I.J. Information Technology and Computer Science, **2017, 11, 72-78**
Published Online November 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2017.11.08

Parallel Bat Algorithm Using MapReduce Model

Kapil Sharma
Department of Computer Engineering, Delhi Technological University, Delhi, India
E-mail: kapil@ieee.org

Sanchi Girotra
Department of Computer Engineering, Delhi Technological University, Delhi, India
E-mail: sanchi.girotra@gmail.com

Received: 04 June 2017; Accepted: 10 August 2017; Published: 08 November 2017

***Abstract*—**Bat Algorithm is among the most popular meta-heuristic algorithms for optimization. Traditional bat algorithm work on sequential approach which is not scalable for optimization problems involving large search space, huge fitness computation and having large number of dimensions E.g. stock market strategies therefore parallelizing meta-heuristics to run on parallel machines to reduce runtime is required. In this paper, we propose two parallel variants of Bat Algorithm (BA) using MapReduce parallel programming model proposed by Google and have used these two variants for solving the Software development effort optimization problem. The experiment is conducted using Apache Hadoop implementation of MapReduce on a cluster of 6 machines. These variants can be used to solve various complex optimization problems by simply adding more hardware resources to the cluster and without changing the proposed variant code.

***Index Terms*—**Bat Algorithm, Big Data, COCOMO Model, Distributed System, Hadoop MapReduce Model, Parallel Algorithms.

I. Introduction

Over the last few years, Meta-heuristic (discover solution by trial and error) approximation algorithms are widely used to solve many continuous and combinatorial optimization problems. And these algorithms often find good solution with less computation effort than exhaustive, iterative and simple heuristic methods. Some of the meta-heuristic algorithms are virus optimization algorithm (VOA), symbiotic organism search (SOS), particle swam optimization (PSO), genetic algorithm (GA), cuckoo search (CS) etc. These algorithms are problem independent thus suits many optimization problems. Bat Algorithm is one of the swarm intelligence based nature inspired Meta - heuristic algorithm proposed by X.-S. Yang [1] in 2010 and it is based on the echolocation behavior of bats to detect its prey, avoid obstacle and to find its dwell in caves. BA has been successfully applied on many continuous optimization problems of topology design [2], effort estimation [3], economic dispatch [4], Medical Image Segmentation [15], Training Feed forward Neural Networks in e-Learning Context [14], Task Scheduling [16] etc. and it has shown better convergence to optimal solution than other meta - heuristic algorithms.

But BA takes reasonable amount of time to solve problems involving large volume of data (big data), involving huge fitness computation or having large number of dimensions. One solution to handle this would be to parallelize bat algorithm and scale it to large number of processors. But after program successfully parallelize, it must still needs to focus on communication of best bat across all the processors this can be done through shared memory which provides a global address space which parallel tasks can access asynchronously but it lacks locality of reference thus takes much time as compare to message passing interface (MPI) which provides an interface, protocol and semantic specifications to pass messages between parallel processes. These parallelized programs still need to handle distribution of data and node failure explicitly. In order to deal with these concerns, Google's Dean and Ghemawat [5] introduced an abstraction called MapReduce that provides a parallel design pattern for simplifying application development for distributed environments i.e. allows expressing the simple computation in that design model with hiding complex details of parallelization i.e. fault-tolerance, data distribution and load balancing in a library. These abstractions are inspired by map and reduce primitives present in Lisp and many other functional languages. This paper makes the following research contributions:

1. It demonstrates the modification of bat algorithms into the MapReduce Programming Model.
2. It proposes 2 variants of parallel bat algorithm.
3. It implements these variants on software effort estimation problem and demonstrates its scalability on different experimental set up.

From this proposed work, many complex optimization problems based on BA can easily be scaled only by increasing the no. of hardware resources.

The rest of the paper is organized as follows. Section II presents the study of BA and MapReduce in literature. In Section III, we revisit the Bat Algorithm. Section IV

provides more detailed explanation of MapReduce Model. In section V, we discuss how bat algorithm can be parallelized using the MapReduce Model. Section VI provides implementation details and we conclude with results in section VII and VIII.

II. Related Work

In 2004, Google's Dean and Ghemawat [5] published a white paper to parallelize large data on clusters by using map reduce model. After that MapReduce became widely-used parallel programming model and was employed on various large processing applications like Mining Interesting Infrequent Item sets from Very Large Data [17] etc. Nowadays, heuristic algorithms have started using Hadoop Map Reduce Model to scale data intensive problems as McNabb, et al. [6] have successfully parallelized Particle Swarm optimization Algorithm PSO on a MapReduce framework and evaluated it on RBF Network Training function and sphere function then in 2014 Wang, et al. [2] proposed MPSO (Map-PSO) and applied this on Performance-based Reward Strategy in stock markets which took less running time then sequential and MRPSO. Jin, et al. [7] used MapReduce model to parallelize GA and proposed MRPGA. Verma, et al. [13] presented parallel model of GA for data-intensive computing and tested it on one max problem. Keco and Subasi [14] proposed Model 2 and compared it with model 1 for same one max problem implementation. Then these models were used for various problems like Job Shop Scheduling Problem [3], automatic generation of JUnit Test Suites [12] etc. Lin, et al. [8] scaled Modified Cuckoo Search using MapReduce Architecture and evaluated it on Griewank function, Rastrigrin function, Rosenbrock function and Sphere function.

III. Bat Algorithm

BA is swarm intelligence based meta-heuristic optimization algorithm proposed by X.-S. Yang [1] in 2010. BA finds the solution in the search space by exploiting the echolocation behavior of bats in which they locate their prey by emitting a short - loud sound pulse and listen for the echo that bounces back from the surrounding obstacle from which they see the time delay from the emission of sound pulse to perception of the echo and the time difference between their two ears and also the loudness variations (signal intensity) of the echo's to get a view about surrounding and prey size.

In addition to this, BA uses a dynamic strategy for exploration (global search) and exploitation (local search) to improve the solution. Like the variations in pulse emission rates r and loudness A_0 controls exploration and exploitation in BA.

In original bat algorithm, the bat behaviour of finding the best location is captured to optimize the fitness function of the problem to be solved. It consists of the following steps:

Step 1: Initialization of Bat Population

Initially, N bats are randomly spread over the search space as they have no idea where the prey is located thus population is randomly generated for each dimension d with default values for frequency, velocity, pulse emission rate and loudness.

$$x_{ij} = x_{\min j} + rand(0,1)\ (x_{\max j} - x_{\min j}) \quad (1)$$

Where $i = 1, 2 \ldots N$, $j = 1, 2 \ldots d$, $x_{\min j}$ and $x_{\max j}$ are lower and upper boundaries for dimension j respectively.

Step 2: Update Process of Frequency, Velocity and Solution

In the iterations of BA, each bat in the population emits sound pulses of random frequency dispersed between [f_{min}, f_{max}] and this frequency controls the speed and new position of bat using below equations:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta \quad (2)$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x^*) f_i \quad (3)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (4)$$

Where β denotes a randomly generated number with in the interval [0, 1] and x^* is the current global best solution which is located after comparing all the solutions. Upper and lower bounds of frequency are chosen such that it is comparable to the size of search space of that variable.

Step 3: Local Search

Each bat performs Random walk in order to exploit a position around the best solution.

$$x_{new} = x_{old} + \varepsilon A^t \quad (5)$$

Here x_{old} is selected best solution from current best solutions, ε denotes a random number within the interval [-1, 1], while A^t is the average loudness of all the bats at that time step.

Step 4: Updating Loudness and Pulse Emission Rate

If new solution is improved than previous solution i.e. as the bat approach their prey, loudness of emitted sound pulse decreases and pulse emission rate increases so loudness and pulse rate needs to be updated using below equations:

$$A_i^{t+1} = \alpha A_i^t \quad (6)$$

$$r_i^{t+1} = r_i^0 (1 - e^{-\gamma t}) \quad (7)$$

Where α and γ are constants. r_i^0 & A_i^0 consist of random

values and $r_i^0 \in [0, 1]$ and $A_i^0 \in [1, 2]$.

Step 5: Find the current best solution.

Find the current best bat by comparing last best fitness value with improved bat fitness if its better then update the best bat solution.

After all the iterations, the current best bat x^* gets improved and acts as the solution to the problem.

IV. MapReduce Model

Map Reduce is programming model and an abstraction proposed by Google - Dean and Ghemawat [5] for processing large volumes of data in parallel. This Model exploits parallelism by splitting the input data into blocks which are processed by the map and reduce tasks in a completely parallel manner. And it is applicable to those parallel computing problems in which same operation is applied on each data item and whose output is consumed by other operations in groups.

MapReduce Programming Model takes the input in the form of (key; value) pairs and operate on data in mainly three stages:

1. Map Stage:

In the map stage, the user defined map () function takes as input a single (key; value) pair and produces as output any number of intermediate (key; value) pairs. It is crucial that the map operation is stateless - that is, it operates on one pair at a time. This allows for easy parallelization as different inputs for the map can be processed in different machines.

$$MAP : (K_1, V_1) \Rightarrow LIST(K_2, V_2) \quad (8)$$

2. Shuffle Stage:

In shuffle phase, all of the map's output (key; value) pairs are sorted and all values associated with same key is grouped together in a value lists which is then send to reduce () function. This occurs automatically without any programmer intervention.

3. Reduce Stage

In the reduce stage, each user defined reduce () function takes all of the values associated with a single key k and outputs a multi set of (key; value) pairs with the same key k. This highlights one of the sequential aspects of MapReduce computation i.e. all of the maps need to finish before the reduce stage can begin. And in the reduce step, the parallelism is exploited by observing that reducers operating on different keys can be executed simultaneously.

$$REDUCE : (K_2, list(V_2)) \Rightarrow LIST(V_2) \quad (9)$$

Apache Hadoop is an open source software framework for distributed processing which has successfully implemented Google's MapReduce and file system in its modules.

V. Parallel Bat Algorithm using MapReduce Model

As have seen that in each iteration of original BA, all bats find a new location either by flying randomly using equation (5) or by adjusting their frequency and updating their velocity, location using equation (2), (3), (4). Each bat then evaluates the fitness of new solution and accordingly move to new position if that's better than the current position.

In original BA, the steps that can be parallelized i.e. can be executed independently are:

- Each bat updating its position.
- Iterations of generations can be executed in parallel.

But iterations can't be parallelized as population of bat should improve from generation to generation and initial iteration results should be utilized in the next iterations.

Based on above idea, to transform BA into map and reduce primitives following points need to be defined:

- Determine the input to MapReduce Model.
- Determine the jobs of mapper and reducer tasks.
- Exchange information between parallel map tasks.

A. Input Representation

In Parallel BA, large initial random population of bats is given as input to MapReduce framework which splits them into chunks and distributes them across the map tasks.Each bat in input population is represented by key/value pairs:

K_1: String representing individual bat
V_1: fitness

Here the key consists of set of attributes representing bat like position, frequency and velocity whereas value is the fitness of that bat. For e.g. in our software effort optimization problem (K_1: V_1) are:

Coefficient a; Coefficient b; Frequency; Velocity a; Velocity b: Fitness

And g_{best} bat of initial population is also sent in the following form:

Coefficient a; Coefficient b: Fitness

B. Mapper in Parallel Bat Algorithm

Job tracker initiates map tasks equal to the no. of population split and assign map task to each of these population splits. Then each mapper (map task) calls the user defined map () function for each individual (bat: fitness pair) in the population split.

In this paper, we present two variants of parallel bat algorithm. Both variants have corresponding map () algorithms which encapsulates bat updating process. First variant uses one map phase for one generation of bat algorithm i.e. population is evolving generation by generation while in second variant, one map phase for all the generations is used i.e. each bat is evolving for N

generations with in a map () function. That means in variant 1, N Map cycle execute whereas in Variant 2 only one map cycle will generate the optimum output.

As shown in Fig. 1 pseudo code of variant 1, map () function explores and exploits new solution for given bat (input key) using x^* representing m_{best}. And outputs improved bat which is used as input in the next generations (loop cycle). This map () function is called for each bat in the population for N generations.

And as shown in pseudo code in Fig. 2, map () function evolves the given bat for N generations using m_{best} (mapper best) as x*. In this no intermediate data are emitted by any Mapper and final evaluated g_{best} is considered as the output of the job.

At the end of each map task these improved m_{best} are written to the distributed file system which are then evaluated at the end of map phase to calculate g_{best}.

```
[1]   bat ← BATREPRESENTATION(key)
[2]   fitness ← value
[3]   # Generate new solution by adjusting
      frequency and updating velocities and
      locations/solutions.
[4]   newBat ← NEWBAT (bat, m_best)
[5]   if (rand > r_i)
[6]   newBat ← LOCALBAT (m_best)
[7]   end if
[8]   newFitness ← CALCULATEFITNESS
      (newBat)
[9]   if (rand < A_i & newFitness < fitness)
[10]  # Accept the new solution.
[11]  bat = newBat
[12]  fitness = newFitness
[13]  end if
[14]  if (fitness < m_best.fitness)
[15]  m_best = bat
[16]  end if
            EMIT (bat, fitness)
```

Fig.1. Pseudo Code of Variant 1

```
[1]  bat ← BATREPRESENTATION(key)
[2]  fitness ← value
[3]  while t <Max number of Generations do
[4]   newBat ← NEWBAT (bat, m_best)
[5]   if (rand > ri)
[6]    newBat ← LOCALBAT(m_best)
[7]   end if
[8]   newFitness← CALCULATEFITNESS(newBat)
[9]   if (rand <Ai & newFitness < fitness)
[10]   # Accept the new solution.
[11]   bat = newBat
[12]   fitness = newFitness
[13]  end if
[14]  if (fitness < m_best.fitness)
[15]   m_best = bat
[16]  end if
[17] End While
```

Fig.2. Pseudo Code of Variant 2

C. Exchange of Information between Bats

In original bat algorithm, bats use the shared current best x^* in their position updating process but Parallel BA, share generation best bat (g_{best}) across all the parallel map tasks which is used to initialize mapper level best (m_{best}) bat. This m_{best} bat is updated in map () if a better bat is found in map task thus updated m_{best} is used for other bats in the same population split.

In this research, we have explored 2 approaches to generate g_{best} bat:

1. Map Reduce Bat Algorithm (MRBA)

In MRBA, both map and reduce tasks are used. Improved bat as output of all the map tasks are combined and sent to a single Reducer (using same key for all the improved bats). Reduce task sorts all the improved bats and output g_{best} bat which which is used for next iterations as shown in Fig. 3.

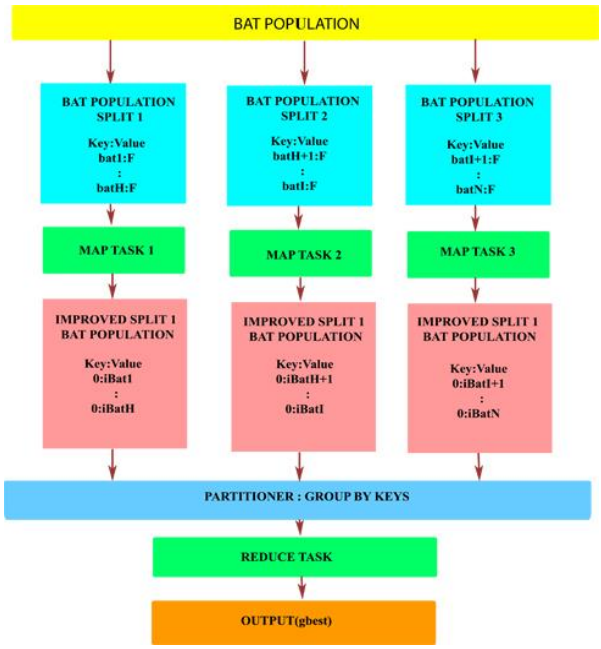

Fig.3. Operation phases in MRBA

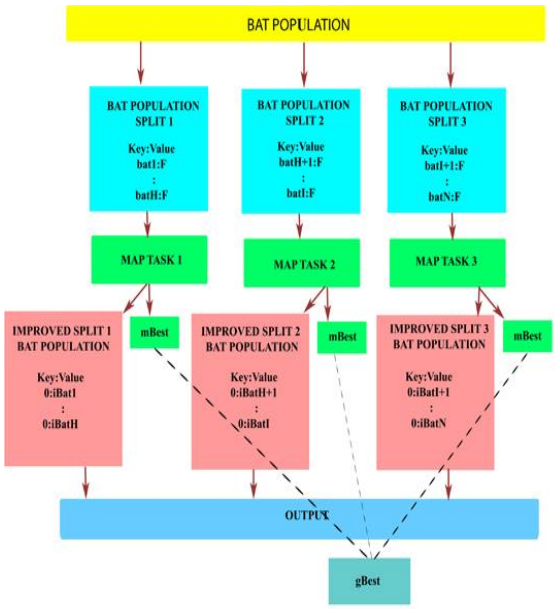

Fig.4. Operation phases in MBA

2. Map Bat Algorithm (MBA)

In MBA, only mapper is used and during the clean-up of each map task this improved m_{best} bat is stored in distributed file system. As there are zero reduce tasks so no mapper output will be sorted/ shuffled thus is directly sent for output. And after every iteration, g_{best} is evaluated from these mapper level bats (m_{best}) as shown in Fig. 4. Thus MBA find the g_{best} with less no. of comparison than MRBA therefore in our proposed algorithm MBA is preferred over MRBA.

VI. Experimental Setup

This section discusses the problem statement on which this proposed algorithm needs to be tested and the environment being set up to emulate the Map Reduce Model.

A. Implementation

These proposed variants are tested on optimization of COCOMO II parameters (a, b) such that calculated efforts approximate to actual efforts for NASA 63 project Dataset.

Formally, this problem can be framed as finding parameter $X = \{x_1, x_2\}$ where, $x_i \in \{0,5\}$ that minimizes the following equation:

$$MRE = [Actual - x_1 (KLOC)^{x_2} \times (EM_1 \times EM_2 \times EM_3 \times \ldots \times EM_{15})] \; / \; Actual \tag{10}$$

$$MMRE_i = \frac{1}{n} \sum_{j=1}^{n} MRE_{ij} \tag{11}$$

Here MMRE is Mean Magnitude of Relative Error which is used as evaluation criteria for assessment of optimized parameters. And since parameter x_1 and x_2 are specific to project modes therefore we execute program for each mode separately.

B. Environment

These variants are implemented on Apache Hadoop (0.19) and ran on 6 node cluster in which one node act as master and other acts as slaves. Each node had Intel 5 dual core, 4GB RAM and 250 GB hard disks. The nodes are integrated with Hadoop Distributed File System (HDFS) giving a potential single image storage space of 2 * 250/3 = 166 GB (since the replication factor of HDFS is set to 3). And execution of no. of mappers and reducer in parallel depends on hardware configuration.

VII. Results

This section discuss results obtained using the developed algorithm and in those experiments, we have used population size of 2 lakhs (input size 13MB), number of iterations as 5, number of nodes as 6 and block size of 5MB.

- Scalability of MBA for effort estimation problem with increasing the no. of Map Tasks (Mapper): Fig. 5 compares the run time of MBA (Variant 1) with different number of map task i.e. setting different block size. On taking block size of 1 MB i.e. 13 map tasks, execution time is 16.84 which decreased with increase in block size due to less no of task distribution among the nodes. But on further decreasing the number of tasks (< no of nodes), run time became almost constant due to decrease in communication overhead.

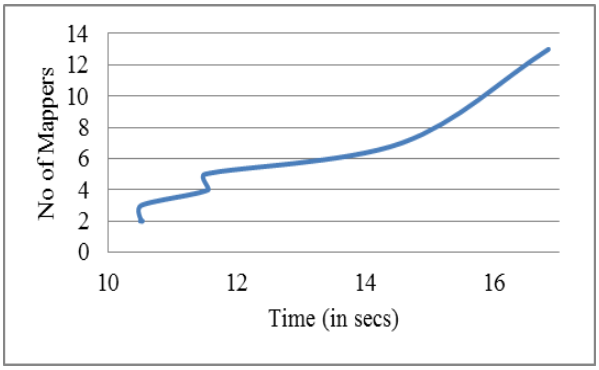

Fig.5. Scalability of MBA for effort estimation problem with increasing the number of mappers

- Comparison between MRBA and MBA: Fig. 6 compares the running time of Map Reduce Bat Algorithm (MRBA) and Map Bat Algorithm (MBA) for variant 1. And as shown, MBA takes less time as compare to MRBA by saving time in intermediate bat's storage with no initialization of Reducer tasks. And also MRBA requires whole population (2 Lakh) sorting to get the g_{best} whereas MBA reads at most 13 m_{best} from the HDFS to find the g_{best}.
- Scalability of MBA for effort estimation problem with increasing the number of nodes: Fig. 7 compares the run time of both MBA based variants by increasing the number of nodes in the cluster. Variant 2 took less time than Variant 1 due to less no. of map cycles and no intermediate generation data handling. And for variant 2, single node cluster took less time than 3 node cluster as there was no communication/task distribution overhead but on further increasing the number of nodes, the execution time reduced and became almost constant.
- Performance tuning with increase in population size: Fig. 8 compares the MMRE of embedded projects for both variants by increasing the size of population. As shown, MMRE reduces with increase in population and became constant after 1500.
- Performance tuning with increase in number of generations: In this experiment, we have taken population size as 10,000 and block size of 5MB and compared the MMRE of embedded projects for both variants by increasing the number of

iterations. As shown in Fig. 9, both variants give better MMRE than COCOMO Model (0.3921) and MMRE decreases with increase in generation due to more refining of output.

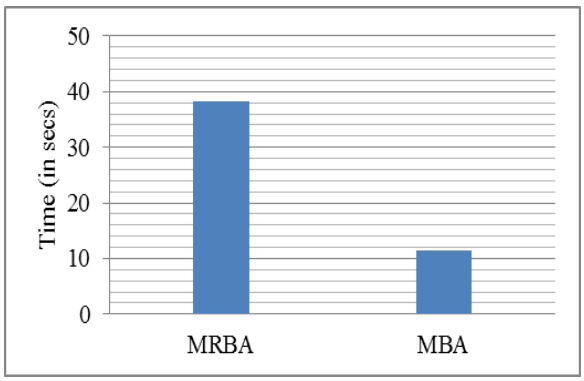

Fig.6. Comparison between MRBA and MBA

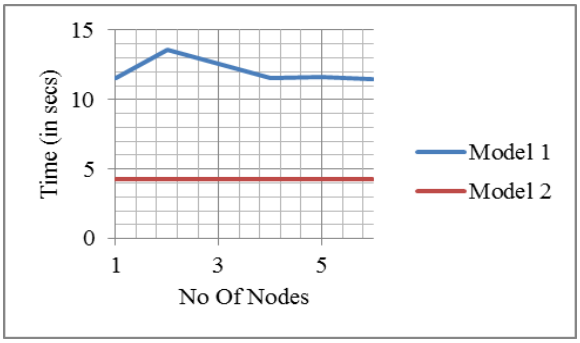

Fig.7. Scalability of MBA for effort estimation problem with increasing the number of nodes

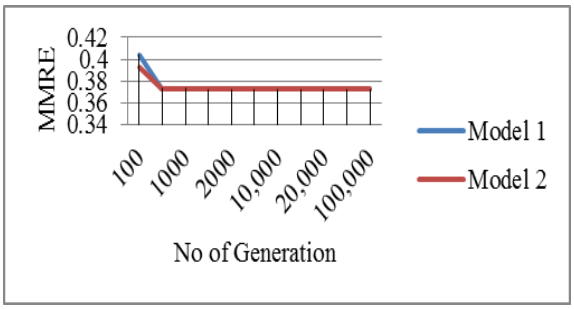

Fig.8. Performance tuning with increase in population size

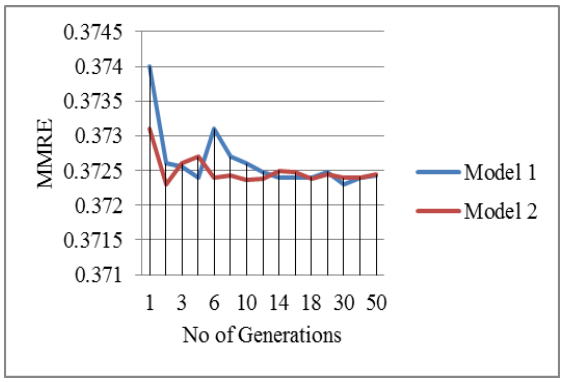

Fig.9. Performance tuning with increase in number of Generations

VIII. Conclusion and Future Work

The proposed variants can easily parallelize BA which than can be used to solve problems involving large search space by simply adding more hardware resources to the cluster and without changing the proposed variant code. And according to experimental results variant 2 shows better convergence than variant 1 and also take less time for execution.

It has been seen that lots of communication, task start up overhead is associated with Hadoop Map Reduce Architecture thus is not suitable for problems involving small search space, few dimension and less computation. And map reduce is suitable for parallel computing problem having huge data with having same processing for each item.

Due to the update process in BA, Parallel BA Variants can't be used for given large dataset in order to find the optimal result from them. So further study on BA modification is required to find best results from given dataset for e.g. getting best quotation from large dataset of quotations.

In future work, both variants should be applied on problems involving large search space, big computation, and large no. of dimensions like in stock market strategies. And these variants running time can also be further improved by examining other features of MapReduce architecture like partitioner, combiner, shuffler etc. which may reduce the processing. We can also compare these variants with existing MPI-based implementation or the results can be compared with other parallel meta-heuristic algorithms.

References

[1] X.-S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," Studies in Computational Intelligence, Springer Berlin, pp. 65-74, 2010.

[2] X. S. Yang, M. Karamanoglu, and S. Fong, "Bat aglorithm for topology optimization in microelectronic applications," presented at the IEEE International Conference on Future Generation Communication Technology (FGCT2012) London, 2012.

[3] N. Gupta and K. Sharma, "Optimizing intermediate COCOMO model using BAT algorithm," presented at the 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India 2015.

[4] Y. A. Gherbi, H. Bouzeboudja, and F. Lakdja, "A Economic dispatch problem using bat algorithm," Leonardo Journal of Sciences, pp. 75–84, June 2014 2014.

[5] J. Dean and S. Ghemawat, "Mapreduce simplified data processing on large clusters," Sixth Symposium on Operating System Design and Implementation, vol. 51, pp. 107-113, 2004.

[6] A. W. McNabb, C. K. Monson, and K. D. Seppi, "Parallel PSO Using MapReduce," presented at the IEEE Congress on Evolutionary Computation, 2007. CEC 2007, Singapore, 2007.

[7] C. Jin, C. Vecchiola, and R. Buyya, "MRPGA An Extension of MapReduce for Parallelizing Genetic Algorithms," presented at the IEEE Fourth International Conference on eScience, 2008., Indianapolis, IN 2008.

[8] C.-Y. Lin, Y.-M. Pai, K.-H. Tsai, C. H.-P. Wen, and L.-C. Wang, "Parallelizing Modified Cuckoo Search on MapReduce Architecture," Journal of Electronic Science and Technology, vol. 11, 2013.
[9] F. Wang, P. L. H. Yu, and D. W. Cheung, "Combining Technical Trading Rules Using Parallel Particle Swarm Optimization based on Hadoop," presented at the International Joint Conference on Neural Networks (IJCNN), Beijing, China, 2014.
[10] D.-W. Huang and J. Lin, "Scaling Populations of a Genetic Algorithm for Job Shop Scheduling Problems using MapReduce," presented at the 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), Indianapolis, IN, 2010.
[11] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," vol. 5931, pp. 674-679, 2009.
[12] L. Di Geronimo, F. Ferrucci, A. Murolo, and F. Sarro, "A Parallel Genetic Algorithm Based on Hadoop MapReduce for the Automatic Generation of JUnit Test Suites " presented at the 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST), Montreal, QC, 2012.
[13] A. Verma, X. Llorà, D. E. Goldberg, and R. H. Campbell, "Scaling Genetic Algorithms Using MapReduce," presented at the ISDA '09. Ninth International Conference on Intelligent Systems Design and Applications, 2009, Pisa 2009.
[14] D. Keco and A. Subasi, "Parallelization of genetic algorithms using Hadoop Map/Reduce," SouthEast Europe Journal of Soft Computing, vol. 1, 2012.
[15] Rabiu O. Isah, Aliyu D. Usman,A. M. S. Tekanyi, "Medical Image Segmentation through Bat-Active Contour Algorithm," I.J. Intelligent Systems and Applications, pp. 30-36, 2017
[16] M.Jaeyalakshmi, Dr.P.Kumar, "Task Scheduling Using Meta-Heuristic Optimization Techniques in Cloud Environment," I.J. Intelligent Systems and Applications, vol. 5, Nov 2016
[17] T Ramakrishnudu, R B V Subramanyam, "Mining Interesting Infrequent Itemsets from Very Large Data based on MapReduce Framework," I.J. Intelligent Systems and Applications, 2015, 07,pp. 44-49

Authors' Profiles

Dr. Kapil Sharma is Associate Professor at the Department of Computer Engineering, Delhi Technological University (formerly Delhi College of Engineering), Delhi, India. He has completed Doctors Degree in Computer Science and Engineering under the Faculty of Engineering and Technology at the M. D. University, Rohtak (Haryana), India. He has obtained his Bachelor of Engineering and Master of Technology Degrees in Computer Science & Engineering and Information Technology.

Sanchi Girotra is Senior Software Engineer at Bureau Veritas, Noida, India. She has completed her Master's Degree from Department of Computer Engineering, Delhi Technological University (formerly Delhi College of Engineering), Delhi, India. She has obtained her Bachelor of Engineering in Computer Science & Engineering at M. D. University, Rohtak (Haryana), India.

How to cite this paper: Kapil Sharma, Sanchi Girotra, "Parallel Bat Algorithm Using MapReduce Model", International Journal of Information Technology and Computer Science(IJITCS), Vol.9, No.11, pp.72-78, 2017. DOI: 10.5815/ijitcs.2017.11.08