_j are the sizes of class i and class j, respectively. For a dataset of size n, the total F-measure is calculated as:

$$F = \sum_i \frac{n_i}{n} \max_j F(i,j) \quad (10)$$

where the upper bound of F is 1, and a higher F-measure value indicates better clustering quality.

For clustering performance comparison, the proposed solution is compared with the PKMeans algorithm and parallel K-PSO algorithm. The performance results of the three clustering algorithms are shown in Table 2 .

Table 2 F-measure Results of Three Clustering Algorithms

The MR-GWOABC algorithm significantly outperforms both PKMeans and parallel K-PSO algorithms. Although all three clustering algorithms use heuristic-based search methods, the K-means algorithm is sensitive to the initial centroid positions in the problem space and tends to converge to local optima near the starting points. Comparing the ABC algorithm with the PSO algorithm, ABC produces better solution quality than PSO. ABC' s search process includes both global search and local exploitation strategies, while PSO' s search process focuses more on local exploitation. In the ABC algorithm, employed bees and onlooker bees are responsible for local exploitation, while scout bees handle global search. If the search process falls into local optima, scout bees will randomly search for a new solution.

4.2 Clustering Algorithm Time Efficiency

4.2.1 Scalability Experiments The scalability of the MR-GWOABC algorithm was first tested by varying the number of Hadoop nodes in each run and 统计 ing the algorithm' s runtime and speedup metrics for the four datasets, as shown in Figures 4 [Figure 4: see original paper] and 5 [Figure 5: see original paper]. From Figures 4(a)-(d), it can be observed that MR-GWOABC' s runtime decreases as the number of Hadoop nodes increases, approaching ideal values. From Figures 5(a)-(d), MR-GWOABC' s speedup metric shows a linear improvement trend, with results close to ideal values.

4.2.2 Impact of Data Scale on Scalability The impact of dataset scale on MR-GWOABC algorithm scalability was also tested, with efficiency calculated using equation (11):

$$\text{Efficiency} = \frac{S_p}{N} \quad (11)$$

Table 3 shows the efficiency results obtained by MR-GWOABC on 10 nodes, demonstrating how the algorithm's performance changes with increasing dataset scale. The results show that efficiency for different dataset scales is above 0.90, leading to the conclusion that MR-GWOABC' s scalability improves as problem scale increases.

Table 3 F-measure Results and Efficiency Values for Different Dataset Scales

The MR-GWOABC algorithm is implemented based on MapReduce. Map tasks transform the dataset into key-value pairs, while Reduce tasks combine these data into collections. MapReduce distributed computing divides tasks into separate processes executed in parallel on large hardware clusters. MapReduce decomposes elements of big datasets into tuples, then further reduces tuples into smaller sets, significantly accelerating data processing through parallel handling of big data. Experimental results demonstrate that the MR-GWOABC algorithm can complete big data clustering within reasonable time while maintaining high solution quality.

5 Conclusion

To improve the processing efficiency and clustering performance of clustering algorithms, this paper designs a big data clustering algorithm based on an improved artificial bee colony algorithm and MapReduce. The GWO algorithm is combined with the ABC algorithm to simultaneously improve ABC' s exploration and exploitation capabilities, effectively enhancing clustering performance. Chaotic mapping and opposition-based learning are adopted as the initialization strategy for the ABC population to improve solution quality. The clustering algorithm is implemented based on Hadoop' s MapReduce programming model, achieving big data clustering by minimizing the sum of squared intra-class distances. Experimental results show that the MR-GWOABC algorithm can efficiently process large-scale datasets while achieving good clustering quality.

References

[1] Zhou Runwu, Li Zhiyong, Chen Shaomiao, et al. Parallel optimization sampling clustering K-means algorithm for big data processing [J]. Journal of Computer Applications, 2016, 36 (2): 311-315.

[2] Hai Mo. Survey of clustering algorithms for big data [J]. Computer Science, 2016, 43 (s1): 380-383.

[3] Song Jie, Sun Zongzhe, Mao Keming, et al. Research advance on mapreduce based big data processing platforms and algorithms [J]. Journal of Software, 2017, 28 (3): 514-543.

[4] Ding Y, Fu X. Kernel-based fuzzy C-means clustering algorithm based on genetic algorithm [J]. Neurocomputing, 2016, 188: 233-238.

[5] Wang Yongzhen, Chen Yan, Zhang Jinsong. Improved differential evolution algorithm for clustering [J]. Application Research of Computers, 2016, 33 (9): 2630-2633.

[6] Villar-Rodriguez E, Gonzalez-Pardo A, Ser J D, et al. A novel adaptive density-based ACO algorithm with minimal encoding redundancy for clustering problems [C]// Proc of IEEE Evolutionary Computation. 2016: 3139-3145.

[7] Santos P, Macedo M, Figueiredo E, et al. Application of PSO-based clustering algorithms on educational databases [C]// Proc of IEEE Computational Intelligence. 2018: 1-6.

[8] Anchalia P P, Roy K. The K-nearest neighbor algorithm using mapreduce paradigm [C]// Proc of IEEE International Conference on Intelligent Systems, Modelling and Simulation. 2015: 513-518.

[9] Yu Yanwei, Qi Jianpeng, Lu Yunhui, et al. Distributed swarm pattern mining algorithm in big spatio-temporal trajectory data [J]. Computer Engineering and Science, 2016, 38 (2): 255-261.

[10] Tsapanos N, Tefas A, Nikolaidis N, et al. Efficient MapReduce kernel K-means for big data clustering [J]. Automatica, 2016, 43 (2): 1-5.

[11] Sinha A, Jana P K. A hybrid MapReduce-based K-means clustering using genetic algorithm for distributed datasets [J]. Journal of Supercomputing, 2017 (8): 1-18.

[12] Xu Manshu, Wang Liwen, Qiu Jianfeng, et al. A fuzzy C-means clustering algorithm based on improved artificial by colony [J]. Computer Engineering and Science, 2016, 38 (6): 1238-1243.

[13] Xu Chenhua, Li Chengxian, Yu Xin, et al. Improved grey wolf optimization algorithm based on chaotic Cat mapping and Gaussian mutation [J]. Computer Engineering and Applications, 2017, 53 (4): 1-9.

[14] Mirjalili S, Mirjalili S M, Lewis A. Grey wolf optimizer [J]. Advances in Engineering Software, 2014, 69 (3): 46-61.

[15] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm [J]. Journal of Global Optimization, 2007, 39 (3): 459-471.

[16] Kishor A, Singh P K. Empirical study of grey wolf optimizer [C]// Proc of the 5th International Conference on Soft Computing for Problem Solving. Singapore: Springer, 2016: 1037-1049.