Parallel Bat Algorithm-Based Clustering Using MapReduce

Tripathi Ashish, Sharma Kapil and Bala Manju

Abstract As we are going through the era of big data where the size of the data is increasing very rapidly resulting into the failure of traditional clustering methods on such a massive data sets. If the size of data exceeds the storage capacity or memory of the system, the task of clustering will become more complex and time intensive. To overcome this problem, this paper proposes a fast and efficient parallel bat algorithm (PBA) for the data clustering using the map-reduce architecture. Efficient using the evolutionary approach for clustering purpose rather than using traditional algorithm like k-means and fast by paralyzing it using the Hadoop and map-reduce architecture. The PBA algorithm works by dividing the large data set into small blocks and clustering these smaller data blocks in parallel. The proposed algorithm inherits the bat algorithm features to cluster the data set. The proposed algorithm is validated on five benchmark data sets against particle swarm optimization with different number of nodes. Experimental results show that the PBA algorithm is giving competitive results as compared to the particle swarm optimization and also providing the significant speedup with increasing number of nodes.

Keywords Bat algorithm · Parallel bat algorithm · Map-reduce · Hadoop

T. Ashish (✉) · S. Kapil · B. Manju
Jaypee Institute of Information Technology Noida, Delhi Technological University Delhi, IP College of Women Delhi, New Delhi, India
e-mail: ashish.tripathi@jiit.ac.in; ashish07@gmail.com; se.jiit@gmail.com

S. Kapil
e-mail: kapil@ieee.org

B. Manju
e-mail: manjugpm@gmail.com

G.M. Perez et al. (eds.), *Networking Communication and Data Knowledge Engineering*, Lecture Notes on Data Engineering and Communications Technologies 4, https://doi.org/10.1007/978-981-10-4600-1_7

1 Introduction

Clustering is a popular analysis technique in data science, used in many applications and disciplines. Based on the values of various attributes of objects, it is used as an important tool and task to identify the homogeneous groups of the same. Clustering can be of following two types: hierarchal and partitioning. Hierarchal clustering works on two techniques, division and agglomeration of data clusters. Division is breaking large clusters into smaller ones, and agglomeration is merging small ones into nearest cluster. While in partition-based clustering, center of each cluster is used to compute an objective function and the value of this function is optimized by updating the center of clusters called as centroids. Clustering has a wide application in problems of data mining, data compression, pattern recognition, and machine learning. K-means is clustering algorithm which works on the greedy principle. It partitions the n data samples into k-clusters to minimize the sum of Euclidean distance of all data samples from their cluster centers. However, major drawbacks of this algorithm are as follows:

- No proper method to initialize. Generally done randomly.
- Due to high dependency on the initial centers, it may get stuck to suboptimal values, only quick solution is to execute it multiple times.
- Accuracy changes with change in number of clusters (k).
- In many cases, it tends to get stuck to local or suboptima.

To avoid the problem of initialization dependency and stuck into the local optima, the researchers now a days are using nature-inspired algorithms for the data clustering. Nature-inspired computing is the type of computing which takes its foundation from the biological aspects of nature which is humans and animals. Four powerful features of nature are self-optimization, self-learning, self-healing, and self-processing. As a self-optimizer, nature manages its resources efficiently so as to meet all enterprise needs in the most efficient way. The main problem with the nature-inspired algorithm is that they are computation intensive in nature. The nature-inspired algorithms are not able to give the satisfactory results in the reasonable amount of time. Today, the amount of data has increased manifold and its processing has become a huge problem. This big data is usually so large that its computations need to be distributed across thousands of machines so that computations can be finished in reasonable time period. Also there are the issues of parallelizing the computation, distributing data, and handling of failures which require large as well as complex codes to be dealt with. As a solution to this problem, a new abstraction has been designed that allows simple computations along with hiding the untidy details of fault tolerance, parallelization, load balancing, and data distribution in a library. This abstraction is conceptualized from map and reduces primitives present in Lisp and in other languages. Most of computations involve applying map operation to each “record” in the input. This computes a set of intermediate key and value pairs.

Then a reduce operation is applied to all the values that share same key, so that the derived data is combined appropriately. This model of user-specified map-reduce operations allow large computations to be parallelized easily and fault tolerance to be handled by re-execution.

The paper is organized as follows: Sect. 2 discusses the work done in this field. Basic bat algorithm is briefly described in Sect. 3. The proposed algorithm PBA has been introduced in Sect. 4. Section 5 presents the experimental results, and Sect. 6 concludes the work.

2 Related Work

With the increasing complexity and size of the data, distributed computation has become quite popular in recent years. Apart from processing of big data, distributed computing is widely used for the evolutionary computation and machine learning. When the size of data is too large and it is also unstructured, scaling of the machine learning techniques is needed to cope with it. It have been seen that when the search space is large in the case of evolutionary computation, then traditional sequential algorithms are not able to give satisfactory result in the specified time. In such case, distributed evolutionary computation becomes important. There are so many distributed computation models present in the literature such as GPU, CUDA, Cloud and Map Reduce-based implementation. Among all these, Map Reduce model is the recent research hot spot because of its simplicity and robustness. Dean et al. [1] given a MapReduce model for data processing on large clusters that have transformed the world of data processing and given the birth to big data processing platforms such as Hadoop.

Kim et al. [2] proposed a density-based clustering using Map-Reduce architecture which is robust to find the clusters with varying densities. The experimental results show that the proposed algorithm is found robust for the massive data applications of real-life data which is used for the experiments, and it is observed that the execution time decreases quite rapidly with increasing the number of machines. Clustering is the most popular machine learning techniques used in the industries and academics. With the evolution of massive data in the recent years, it becomes difficult to manage it with the traditional algorithms in the stand-alone systems.

Apart from the volume issue, velocity and variety of the data are also increasing rapidly. Cui et al. [3] proposed a k-means clustering algorithm which overcome the problem of iterations. The experiments performed on the clusters show that the proposed algorithm is efficient, robust, and scalable. Elsayed et al. [4] proposed ontology-based clustering algorithm for handling massive data. Amazons elastic MapReduce was used to perform the experiments. Li et al. [5] in his paper performed k-means clustering with bagging. Experiments are performed on four node cluster. Again results show that the execution time is decreasing with the increasing number of nodes. Remote sensing data size is too large, and traditional MATLAB implementation of support vector machine for such massive data becomes very time-

consuming process. In such cases, processing data at multiple cores becomes important. Cavallaroa et al. [6] used parallel support vector machine for the classification of classes of land cover types. The PiSVM algorithm achieved a good speedup while maintaining the same training accuracy as compared to traditional serial algorithm.

Nabb et al. [7] have proposed parallel PSO using MapReduce and confirmed that particle swarm optimization can be naturally implemented in the map reduce model without compromising with the any aspect of the original algorithm. Yingjie Xu et al. [8] developed iterative MapReduce-based PSO IMPSO algorithm for minimizing the thermal residual stress in ceramic composites. The proposed algorithm when executed on the cluster of 20 nodes has shown quite good speedup as compared to conventional PSO. Also satisfactory optimization results are obtained as compared to the conventional PSO.

Xingjian Xu et al. [9] modified Cuckoo search and implemented it using MapReduce architecture. The proposed algorithm MRMCS is compared with the parallel PSO using MapReduce (MRPSO). MMRCS shows better results as compared to MRPSO in terms of convergence in obtaining optimality. The proposed MRMCS when compared with MRPSO on the same number of nodes has shown two to four times speedup.

Abhishek Verma et al. [10] scaled Genetic algorithm using map reduce. The proposed models showed the convergence and scalability up to great extent. Author suggested that adding even more resources may enable us in solving compels problems since no performance bottlenecks were introduced in the implementation. Filomena Ferrucci et al. [11] given a framework for Genetic algorithm on Hadoop and tested it on three data sets. The developed framework showed quite promising results. The framework given by the author is can also be executed in the cloud environment with good performance.

3 Bat Algorithm (BA)

Bat algorithm is basically designed for the continuous optimization problems. This algorithm is inspired by the behavior of bats to catch their prey through echo. Microbats can find their prey as well as discriminate various types of insects even in the darkness by their echolocation property. Bat algorithm has its two major advantages: The first is frequency tuning and second is their emission rate. By using these two properties, bats can make a control between their exploration and exploitation. To mimic the behavior of bats, this algorithm uses the frequency-based tuning and pulse emission rate changes. This makes implementation simpler and better convergence when compared with other meta-heuristic algorithms. Also, BA maintains a balance of exploration and exploitation because keeping a simple fixed ratio of exploration to exploitation will not necessarily be an effective strategy.

Algorithm 1 Bat Algorithm (BA)

```
Randomly initialize the initial population and velocity of N bats
Set the value of pulse frequency f_i, pulse rates r_i, and loudness A_i
Evaluate the fitness fit of each bat
while stopping criteria is not satisfied do
  Adjusting the frequency f_i by Eq. (1)
  Compute the velocity v by Eq. (2)
  Update the location by Eq. (3)
  if (rand > r_i) then
    Select candidate solution among best solutions
    Generate a local solution using the selected best solution
  end if
  Create a new solution by flying randomly
  if (rand < A_i & f(x_i) > f(x_*)) then
    Accept the new solutions
    Improve the value of r_i
    Reduce the value of A_i
  end if
  Rank the bats;
  Compute the best x_*
end while
```

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \tag{1}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*)f_i, \tag{2}$$

$$x_i^t = x_i^{t-1} + v_i^t, \tag{3}$$

where, $\beta \in [0,1]$

4 Parallel Bat Algorithm

The main motivation behind the proposed parallel bat algorithm (PBA) is to leverage the strength of bat algorithm and to make it fast by map-reduce architecture. The advantage of bat algorithm is that it have a proper balance between exploration and exploitation. Generally, when the data size becomes large, then sequential evolutionary algorithms are not able to provide results in reasonable amount of time. PBA algorithm is designed to handle the large data sets by distributing data sets on different number of nodes and processing them in a parallel way. The proposed algorithm works in three modules.

- Bat Movement: The location of the bat is updated. If any bat is crossing the boundary of the search space, then it is reinitialized.
- Fitness Calculation: In this step, fitness of all the bats is calculated.
- Reduce Phase: Output from all the mappers is gathered, and the current best bat is updated.

Bat Movement and Fitness Calculation modules are implemented to improve the ability of the bat algorithm for mining massive data sets and updating population set. The key value pair of PBA is associated with each Bat by a numerical ID named *batID* as the key, and the bat information is kept in the value. The bat information contains bat-ID (batID), bat location (bat-loc), best bat location (bb-loc), bat fitness value (bat-fit), and best bat fitness value (bh-fit). The bat-loc and bb-loc are the structure of the cluster centroids. The input to the Fitness module, which is responsible for calculating the fitness of each bat, are the input data set and the output of the Bat Movement module. After the Bat Movement and Fitness modules are finished, the Reduce module updates the information of each bat by combining the two output files from the other two modules and then sends the bats to the next iteration.

4.1 Bat Movement Module

The goal of the Map-Reduce job is to move bats to the new location in the BatMove module. The parameters key, bat component, frequency, and loudness are initialize. After that, the new bat locations are calculated in lines 613. In the first iteration, the bat location will be the output to the reduce function directly because it does not have the current best bat location yet. In the second iterations, the new bat location is generated. The distance between the bat and the current best bat is calculated. The map function outputs the bat fitness to the reduce function by using the emit function when the bat has been moved. The reduce function gets the input, a list of bats produced by the map function. Reduce function outputs the pairs of bats directly. All the bats get the new location after the BatMove module is done.

4.2 Calculate Fitness

Mapper job is to compute the fitness of each bat in the calculate Fitness module. Each map function reads data about all the bats from the distributed cache for the calculation of the fitness values. The input data is split into smaller blocks, and each map function runs on one block. The distance between the cluster center of bat-loc and each data point is calculated by the getmin distance function. After getting the minimum distance of the bat, a new pair of batID and mindistance is formed and output to the reduce function. The reduce function will get the values in the summarized form. The reason is that the key value pairs are grouped by key and will be combined as list and send to the reduce function of Map-Reduce.

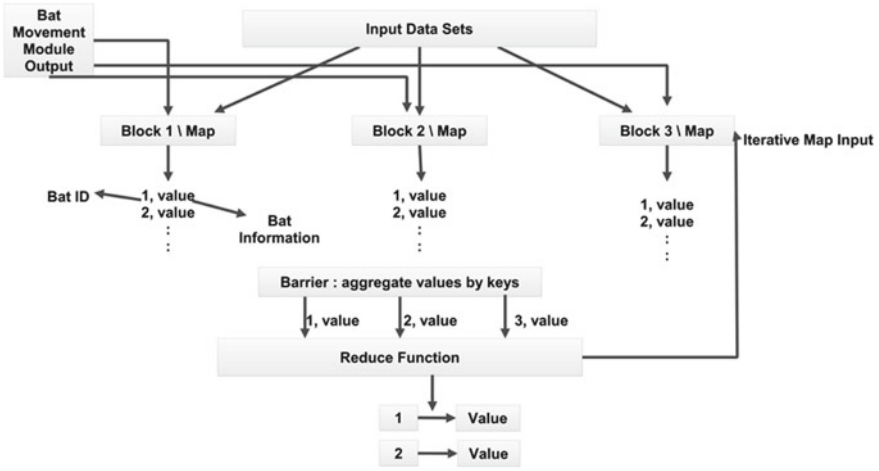

Fig. 1 Parallel bat algorithm on map-reduce

4.3 Combine module

The goal of the combine module is to merge the output files from the Bat Movement module and the calculate Fitness module to refresh the bat information about bat-loc, bat-fit, and best-fit. The fitness values of each bat is assigned to the bat-fit according to the batID number. Now in order to get the best bat, all the fitness values from the fitness output module are compared. After getting the best-fit, the best bat-loc can be found from the output file of the Bat Movement module by checking the batID. After Refreshing the bat information, the new bats are sent to the Bat Movement module to start the next iteration until the terminate criterion has been met (Fig. 1). The pseudocode of the PBA is presented in Algorithms 2–3.

5 Experimental Results

The experimental environment is a Hadoop cluster composed of four computers. All the computers are desktop with Intel Core i5-intel (2.30 GHz * 8), 4 GB RAM, and 1 TB hard disk. In the cluster, one of the desktop is set as the master, while the remaining computers are set as the slave nodes to do the MapReduce jobs. In order to implement proposed algorithm, Hadoop 2.6.0 is used for the MapReduce programming model, and the Java version is 1.7.0. The proposed algorithm is validated on five benchmark data sets: Iris, Glass, Wine, Magic and Poker Hand. The quality of clustering is tested using the total intra-cluster distance given by the PBA algorithm and is compared with the well-known particle swarm optimization algorithm. Each data set is carried out 15 times to get the average fitness value. Table 1 shows

Algorithm 2 PBA : Bat Movement Module

```
/* Map function */
map(Key: batID, Value: bat)
batID = Key
bat = getInfo (Value)
Generate a random number in [0,1]
//Moving bats
for each i on dimension of bat_loc do
  if first iteration then
    write( batID, bat)
  else
    bat.bat_loc_i += rand * (bat.bh_loc_i − bat.bat_loc_i) * f_i
  end if
end for
if rand > r_i then
  Select a solution among the best one
  Generate a local solution in the proximity of the best solution
end if
if rand < A_i f(bat_i) < f(bat_b est) then
  Accept new solution
  Decrease loudness and increase pulse rate
end if
bat.update(batID)
write(batID, bat)
/* Reduce function */
Reduce(Key: batID, Value: bat_list)
for each bat in the bat_list do
  write(batID, bat)
end for
```

Algorithm 3 PBA : Bat Fitness Module

```
/* Map function */
map(Key: dataID, Value: data)
dataID = Key
data = Value
//Select the bats from the Bat Movement Module
batList = getInfo (output of Bat Movement)
// Calculating minimum distance
for each bat of the batList input do
  mindistance = readmindistance(data, bat.bat_loc)
  write(bat.batID, mindistance)
end for
/* Reduce function */
Reduce(Key: batID, Value: mindistance_list)
for each mindistance in mindistance_list do
  sum+ = mindistance
end for
```

that the PBA algorithm outperforms PSO for all the data sets. The advantage of the PBA algorithm is in that the bats make a proper balance between the exploration and exploitation. In order to test the measure of the speedup, we have run our algorithm on the cluster of four nodes. Table 2 contains the running time on the different number of nodes. It can be observed from the Table 2 that proposed algorithm is showing good speedup as the size of the data set is increasing.

Table 1 Intra-cluster distance between considered algorithms

Algorithm	50	100	100	100	50
	Iris	Glass	Wine	Magic	Poker Hand
PBA	106.48	333.11	17163.32	1274574.42	660627.9
PSO	130.5	340.5	17888.77	1648270.8	6707348.5

Table 2 Running time (in seconds) with increasing number of nodes

Iteration	Single node	Two nodes	Three nodes	Four nodes
Iris	59.1	58.9	58.9	58.7
Glass	60	60	59	59.3
Wine	115	116.5	115.2	116.4
Magic	435	333.4	218.1	215.7
Poker Hand	27719.7	5077.1	4766.2	4235.2

6 Conclusion

In this paper, we proposed a novel clustering algorithm called parallel bat algorithm (PBA) to solve the massive data set clustering problems. The proposed algorithm leverages the strength of the bat algorithm and MapReduce. The simulation results show that PBA algorithm can be efficiently used for clustering large data set. Also simulation results show that the proposed algorithm performs very well when tested multiple nodes. In the future, the computation time can be reduced by implementing it on the spark architecture. Also the future work will include the testing of the proposed algorithm on some real-time massive data set.

References

1. D. Che, M. Safran, and Z. Peng, "From big data to big data mining: challenges, issues, and opportunities," in *Database Systems for Advanced Applications*, 2013.
2. X. Cui, P. Zhu, X. Yang, K. Li, and C. Ji, "Optimized big data k-means clustering using mapreduce," *The Journal of Supercomputing*, vol. 70, pp. 1249–1259, 2014.
3. J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, pp. 107–113, 2008.
4. A. Elsayed, H. M. Mokhtar, and O. Ismail, "Ontology based document clustering using mapreduce," arXiv preprint arXiv:1505.02891, 2015.
5. L. D. Geronimo, F. Ferrucci, A. Murolo, and F. Sarro, "A parallel genetic algorithm based on hadoop mapreduce for the automatic generation of junit test suites," in *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*, 2012.
6. Y.-J. Gong, W.-N. Chen, Z.-H. Zhan, J. Zhang, Y. Li, Q. Zhang, and J.-J. Li, "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art" *Applied Soft Computing*, vol. 34, pp. 286–300, 2015.

7. Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, and J. Fan, "Mr-dbscan: an efficient parallel density-based clustering algorithm using mapreduce," in *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, 2011.
8. H.-G. Li, G.-Q. Wu, X.-G. Hu, J. Zhang, L. Li, and X. Wu, "K-means clustering with bagging and mapreduce," in *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, 2011.
9. A. W. McNabb, C. K. Monson, and K. D. Seppi, "Parallel pso using mapreduce," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, 2007.
10. A. Verma, X. Llorà, D. E. Goldberg, and R. H. Campbell, "Scaling genetic algorithms using mapreduce," in *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on*, 2009.
11. Y. Xu and T. You, "Minimizing thermal residual stresses in ceramic matrix composites by using iterative mapreduce guided particle swarm optimization algorithm," *Composite Structures*, vol. 99, pp. 388–396, 2013.