Annals of Emerging Technologies in Computing (AETiC)
Vol. 4, No. 1, 2020

Research Article

Performance of Parallel Distributed Bat Algorithm using MPI on a PC Cluster

Fazal Noor*, Abdulghani Ibrahim and Mohammed M. AlKhattab
Islamic University of Al-Madinah, Saudi Arabia
mfnoor@gmail.com; a.a.ghanii@gmail.com; mostafa.kht9@gmail.com
*Correspondence: mfnoor@gmail.com

Received: 22nd November 2019; Accepted: 12th December 2019; Published: 1st January 2020

Abstract: Optimization algorithms are often used to obtain optimal solutions to complex nonlinear problems and appear in many areas such as control, communication, computation, and others. Bat algorithm is a heuristic optimization algorithm and efficient in obtaining approximate best solutions to non-linear problems. In many situations complex problems involve large amount of computations that may require simulations to run for days or weeks or even years for an algorithm to converge to a solution. In this research, a Parallel Distributed Bat Algorithm (PDBA) is formulated using Message Passing Interface (MPI) in C language code for a PC Cluster. The time complexity of PDBA is determined and presented. The performance in terms of speed-up, efficiency, elapsed time, and number of times fitness function is executed is also presented.

Keywords: ***Bat Algorithm, Computational Complexity, Distributed, Message Passing Interface (MPI), Optimization Algorithm, Parallel, PC Cluster, Neural Networks.***

1. Introduction

Many real-world optimization problems in the areas of controls, chemistry, biology, engineering, and may other fields require large amounts of computations and simulations. The problem may be formulated as a continuous function and nature inspired optimization methods utilized to obtain optimal solutions. Bat algorithm (BA) is a heuristic optimization algorithm and has been reported to be efficient in providing optimal solutions to continuous nonlinear constrained problems. In cases where the search space is extremely large a PC cluster is useful for large computations [1]. A PC cluster consists of off the shelf machines connected to a fast Ethernet switch. A PC cluster can be thought of as an affordable supercomputer. Super computers are used in many parallel computing applications to solve very complex problems [2].

In this research, our main aim is to develop a parallel bat algorithm for optimization problems and use a PC cluster for parallel distributive computations [3, 5, 6, 7]. RedHat Linux is used as the operating system with Local Area Multicomputer (LAM) software for building the cluster. Parallel Bat algorithm using Message Passing Interface is implemented in C language and its performance is studied on a PC Cluster.

The BA and its variants have been employed in solving variety of real-world optimization problems, the reason is faster convergence, fewer parameter adjustments, and efficient

Fazal Noor, Abdulghani Ibrahim and Mohammed M. AlKhattab, "Performance of Parallel Distributed Bat Algorithm using MPI on a PC Cluster", *Annals of Emerging Technologies in Computing (AETiC)*, Print ISSN: 2516-0281, Online ISSN: 2516-029X, pp. 19-27, Vol. 4, No. 1, 1st January 2020, Published by International Association of Educators and Researchers (IAER), DOI: 10.33166/AETiC.2020.01.003, Available: http://aetic.theiaer.org/archive/v4/v4n1/p3.html.

implementation [4]. Bat Algorithm has been used in various optimization problems. Bat Algorithm (BA) is used to find optimum capacitor size to reduce transmission and distribution losses in bus system [4]. BA has also used in Wireless Sensor Networks for distributed iterative node localization. BA has also used to resolve combined economic and emission dispatch problem.

2. PC Clusters

In the literature there are many types of clusters, however a Beowulf cluster is an affordable alternative. It consists of multiple PCs of N nodes connected with a Fast Ethernet switch [2]. The main reason to use a PC cluster is for performance. Multiple computers connected together and working on multiple tasks or problems is expected to finish the tasks faster than using a single computer. PC cluster is useful for real-time constraints such a task with computations finish in a certain time period. For example, weather forecasting has to be performed in real-time. PC cluster provides throughput, it provides computing power much more than a single processor. An example is Beowulf Linux cluster used by Google, in which 15,000 PCs are used to provide high performance Web search service [3].

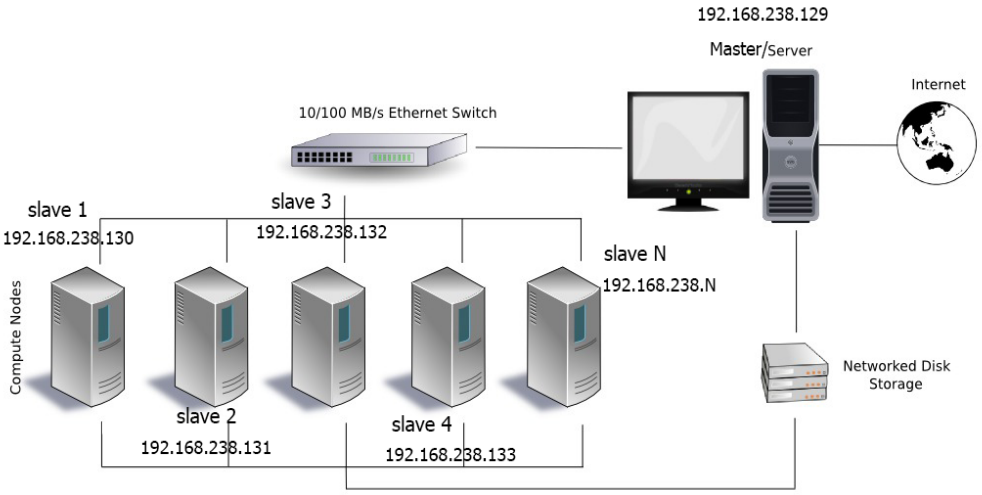

Figure 1. PC Cluster with MPI.

PC cluster provides memory for applications requiring huge amounts of data for example in terabytes. PC clusters provide computational power such as for parallel programming. Breaking up a problem into parallel tasks and submitting them to a PC cluster provides a solution in a fraction of a time. For example, solving a system of M linear equations or matrix vector multiplication or image processing, etc. Tasks that may be easily divided in small tasks and executed independently are called in the literature as embarrassingly parallel [7]. The metric used to measure performance is called a Speedup Factor and is a measure or indicator of relative performance given as

$$S(p) = \frac{\text{Execution time using single processor system (with best sequential algorithm)}}{\text{Execution time using multiple processor (with p processors)}} \quad (1)$$

$$S(p) = \frac{T_s}{T_p} = \frac{T_s}{(T_{computation} + T_{communication})} \quad (2)$$

Ts: Execution time with the best sequential algorithm running on single processor, Tp : Execution time for solving the same problem on a PC cluster. Note: Tp is the total time which consists of computation time (Tcomp) and time spent communicating (Tcomm) among nodes in a PC cluster. Efficiency: useful to known how long processors are being used on the computation. Efficiency is defined as

$$E = \frac{Execution\ time\ using\ single\ processor}{Execution\ time\ using\ \ multiple\ processor\ \ x\ number\ of\ processors} \quad (3)$$

$$= T_s / (T_p \times p) \quad (4)$$

which leads to:

$$E = \frac{S(p)}{p} \text{ x } 100\ \% \quad (5)$$

3. Optimization Method – Sequential Bat Algorithm

In 2010 Xin-She Yang, proposed a meta-heuristic optimization algorithm based on Bat's echolocation behavior and named it Bat algorithm [1,4]. The bats emit sound waves and based on the reflected echoes of the sound waves; the bats can discern its prey from other objects. The bats biological system is so sophisticated that it is able to even compute the prey's size and location. The bat emits loud sound waves when it is in search phase and then decreases the loudness as it approaches its prey and at same time increases the pulse emission rate. The bat algorithm uses dynamic strategy to perform global search (exploration) and local search (exploitation) to get a better solution. Yang developed the bat algorithm with the following 3 rules:

1. Echolocation is used by the bats to compute distance and be able to discern between food/prey and background objects.
2. Each bat is flying randomly with velocity at position x and with a frequency f. The frequency may have varying wavelength lambda, and loudness A0 to search for prey.
3. Assumption that loudness changes from a large positive value for A0 to a minimum constant value Amin.

BA approach is based on the hunting behavior of bats. Bats use echolocation to detect prey, avoid obstacles and locate their resting location. The bats emit high-pitched sounds and interpret their echoes to determine the distance and direction of targets. The Bat algorithm has the following main features, automatic zooming via loudness and pulse emission rates, parameter control, frequency tuning. It has the following advantages, ability to solve efficiently wide range of nonlinear optimization problems with optimal solutions. The Bat algorithm has proven to provide solutions in a variety of applications, namely, Engineering design, Protein Structure prediction, Classification of genes, PID controllers, and Neural Networks.

3.1 Sequential Bat Algorithm

Step 1: Initialization of Bat Population

Generate population randomly of N bats (possible solutions) each with dimension d and initial values for frequency, velocity, pulse emission rate and loudness,

$$x_{ij} = x_{\min j} + rand(0,1)(x_{\max j} - x_{\min j}) \quad (6)$$

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

Step 2: Update Frequency, Velocity and Solution

Each bat (solution) emits sound pulses of random frequency having value between fmax and fmin, and controls the velocity and provides a new position. Use the following equations for frequency, velocity, and position to update the bat's position.

$$f_i = f_{min} + (f_{max} - f_{min})\beta \quad (7)$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x^*)f_i \quad (8)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad (9)$$

Where β is a random number 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.

$$\beta f_{min} = 0,\ f_{max} = 100 \quad (10)$$

- x* is the current global best location
- t is number of iteration

Step 3: Local Search

Use the following equation to compute a random walk in the vicinity of the best solution.

$$x_{new} = x_{old} + \varepsilon A^t \quad (11)$$

Note, x_{old} is the best solution chosen from among the best solutions, ε is a random number in the interval [-1, 1], while A^t represents an average loudness of all the bats.

Step 4: Updating Loudness and Pulse Emission Rate

As the bat approach prey, they decrease the loudness of their emitted sound pulse and increase their pulse emission rate, these 2 are updated according to the following equations,

$$A_i^{t+1} = aA_i^t \quad (12)$$

$$r_i^{t+1} = A_i^0(1 - e^{-\gamma t}) \quad (13)$$

Where α and γ have constant values, r_i^0 and A_i^0 have random values in the following range, $r_i^0 \in [0, 1]$ and $A_i^0 \in [1, 2]$.

Step 5: Find the current best solution.

Compare the current best bat with the last best fitness value, if it is better then perform an update of best solution.

3.2 Parallel Models - Bat Algorithm

The parallel bat algorithm is based on the inherent behaviour of each bat echolocation. Every bat is flying independently with its own frequency, velocity, and location. Therefore, a straight forward parallel method would be to have number of processors equal to the number of flying bats. Each processor executes in a parallel and distributed fashion. However, when there are limited number of processors, then a group of X bats or solutions will be assigned to each processor. In a parallel distributed algorithm, each worker node works on a split population (i.e. solutions) depending on the size of the PC cluster. That is if a PC cluster consisted of n=16 nodes and the population size was 160, then each node works on population size of 160/16 = 10 solutions. The worker nodes each performs the same sequential steps, but now with reduced population size. Each worker after a specified number of iterations, sends the current best of its group to the Master node. The master node after receiving from all the workers their best, it then compares them and sends back to each worker, the best of all best_k. The parallel-distributed bat algorithm using the Master-Worker model is summarized as follows:

3.3 Parallel Models: Master-Slave Model

Parallel-Distributed Bat Algorithm – Master – Worker (PDBA-MW)

Initialize every_send_time to 50

For (iteration = 1 to Max_iteration)

IF (Worker Node) then

For (i=1 to newpopsize) **Note: newpopsize is N / total_workers**

1. Perform Steps 1 to 5 of sequential algorithm

End

If (iterations equals every_send_time) then

a. Each worker_k sends its best candidate solution and corresponding fitness to the Master.

b. Receive the best solution and fitness from Master node.

Else if (Master Node) then

For (k = 1 to Total_Workers)

1. Receive best_k from each worker.
2. Compare and choose best among all the best received.
3. Send to each worker the best.

ENDIF

The process is terminated when desired accuracy is achieved or maximum number of iterations has been reached.

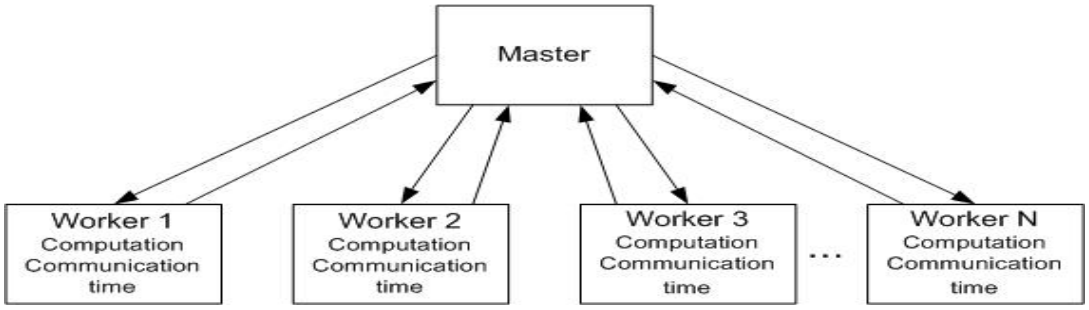

Figure 2. Parallel Bat Algorithm on PC Cluster with MPI.

In the PDBA the master may be used to perform primarily the exploration part and workers primarily perform the exploitation part more than exploration of the search space. Other scenarios are possible. The main objective in the PD method is to reduce the communication time as much as possible compared to the computation time.

3.4 Parallel Models: Group Model

In this model, after a fixed number of iterations, each node sends or broadcasts its best solution to the group nodes. This means if there are N nodes, then there will be N broadcasts in total. Each node stores the received best and performs comparison. If received solution is better than the node's best, it is updated otherwise not. In the Bat algorithm the steps that can be parallelized (executed independently are bats updating its position) and iterations from generation to generation can not be parallelized since the improvement in population of bats depend on earlier update of the solutions.

In the proposed PDBA, the MPI send and receive communication routines are used. The Send and Receive routines are Blocking, i.e. is do not return until transfer is completed. The syntax of send is MPI_Send(buf, count, datatype, dest, tag, comm), where buf is address of send buffer, count is number of items to send, datatype is datatype of each item, dest is rank of destination process, tag is message tag, and comm is the communicator. The syntax of receive is MPI_Recv(buf, count, datatype, src, tag, comm, status), where parameters are similar to send, src is rank of source process, and status is status of operation.

Time Complexity Analysis of Bat Algorithm

In this section, the time complexity of the serial Bat algorithm and the PDBA. Since the time complexity of a Heuristic algorithm depends on the number of iterations. Here time complexity of Bat algorithm time complexity is first derived based on time taken to perform floating point multiplication, and addition. The following Theorem is used for determining complexity.

Theorem: $f(x) = O(g(x))$ if and only if there exists positive constants, c and x_0, such that $0 < f(x) < cg(x)$ for all $x \geq x_0$ where $f(x)$ and $g(x)$ are functions of x. The computational complexity of the BA algorithm is determined to be as follows,

$$T_{BA} = O\big(3N(d+1)\big) + O(N(6d+1) + O(N\ x\ fitness\ function) \quad (14a)$$

When considering a multiplication or an adds as an operation, then equation (14a) can be rewritten as,

$$T_{BA} = O(Nd) + O(N\ x\ fitness\ func\ complexity) \quad (14b)$$

$$T_{total} = G_{generations} x\, T_{BA} \quad (15)$$

where N is the population size, d is the number of dimensions, and G is the number of generations.

The time complexity of PDBA is similar to the serial or sequential Bat algorithm with N replaced with p=N/M where N is the population size and M is the number of worker nodes, in addition PDBA has T_{comm} (communication time). The communication time complexity of parallel algorithm is T_{comm} is defined to be $t_{startup}$ + n x data, where n is the size of data. Start-up time is assumed to be constant and data time is transmission time to send one data and n data words to be transmitted. The total time complexity of the PDBA is the sum of complexity of computation and communication times.

$$T_{PDBA} = O(3p(d+1) + O\big(p(6d+1)\big) + O(p\ x\ fitness\ function) + T_{comm} \quad (16a)$$

For large p or d, then eq. (16a) can be written as follows,

$$T_{PDBA} = O(pd) + O(p\ x\ fitness\ function) + T_{comm} \quad (16b)$$

Usually, T_{comm} is small and can be ignored. The theoretical Speed up (S) of PDBA is then computed as follows,

$$S_{PDBA} = T_{BA}/T_{PDBA} \quad (17)$$

4. Results

In this section, in order to determine how well the PDBA optimization algorithm works several benchmark functions are used as a check. The PDBA simulations are run on a PC cluster consisting of a Master node and M worker nodes. The population size N is chosen to be 160. Each node works on pop = N/M with M consisting of 1, 2, 4, 8, and 16 nodes, respectively. The parameters used for the simulation are as follows; Maximum number of generations is 10000, population size is 160 for 1 node, 80 for 2 nodes, 40 for 4 nodes, 20 for 8 nodes, and 10 for 16 nodes. All the benchmark functions have known global optimum values. The PDBA is terminated when either maximum number of generations are reached or the following condition is satisfied

$$\left|F_{actual\ min} - F_{computed\ min}\right| < 10^{-4} \quad (18)$$

Three performance measurements are speed-up, efficiency, and elapsed time are used to evaluate the performance of the PDBA algorithm. Figure 3 and Figure 4 shows the speedup and efficiency, respectively, for 3 different population sizes, double (160), original (80), and half size (40). Figure 5, shows time taken for the 5 different PC cluster sizes, the time decreases with increase in PC cluster size. Figures 5 and 6 show the number of times a fitness function is called and maximum, minimum, and average values are plotted. To check the performance of PDBA, the following benchmark functions out of many available were tested. A benchmark function named F1: Schaffer 2 is a global optimization problem, having the following features, being continuous, not convex, unimodal, differentiable, and non-separable. This is a unimodal minimization problem defined as follows:

$$f_{Shaffer}(\mathbf{x}) = 0.5 + \frac{sin^2(x_1^2 - x_2^2)^2 - 0.5}{1 + 0.001(x_1^2 + x_2^2)^2} \quad (19)$$

Here, n represents the number of dimensions and $x_i \in [-100, 100]$ for $i = 1, 2$. For 2 dimensions f(x_i) = 0, for x_i = 0 and for i=1, 2.

Another benchmark function F2 is the Rosenbrock global optimization problem. This is a minimization problem defined as follows:

$$f_{Rosenbrock}(\mathbf{x}) = \sum_{i=1}^{n-1}[100\ (x_i^2 - x_{i+1})^2 + (x_i - 1)^2] \quad (20)$$

Here n represents the number of dimension and for i=1,, n. Global optimum occurs at f(xi) =0 for x_i = 1 and for i=1,2,….,n. The features of the Rosenbrock function are that it is continuous, convex, multimodal, differentiable, and non-separable.

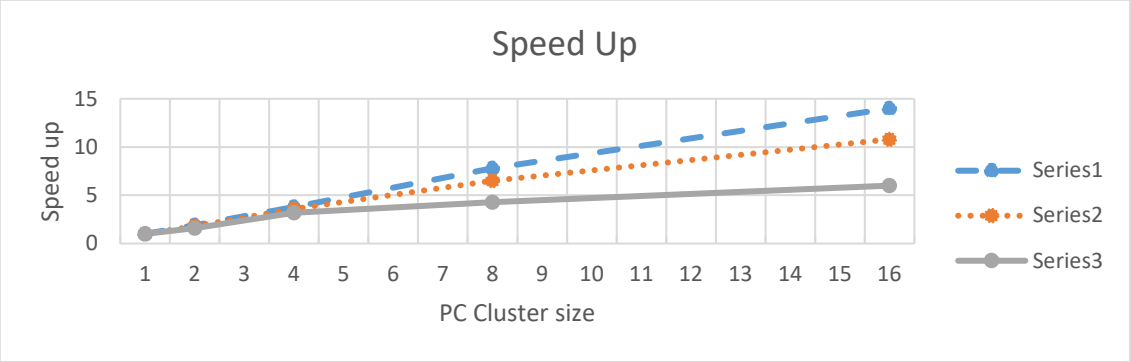

Figure 3. Speedup for Cluster size for original population size of 80, half (40), and double (160).

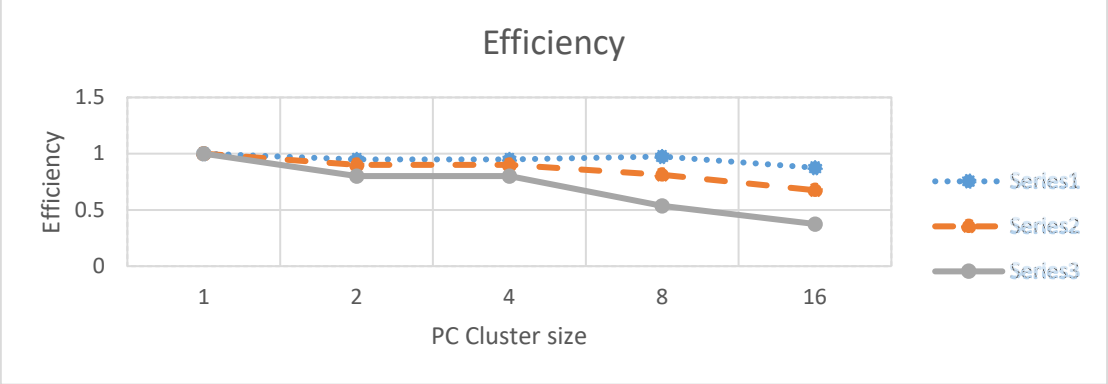

Figure 4. Efficiency for PC Cluster size for original population size of 80, half (40), and double (160).

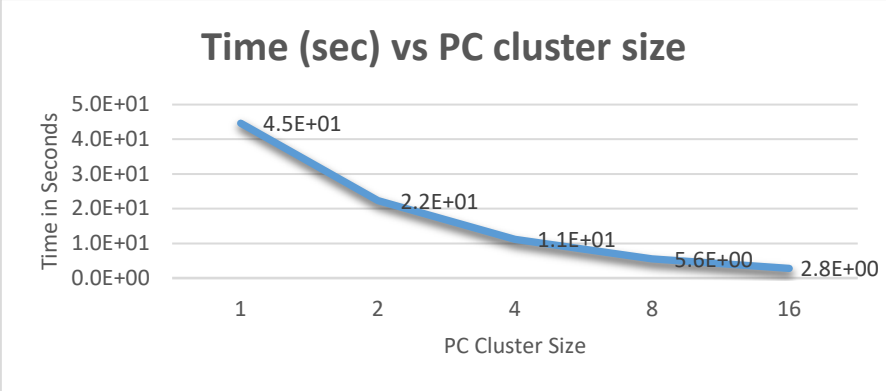

Figure 5. Elapsed time in seconds versus PC cluster size.

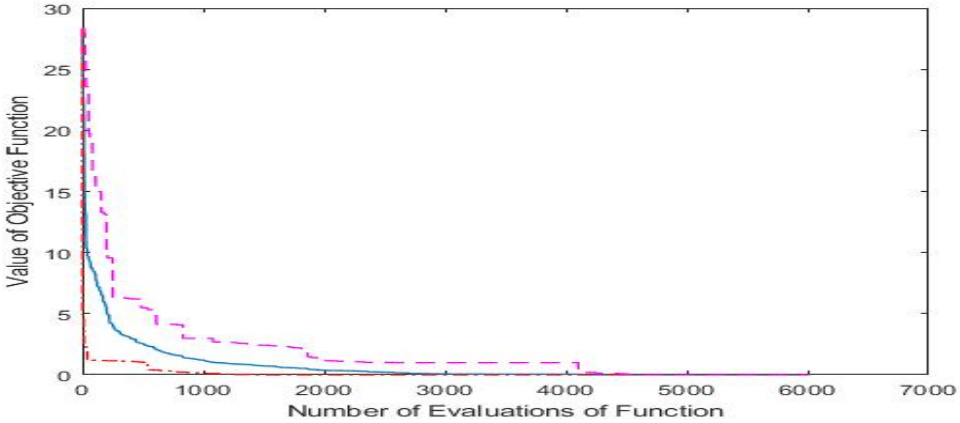

Figure 6. Number of times the fitness function is evaluated for benchmark function F1.

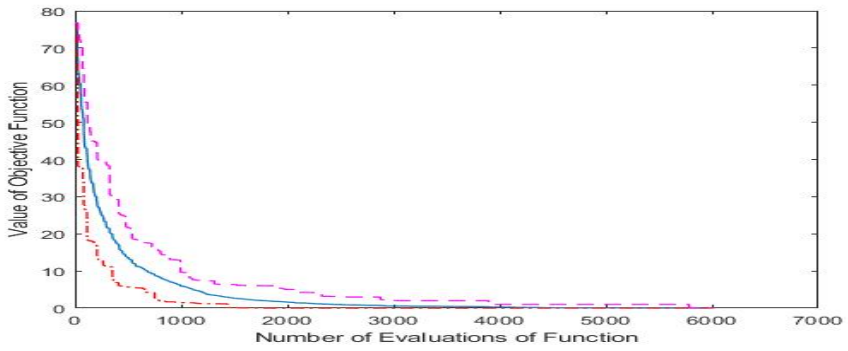

Figure 7. Number of times the fitness function is evaluated for benchmark function F2.

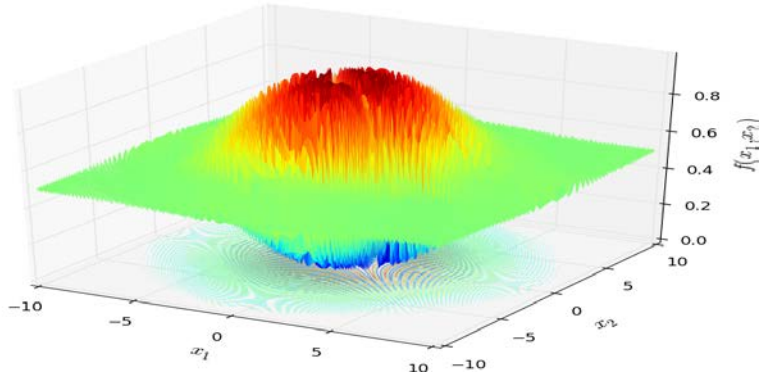

Figure 8. Two-dimensional Schaffer 2 function

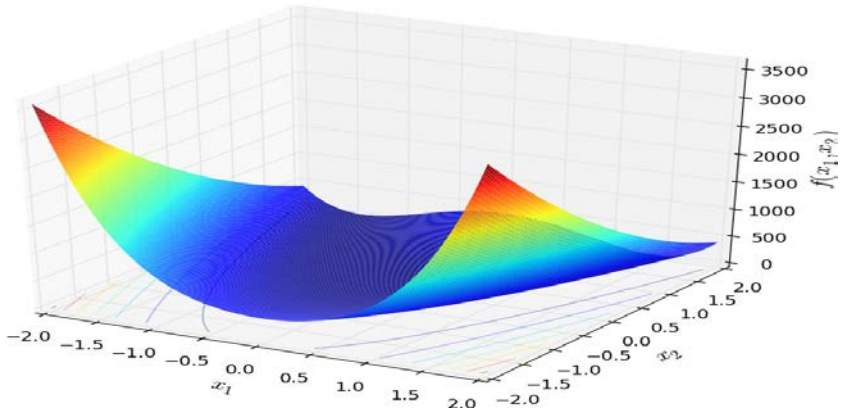

Figure 9. Two-dimensional Rosenbrock function

5. Concluding Discussions

A parallel distributed Bat algorithm is formulated for continuous constrained optimization problems. The results show the efficacy of the algorithm for optimization problems with large population size and vast search space. The Bat's algorithm has several parameters which can affect the convergence rate if they are fine tuned. New solutions are formulated by computing the frequencies, loudness and pulse emission rates. The new generated solutions are accepted or not depends on several factors such as loudness, pulse rate, and fitness of the solution to the global optimum. It is seen the parallel distributed version using MPI, namely the PDBA, the convergence rate and speed up increases as PC cluster size is increased. The time computational complexity for BA and PDBA was derived and speed up in terms of computational and communication presented.

The Schaffeur2 and Rosenbrock are standard benchmark functions were used to evaluate the characteristics of PDBA such as convergence rate, precision, and robustness. It was seen the convergence to the minimum took longer in the Rosenbrock function case compared to the Schafeur2 function. The common features of these two functions are both are continuous, differentiable, and non-separable, whereas Schaffer2 is not convex and unimodal and Rosenbrock is convex, and multimodal.

In real-world applications involving thousands or more design variables are very challenging and in the future the work should be directed towards using the PDBA with multi-objective constraints. Another direction of future research is the use of PDBA in obtaining optimum weights of Neural Networks.

Acknowledgement

This research was supported by the Takamul Grant #25 from the Deanship of Research at the Islamic University of AlMadinah. The authors are also thankful to the support provided by the Faculty of Computer and Information Systems.

References

[1] X.-S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," Studies in Computational Intelligence, Springer Berlin, pp. 65-74, 2010. https://doi.org/10.1007/978-3-642-12538-6_6.

[2] Syed Misbahuddin and Fazal Noor, "Hands-on Workshop on Parallel Processing", University of Hail, Saudi Arabia,: Syed Misbahuddin, 2007.

[3] 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, pp. 780–785, 2010. DOI 10.1109/CloudCom.2010.18

[4] Y. Xin-She "Bat algorithm for multi-objective optimisation," Internal Journal of BioInspired Computation, vol. 3, pp. 267-274, 2011. Doi 10.1504/IJBIC.2011.042259.

[5] 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, pp.3987-3994. 2014. DOI: 10.1109/IJCNN.2014.6889599.

[6] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," vol. 5931, pp. 674-679, 2009.

[7] Barry Wilkinson and Michael Allen, Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, North Carolina at Charlotte: Prentice Hall, 2005.

[8] K. Khan, A. Nikov, and A. Sahai, "A Fuzzy Bat Clustering Method for Ergonomic Screening of Office Workplaces," in Third International Conference on Software, Services and Semantic Technologies S3T 2011. vol. 101, ed: Springer Berlin Heidelberg, 2011, pp. 59-66. https://doi.org/10.1007/978-3-642-23163-6_9

[9] Eduardo R. Hruschkaaand Nelson F.F. EbeckenbCOPPE, "researchgate," 27 May 2002. [Online] Available: https:// www.researchgate.net /publication/ 220571471_A_ genetic_ algorithm _for _cluster _analysis. [Accessed 3 Mar 2019].

[10] Chandra R., L. Dagum D. Kohr. D. Maydan, J. McDonald and R. Menon, Parallel Programming in OpenMP, San Francisco, CA: Margon Kaufmann Publishers, 2001.