Bi-Objective Task Scheduling in Cloud Computing using Chaotic Bat Algorithm

Fereshteh Ershad Farkar
Department of Computer Engineering,
Tabriz Branch, Islamic Azad University,
Tabriz, Iran

Ali Asghar Pourhaji Kazem*
Department of Computer Engineering
Tabriz Branch, Islamic Azad University,
Tabriz, Iran

***Abstract*—Cloud computing is a technology for providing services over the Internet. It gives approach to renting IT infrastructures on a short-term pay- per-usage basis. One of the service provider's goals is to use the resources efficiently and gain maximum profit. Cloud processes a huge amount of data, so tasks scheduling is a vital role in the cloud computing. The purpose of this paper is to propose a method based on chaos theory and bat algorithm for task scheduling in Cloud computing. Task scheduling is a core and challenging issue in cloud computing. The nature of the scheduling issue is as an NP-Hard problem and because of the success of heuristic algorithms in optimization and NP-Hard problems, the authors use a newly inspired bat algorithm and chaos theory to scheduling the tasks in the cloud. In this method, bat or candidate solutions are represented by a one-dimensional array. The fitness function is calculated based on makespan and energy consumption. The results show that the proposed method can schedule the received tasks in proper time than other compared heuristic algorithms, also the proposed method has better performance in term of makespan and energy consumption than compared methods.**

Keywords*—Cloud computing; scheduling; chaos theory; bat algorithm

I. Introduction

Cloud computing (a recent computing model) comes from distributed computing, parallel computing and grid computing. The dynamism and heterogeneity are properties of cloud computing resources. In cloud computing, resources such as storage, memory, processors, and applications are provision as services. Cloud computing environment is a commercial platform. Currently, there exist many commercial clouds, such as Amazon, which provide virtualized computational and storage hardware. Virtual Machine (VM) is a critical component of software stacks in the cloud computing, for example, Amazon Elastic Computing Cloud (Amazon EC2) [1] is a cloud platform that provides infrastructure as service in the form of VMs. The cloud computing greatly decreases the financial cost of acquiring hardware and software for application deployment, as well as maintenance costs [2]. So, how to use efficiently and effectively cloud computing resources becomes more important. Cloud computing provides a pool of resources in a self-service, dynamically scalable and metered method with guaranteed quality of service to users. To achieve guaranteed Quality of Service (QoS) to users, that is important the tasks be assigned efficiently to defined resources by providing multiple VMs for executing the tasks included in a program. Cloud computing also offers pay-per-use metered service. There are motivational research results for efficient task scheduling in cloud computing, but task scheduling problems are still considering as an NP-complete issue. There are some objective functions and optimization criteria while tasks scheduling in the cloud environment, such as makespan, cost, flow time, tardiness, waiting time, turnaround time [3], and energy consumption [4]. In our proposed work, we propose QoS-based task scheduling algorithm called the Chaotic Bat Algorithm for task scheduling on cloud computing, which aims to create a schedule to minimize the total makespan and energy consumption of tasks executions. Bat algorithm, first proposed by Xing-She Yang [5], is a new meta-heuristic algorithm inspired by the echolocation of micro-bats to sense distances while detecting their prey. Micro-bats using this technique can find their prey and recognizes prey even in complete darkness Echolocation is the main specification of bats behavior. This means that the bat gives out sound pulses and listens to echoes to find preys and avoid collisions obstacles while flying. The Bat algorithm can have superiority performance than optimization algorithms and can solve many problems, including real world and practical engineering optimization problems [6]. So one aim of this paper is to introduce chaos into the standard bat algorithm.

The rest of the paper is classified as follows: Section 2 is talking about related work for scheduling in cloud computing; Section 3 includes the background, classical Bat Algorithm and Chaos Theory; Section 4 describes the problem; Section 5 discusses the main idea and how the new Bat Algorithm and Chaos Theory are integrated; Section 6 contains the simulations and results obtained; and Section 7 tells about the future scope and conclusion of this paper.

II. Related Works

Task scheduling is a critical issue in cloud computing, so a lot of researches have been done in this scope. This problem is of a known Np-Hard type issue [2], [4], [7]-[9]. They belong Np-Hard including thousands of different issues with many applications. So far, no solution has been found for these issues in a reasonable time, and may not be found in the future at all. These also prove that there is no quick solution for them. If a solution found only for any of these issues, this solution would solve the most parts of such this issues. Solution based techniques on full search are not feasible for this kind of problems. The cost of schedules is very high. Metaheuristic-based techniques can overcome these problems

*Corresponding author

by providing near optimal solutions in reasonable time. For example, ACO algorithm is useful for solving separate optimization problems which requiring one path to reach a goal. It has been successfully for solving multidimensional knapsack problem, traveling salesman problem, job shop scheduling, and quadratic assignment problem, task scheduling in grid and cloud computing, and much more [3]. In [10], they have considered minimization of makespan as the objective function. Their objective function based on execution time and transfer time of tasks on VMs. The algorithm simulated with the number of tasks changing from 100 to 1000 in the cloudsim simulation environment. ACO has been compared with RR and FCFS algorithms and experimental results show that when the number of tasks increased, ACO execution gets a short time compared with RR and FCFS. For 1000 tasks, the algorithm able to decrease makespan in comparison with RR and FCFS. Recently, the genetic algorithm is useful metaheuristic for taking high-quality solutions for combinatorial optimization problems, including the task scheduling problem [11]. Another competency of genetics is that its inherent parallelism can activate to reduce running time [12].

III. BACKGRPOUND

A. Bat Algorithm

The bat algorithm has been a recently proposed metaheuristic algorithm by Xin-She Yang [5], based on the echolocation of micro bats. In the real world, echolocation can diffuse within only a few thousandths of a second with a changing frequency. Micro bats use echolocation to search preys. All micro bats are insectivores and they use a type of sonar, called echolocation, to find preys and avoid obstacles. Now, we remind the standard bat algorithm according to the following rules:

All bats use echolocation to sense distance and find prey, also they know the difference of food and obstacles with some magical method.

Bats fly randomly with velocity V_i at location X_i with a permanent frequency Fmin, Fmax changing loudness A_0 and wave length λ to search prey. They can intelligently tune the wavelength (or frequency) of their emitted pulses and tune the rate of pulse emission r ∈ [0, 1], relevant on the proximity of their aim.

Although the loudness can change in many ways, we consider that the loudness changes from positive A_0 to a minimum fixed value A_{min}.

Initialize solutions: the virtual bats (solutions) have the positions X_i, and velocities V_i in a D-dimensional search space. They are randomly distributed in the possible search space.

Generate new solutions: the values of the frequency F_{min} and F_{max} is dependent on the dimensions of the issue. The positions and velocities of bats in every temporal interval are defined as follows:

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

$$v_i^{t+1} = v_i^t + (x_i^t - x_*) \times \beta , \quad (2)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (3)$$

Where $\beta \in [0,1]$ is a random vector and X_* is the current global best solution.

Local search: create a random number. If it is bigger than the i^{th} bat pulse emission rate (r_i), then create the new bat. A new solution is generated around the current best solution

$$\text{If rand} > r_i \quad (4)$$

$$X_{new} = X_{old} + \varepsilon * A_{mean}^t .$$

Where $\varepsilon \in [-1, 1]$ is a random vector and A_t is the average loudness of all bats at time step t. r_i is the i^{th} solution pulse emotion rate.

Updating solutions by flying randomly: loudness is reduced and pulse emission rate is increased by using the equations as follows:

$$\text{If rand } () < A_i \ \&\& \ f(x_i) < f(x_*),$$

$$F(x) = f(x_i), \quad (5)$$

$$A_i^{t+1} = \alpha A_i^t$$

$$r_i^{t+1} = r_i (1 - \exp(-\gamma t))$$

Where α and $\gamma \in [0, 1]$, A_i is the i^{th} bat loudness (x_i) is the fitness value of i^{th} bat and $f(x_*)$ is fitness value of the best bat.Update the current global best solution by described the formulas until reached the termination condition.

B. Chaos Theory

In this world numbers are essential for seeing most natural phenomena, our surrounding world isn't a static system and this system change with the dynamism of time. When system change, the numbers represent system state in a temporal step. The dynamic systems have not a lawful period for representing the system states with numbers. The system can change in discrete time. For example, all animals and most of the insects have a one-year life cycle, and study of them requires that only we look on their life system once in a year. These systems are known as non-linear systems. In such systems, the system output is not proportional to the input. If variables changes in an initial time result may change of the same or another variable time. Therefore, the system changes are not proportional with the systems input. The non-linear systems could not be divided into smaller sections and be solved separately. They possess complete complexities. In non-linear sciences, non-linear dynamic systems are studied. Nature is a non-linear system. The non-linear systems are employed in studying various fields, such as Mathematics, Biology, Physics, Chemistry, and Computer sciences. Furthermore, the chaotic systems are very sensitive to initial values, and a small change in the initial values will have great changes in the output. The changes of dynamic systems in the discrete time called a map. In another discussion, the convergence of evolutional algorithms is mainly dependent on the initialization of its parameters. When the random parameters are used for initialization, different results are seen in various executions of this algorithm. For this reason, the

*Corresponding author

random variables are a key which may lead the algorithm to escape from local optimum to better results. Some chaotic maps are well known and we can use them in the algorithm parameters initialization, such as Gauss Map, Tent map, Circle map, Iterative map, logistic map, Sinusoidal map.

IV. Problem Definition

A. System Model

Cloud computing environment is used virtualization to map the resources to the virtual machines. The tasks classify according to QoS requirement, such as bandwidth, cost, resource distance, credibility Finally, schedule the tasks to physical resources. This paper intends that when virtual resources meet tasks QoS requirement, using makespan and energy-aware algorithm schedule tasks to physical resources.

B. Resources and Tasks

$R= \{R_1, R_2 \dots R_n\}$ represents a set of resources, where R_i is the ith resource. Each resource implemented with a Dynamic Voltage scaling module [4]. If supply voltage and frequency decrease, it causes to reduce the energy consumption consumed by the resource. Resource R_i represented as R_i {rcc, svs} where rcc is the resource computing capacity parameter of R_i, svs is the supply voltage strategy of ri. In svs, there is a relational vector between its supply power and frequency. That is $V_i=[\{vs_1(i),fs_1(i)\};\{vs_2(i), fs_2(i)\};\dots;\{vs_l(max)(i),fs_l(max)(i)\}]$, where $vs_l(i)$ supplies power of resource R_i at DVS level s_l, $fs_l(i)$ is the relative frequency coefficient within the range of [0,1]. In this paper, we represent 3 power supply strategies (voltage and relative frequency pairs), and 16 DVS levels which is shown in Table 1.

$T= \{T_1, T_2 \dots T_n\}$ represents a set of independent tasks, where T_j is the jth task, $W= \{w_j, 1<j<n\}$ represents set of task computational workload and EXT= EXT [i, j] m×n is the matrix of task execution times in each resource. EXT [i, j] denotes an expected time for the execution of task T_i on resource R_j.

C. Energy Consumption

According to task computational workload and resource computing capacity, the execution time needed for executing task T_i on resource R_j defined as:

$$EXT [i, j] = W (t_i)/rcc (r_j) \quad (6)$$

Supply power and frequency decrease while tasks execution time increase, when task T_i execute on resource R_j at DVS level si, EXT matrix can be defined as follows:

$$EXT'[i,j] = [\frac{1}{fs1(i)}.EXT[i,j], \frac{1}{fs2(i)}.EXT[i,j], \dots, \frac{1}{fs1\,max(i)}.EXT[i,j] \quad (7)$$

Where EXT [i, j] can be calculated according to the Equation (7), $\{fs_1(i), fs_2(i), \dots, fs_l(max)(i)\}$ denotes the relative frequency insufficient, specified for strategy si at $\{s_1,s_2,\dots,s_l(max)\}$ DVS levels. The energy consumption model is derived from the power consumption module in complementary metal-oxide-semiconductor (CMOS) logic circuits. The power consumption of the CMOS-based processor defined to be the summation of capacitive, short-circuit and leakage power. The capacitive power (dynamic power consumption) is the most significant factor of the power consumption [13]. It can be calculated in the following way:

$$P=A.C.V^2.F \quad (8)$$

Where A is the number of switches per clock cycle, C is the total capacitance load, v is the supply voltage, and f is the frequency. The energy consumed by resource R_j for the execution task Ti at DVS level sl can be defined as follows:

$$E[i,j]=Y\times[(vs_1)(i)]^2\times[(fs_1)(j)]\times f\times EXT'[i,j,s_1] \quad (9)$$

Where Y=A*C assumed constant for a given resource, $vs_l(i)$ is a voltage supply value for strategy s_i, R_i at DVS level sl for computing task t_i, $fs_l(j)$ is the relative frequency, and ETC' [i, j, sl] is the lth coordinate of ETC' [i, j] vector.

$$Ei=\sum_{i\in L(j)}^{j\in T(i)}\{Eijl\} + \Upsilon\,[vs_{min}(i)]^2.[fs_{min}(i)].f.\,Idle\,(i)]$$
$$= \Upsilon.f.\{\sum_{i\in L(j)}^{j\in T(i)}[vs_{min}(i)]^2.[fs_{min}(i)].f.\,Idle\,(i) \quad (10)$$

Where T (i) is the set of tasks assigned to resource R_j, L (j) is the set of DVS level used for these tasks on resource R_i, vs_{min} (i), fs_{min} (i) represents the minimum supply voltage and relative frequency in the idle time that resource turn into sleep mode and Idle (i) is an idle time of resource R_i. The idle time for resource R_i can be calculated in following way:

$$Idle\ i= Makespan – completion\ (i) \quad (11)$$

For the resource with makespan, the idle time is equal to zero. So total energy consumption is as follows [4]:

$$E\_total=\sum_{i\in m} Ei \quad (12)$$

TABLE I. DVS Levels And Pairs

	Pair 1		Pair 2		Pair3	
Level	**Vol (Vs1)**	**Rel.f (fs1)**	**Vol (Vs1)**	**Rel.f (fs1)**	**Vol (Vs1)**	**Rel.f (fs1)**
0	1.5	1.0	2.2	1.0	1.75	1.0
1	1.4	0.9	1.9	0.85	1.4	0.8
2	1.3	0.8	1.6	0.65	1.2	0.6
3	1.2	0.7	1.3	0.50	0.9	0.4
4	1.1	0.6	1.0	0.35		
5	1.0	0.5				

D. Makespan

In this paper the scheduling aim is to minimize makespan and total energy consumption. Generally, there are two solutions:

1) Scheduling aims to find the smallest energy consumption when execution time is limited. That is E_opt=min(E_total),while makespan_opt<=makespan_expected.

Where E_opt is the minimum energy consumption, makespan_opt is the minimum makespan that can have.

*Corresponding author

2) Scheduling aims to find the smallest energy consumption when the cumulative energy consumption limited, that is makespan_opt=min (makespan), while E_opt <= E_expected. Makespan can be described as follows:

$$\text{Minimize } f_{(cs)} = F_{max}(s) \quad (13)$$

Where $f_{(cs)}$ is a candidate solution and F define the completion time of task T_i on resource R_j. For example, the scheduler has eight independent tasks scheduled on two resources and the sequence of tasks is T_6-T_5-T_2-T_1-T_7-T_4-T_3-T_0, Fig. 1 explains calculation method of makespan:

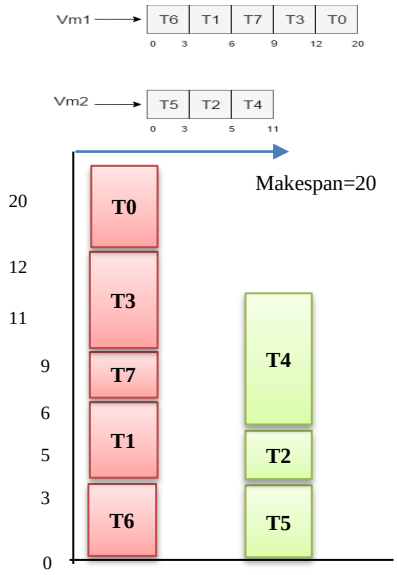


Fig. 1. Makespan definition.

We consider the scheduling as a bi-objective problem which aiming to find the right compromise between makespan and E_total.

The obtained results of the considered functions have not same unit. For this purpose, we use normalization. The used normalization formula is as follows:

$$\text{Normalization}(Q_c)=\begin{cases}\dfrac{Q_c\text{-}Q_i(max)}{Q_i(max)\text{-}Qi(min)} & Q_i(max)\neq Q_i(min)\\ 1 & \&Q_i(max)=Q_i(min)\end{cases} \quad (14)$$

Where Q_c is the obtained result, Q_i (max) is the maximum obtainable value, and Q_i (min) is the minimum value can be obtained. We are used minimum makespan and minimum energy consumption equal to zero and maximum makespan and energy consumption obtained by sends all the tasks to the weakest resource. There are two parameters in the total fitness function, any changes in the parameters show the user's demands from the scheduler. In other words, these two parameters used as coefficients. The sum of the parameters should be equal to one so the fitness function of the bi-objective scheduling is as follows:

$$(\theta 1)\times(\text{normal-makespan})+(\theta 2)\times(\text{normal}-\text{energy}) \quad (15)$$

Table 2 shows the examples of possible states:

TABLE II. FITNESS FUNCTION PARAMETERS

θ1	θ2	Makespan	Energy
0	1		✓
1	0	✓	
0.5	0.5	✓	✓

V. PROPOSED ALGORITHM

A. Chaotic Bat Algorithm

We describe a chaotic heuristic algorithm to send tasks to the makespan and energy aware resources, and we call it bi-objective chaotic bat Algorithm for task scheduling. CBA use the execution time, the execution energy to improve the task scheduling. All bats have properties that explained in related works (Bat algorithm) in five sections. The initial population mainly aims to find the food/prey and a faster convergence, as well as improvement in the best global solution. Each bat has some other parameters as follows:

α: is a loudness decay factor. It is also used as a cooling factor in the traditional simulated annealing algorithm.

γ: is the pulse enhancement factor that used for adjustment of the pulse frequency.

r_i: which makes the local search is done further and with more accuracy.

Ai: is loudness, which makes the algorithm explore the search space globally.

In this section, we are used chaotic maps in different ways to tune the BA parameters and improve the performance. In chaotic sequences, the numbers are well distributed. Iterative map and sinusoidal map has had good performance in the initialization process. This specification can be effective in an evolutionary algorithm. This feature can get a better exploration in the evolutionary algorithm. So, for each bat, we used the sinusoidal map to initialize pulse emission rate and the iterative map for loudness frequency initialization[6]. Descriptions of the two maps are as follows:

Iterative map: The iterative chaotic map with infinite collapses can be written as [6].

$$X_{k+1=Sin}(\frac{\alpha\pi}{xk}) \quad (16)$$

Sinusoidal map**:** we can define Sinusoidal map by the following equation:

$$X_{k+1}=\alpha x k^2 \sin(\pi x k) \quad (17)$$

Tasks randomly distributed between resources. If we consider 'n' as an initial population, so we have 'n' solutions (bats).We used the chaotic value distributions of 100 iterations for two maps with random initial values.

VI. SIMULATION AND RESULTS

Simulations were carried out to compare the optimization ability of the proposed algorithm (CBA) [14] in scheduling problem with the classical BA [5], GA [4], PSO with dynamic

*Corresponding author

inertia weight [15], [16] and Symbiotic organism [2]. We use chaotic maps for Performance improvements. The simulation carried out using Matlab (R2014a) and Table 3 shows parameter settings of the algorithms for task scheduling. We are considered the maximum algorithm iterations equal to 500, and in energy consumption and total fitness function we are used 1500 iteration to obtain convergence which is also considered as a condition for termination, and a fixed population n = 100 is used for all simulations.

TABLE III. ALGORITHMS PARAMETERS

BA	[r=random initialize] [fmin=0,fmax=number of resource] [**α=0.9] [γ=0.9] [** A= random initialize]
CBA	[r= initialize using Sinusoidal map][fmin=0,fmax=number of resource][**α=0.9][γ=0.9]** [A=initialize using Iterative map]
PSO	[w=0.4-0.9] [c1,c2=2]
GA	[cross over rate=0.9] [mutation rate=0.01]
DSOS	[number of organism=100]

Web applications such as web services are usually run for a long time and their CPU requests are variable. Moreover, High-Performance Computing (HPC) applications have short life span and place a high demand on CPU. Furthermore, chosen statistical models for task sizes represents different scenarios of concurrently scheduling HPC and web applications. Uniform distribution depicts tasks where HPC and web applications have the same value. The left-skewed distribution represents a state where HPC applications to be scheduled more than web applications and right skewed distribution represents the Reverse this state. The normal distribution represents a tasks where a single type of application is scheduled. To test the ability of the algorithms, we randomly generated five types of scenarios (tasks) which shown in Table 4. We use 15 resources and random numbers (1000 to 10000) for processing capacity of each resource (million instructions per second) and a number of machine instructions for each task generated using normal, uniform, right-skewed and left skewed distribution. The normal distribution contains more medium size tasks and fewer small and large size tasks. Left-skewed represents a few small size tasks and rather a large size tasks while right skewed is the opposite. Uniform distribution depicts an equal number of large, medium, and small size tasks. For each distribution, 20, 30, 50, 100, 200, 300 tasks were generated which they have been named as scenarios.

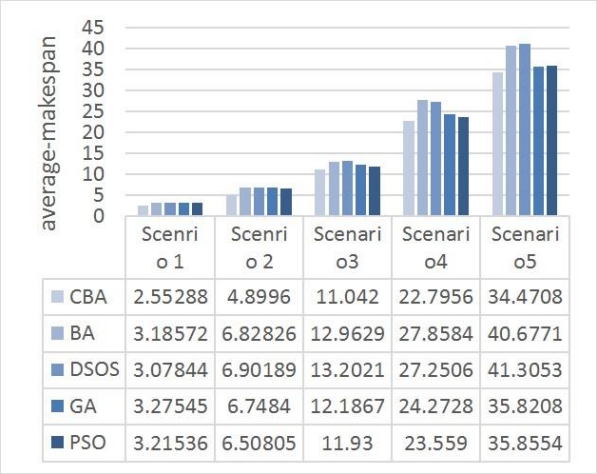
TABLE IV. SCENARIOS

Scenarios	**Number of tasks**	**Number of resources**
Scenario1	20	15
Scenario2	50	15
Scenario3	100	15
Scenario4	200	15
Scenario5	300	15

*Corresponding author

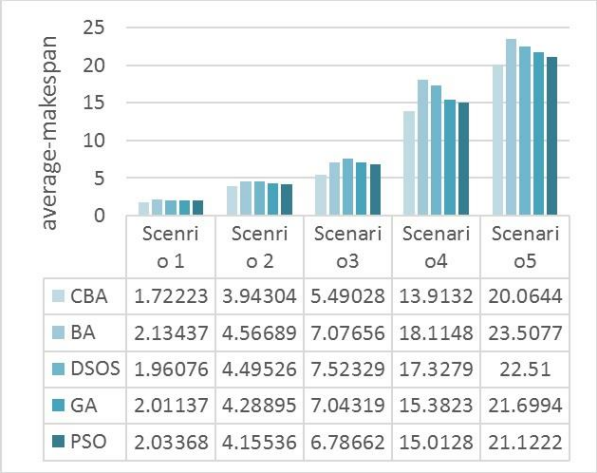
A. Scheduling Scenarios by Considering Makespan

The following experiments and analysis are based on the makespan including CBA, BA, GA, DSOS and PSO algorithm with dynamic inertia weight for normal, uniform, right and left skewed generated tasks.



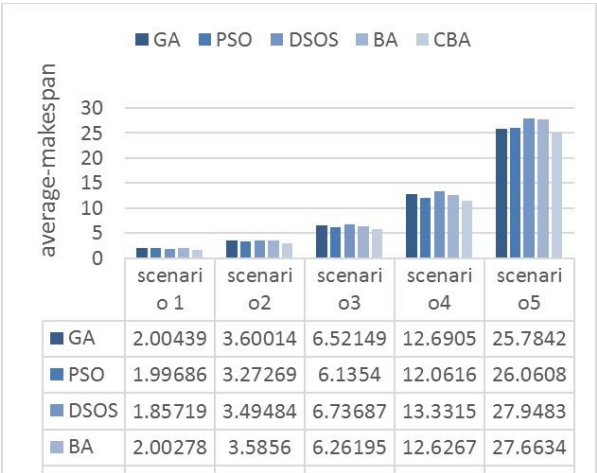
	Scenrio 1	Scenrio 2	Scenario3	Scenario4	Scenario5
CBA	2.55288	4.8996	11.042	22.7956	34.4708
BA	3.18572	6.82826	12.9629	27.8584	40.6771
DSOS	3.07844	6.90189	13.2021	27.2506	41.3053
GA	3.27545	6.7484	12.1867	24.2728	35.8208
PSO	3.21536	6.50805	11.93	23.559	35.8554

Fig. 2. Average of makespan in 10 repetition (normal).



	Scenrio 1	Scenrio 2	Scenario3	Scenario4	Scenario5
CBA	1.72223	3.94304	5.49028	13.9132	20.0644
BA	2.13437	4.56689	7.07656	18.1148	23.5077
DSOS	1.96076	4.49526	7.52329	17.3279	22.51
GA	2.01137	4.28895	7.04319	15.3823	21.6994
PSO	2.03368	4.15536	6.78662	15.0128	21.1222

Fig. 3. Average of makespan in 10 repetition (uniform).



	scenario 1	scenario2	scenario3	scenario4	scenario5
GA	2.00439	3.60014	6.52149	12.6905	25.7842
PSO	1.99686	3.27269	6.1354	12.0616	26.0608
DSOS	1.85719	3.49484	6.73687	13.3315	27.9483
BA	2.00278	3.5856	6.26195	12.6267	27.6634

Fig. 4. Average of makespan in 10 repetition (Right-Skewed).

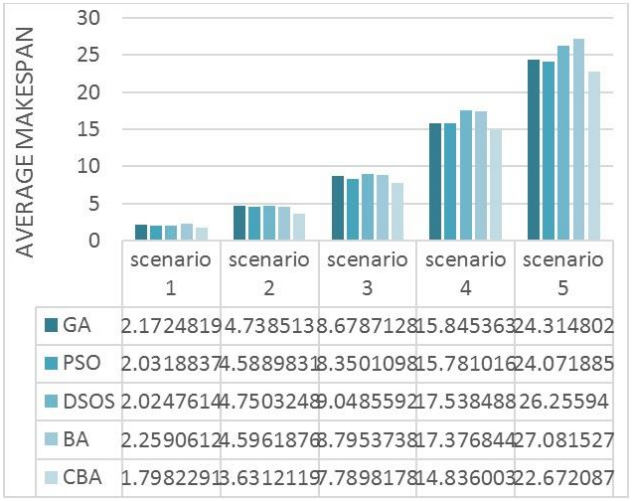


Fig. 5. Average makespan in 10 repetition (left skewed).

B. Scheduling Scenarios by Considering Energy Consumption

In this section, we test GA [4] and proposed algorithm by considering the energy consumption in four different distributions tasks.

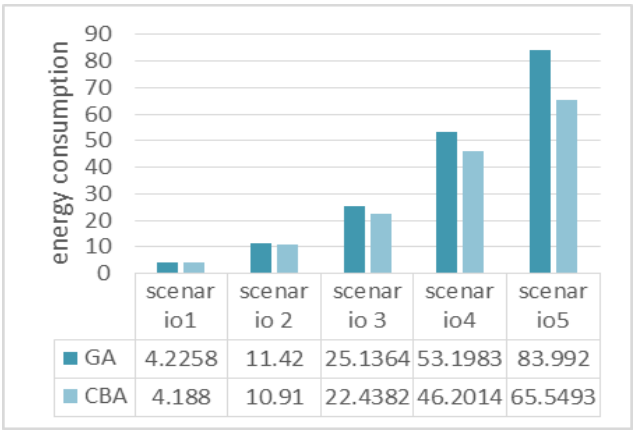


Fig. 6. Average energy in 10 repetition (normal).

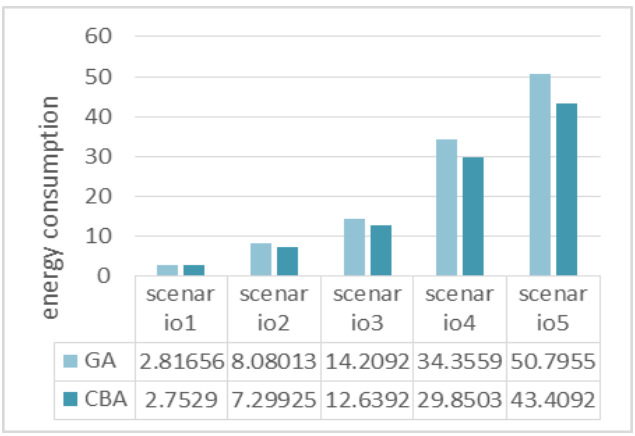


Fig. 7. Average energy in 10 repetition (uniform).

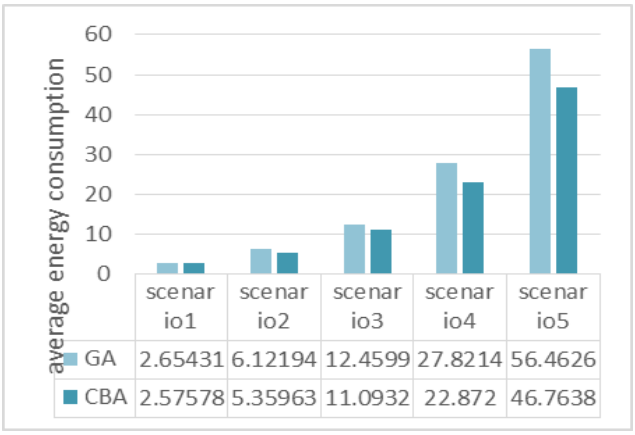


Fig. 8. An example of convergency (right-skewed).

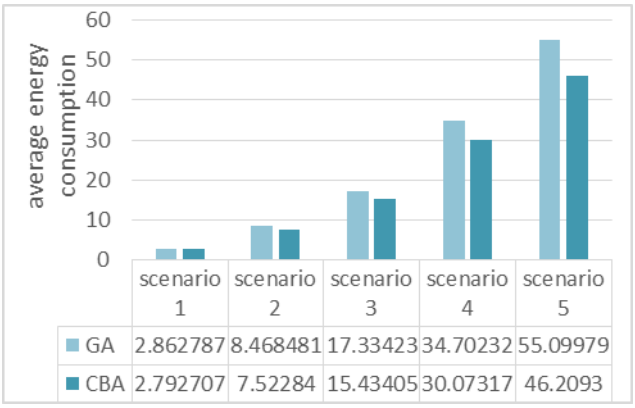


Fig. 9. Average energy in 10 repetition (left-skewed).

C. Scheduling Scenarios by Considering Total Fitness

In this section, experimental results show the Bi-Objective results of running each algorithm. In this results, θ value is equal to 0.5. Results shows that, by increasing the number tasks amount of energy consumption and makespan increases and proposed algorithm can obtain the better result than other algorithms.

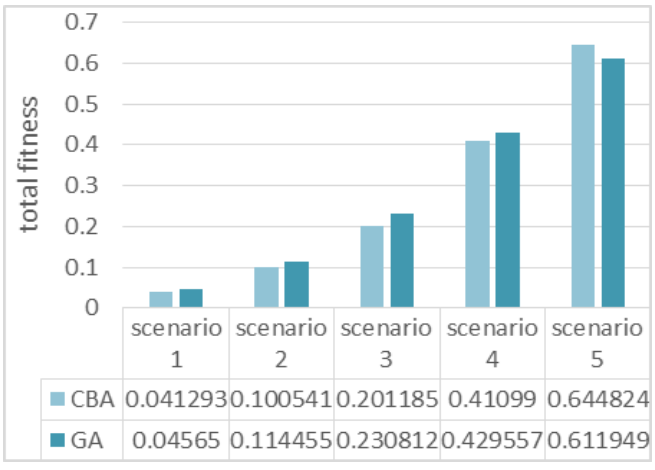


Fig. 10. Average total fitness in 10 repetition (normal).

*Corresponding author

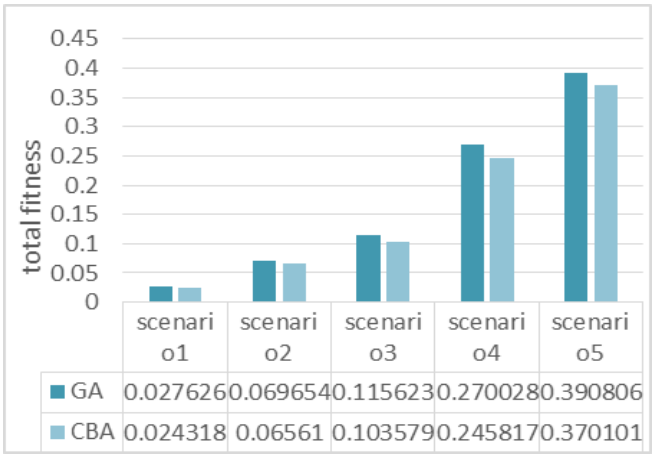


Fig. 11. Average total fitness in 10 repetition (uniform).

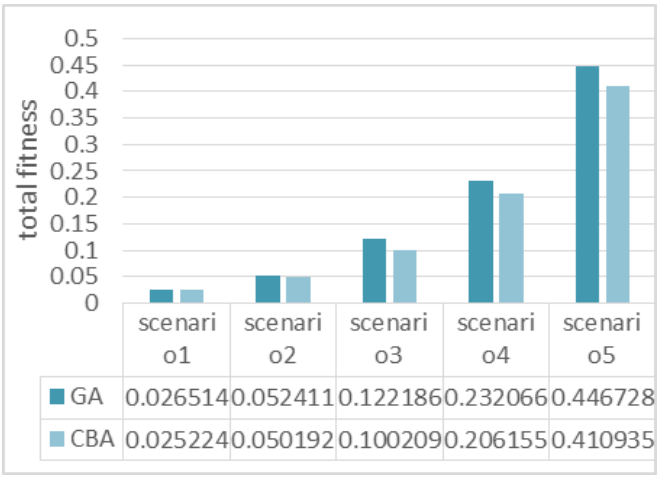


Fig. 12. Average total fitness in 10 repetition (right-skewed).

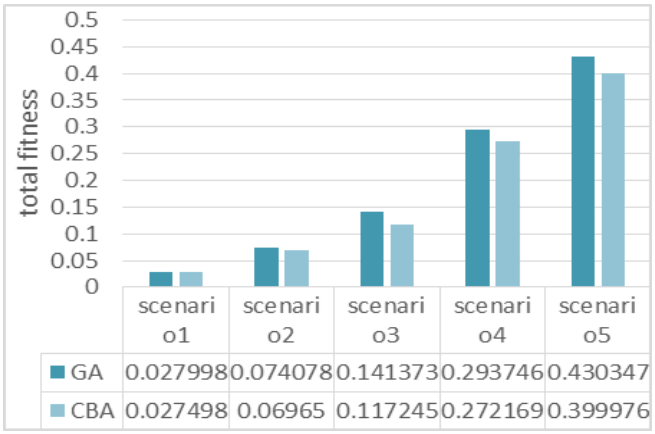


Fig. 13. Average total fitness in 10 repetition (left-skewed).

In Section VI-A, Fig. 2 to 5 show the average makespan of each algorithm by run 10 times in 5 scenarios with 5 type of tasks. It shows that the makespan will increase when the number of tasks increased. Also according to these figures, the proposed algorithm has achieved lower makespan than the other algorithms. It is noteworthy that when the number of incoming tasks are increased proposed algorithm has gained better results than GA but when the number of tasks is much, sometimes the CBA converged later than GA but obtained results are better than it. In section B As indicated, energy consumption is considered. Fig. 6 to 9 show the average energy consumption of each algorithm by run 10 times in 5 scenarios with 5 type of tasks. In the last part (total fitness), Fig. 10 to 13 represent obtained results by considering the energy consumption and makespan. Results shows the sum of the normalized energy consumption and makespan multiplied by Theta coefficient. It is obvious the proposed algorithm can obtain better total fitness than another algorithm. It is noteworthy that in some scenarios, especially when the number of tasks is less both algorithm results are very near but with the increasing number of tasks proposed algorithm has overtaken from genetic algorithm.

VII. Conclusion and Feature Scope

We have studied scheduling problem in cloud computing environments. This paper explained an advanced task scheduling algorithm based on chaos and the effect of using chaotic sequences for improvement of results. In this paper, the chaos maps have used to improve the performance of Bat Algorithm, as well as the global search by using a good distribution of numbers in order to escape the local optimum. The use of chaos is one of the techniques to tune some of the parameters in algorithms. In the recent optimization literature, chaos has become an active research topic and researchers paid special attention to it. By comparing obtained results by Chaotic Bat Algorithm and other algorithms, the results showed that the improvement of the makespan and energy consumption, due to use of deterministic chaotic signals in part of constant parameters. Experimental results of the CBA proposed that the tuned algorithms can clearly improve the reliability and the convergence of the global optimality, and they also enhanced the quality of the results. When we use a CBA, running time of chaos maps leads to increase the total running time of the algorithm However, this time is minimal. Second, the algorithm late converged by taking energy Criterion when the number of tasks was much. An interesting question arises how some chaotic maps can improve the performance of an algorithm, while others do not. It is still not clear why the use of chaos in an algorithm to replace some parameters can change the performance. Experimental results show that the proposed algorithm in this problem is superior to other heuristics algorithms.

References

[1] "Amazon EC2. https://aws.amazon.com/ec2."

[2] M. Abdullahi, M. A. Ngadi, and S. M. Abdulhamid, "Symbiotic Organism Search optimization based task scheduling in cloud computing environment," Futur. Gener. Comput. Syst., vol. 56, pp. 640–650, 2016.

[3] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," Egypt. Informatics J., vol. 16, no. 3, pp. 275–295, 2015.

[4] C. T. Ying and J. Yu, "Energy-aware genetic algorithms for task scheduling in cloud computing," Proc. - 7th ChinaGrid Annu. Conf. ChinaGrid 2012, pp. 43–48, 2012.

[5] X.-S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," in Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), J. R. González, D. A. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 65–74.

[6] A. H. Gandomi and X. S. Yang, "Chaotic bat algorithm," J. Comput. Sci., vol. 5, no. 2, pp. 224–232, 2014.

*Corresponding author

[7] P. Kumar and A. Verma, "Independent Task Scheduling in Cloud Computing by Improved Genetic Algorithm," Int. J. Adv. Res. Comput. Sci. Softw. Eng. Res., vol. 2, no. 5, pp. 5–8, 2012.

[8] R. Aron, I. Chana, and A. Abraham, "A hyper-heuristic approach for resource provisioning-based scheduling in grid environment," J. Supercomput., vol. 71, no. 4, pp. 1427–1450, 2015.

[9] C.-W. Tsai, W.-C. Huang, M.-H. Chiang, M.-C. Chiang, and C.-S. Yang, "A Hyper-Heuristic Scheduling Algorithm for Cloud," Cloud Comput. IEEE Trans., vol. 2, no. 2, pp. 236–250, Apr. 2014.

[10] M. A. Tawfeek, A. El-Sisi, A. E. Keshk, and F. A. Torkey, "Cloud task scheduling based on ant colony optimization," in Computer Engineering Systems (ICCES), 2013 8th International Conference on, 2013, pp. 64–69.

[11] A. S. Wu, H. Yu, S. Jin, K. C. Lin, and G. Schiavone, "An incremental genetic algorithm approach to multiprocessor scheduling," IEEE Trans. Parallel Distrib. Syst., vol. 15, no. 9, pp. 824–834, Sep. 2004.

[12] S. M. Alaoui, O. Frieder, and T. El-Ghazawi, "A parallel genetic algorithm for task mapping on parallel machines," in Parallel and Distributed Processing: 11th IPPS/SPDP'99 Workshops Held in Conjunction with the 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing San Juan, Puerto Rico, USA, April 12--16, 1999 Proceedings, Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 201–209.

[13] R. Ge, X. Feng, and K. W. Cameron, "Performance-constrained Distributed DVS Scheduling for Scientific Applications on Power-aware Clusters," in Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, 2005, p. 34–.

[14] A. H. Gandomi and X.-S. Yang, "Chaotic bat algorithm," J. Comput. Sci., vol. 5, no. 2, pp. 224–232, 2014.

[15] G. Yue-lin, "A New Particle Swarm Optimization Algorithm with Random Inertia Weight and Evolution Strategy," Int. Conf. Comput. Intell. Secur. Work., pp. 199–203, 2007.

[16] H. S. Al-Olimat, M. Alam, R. Green, and J. K. Lee, "Cloudlet scheduling with particle swarm optimization," Proc. - 2015 5th Int. Conf. Commun. Syst. Netw. Technol. CSNT 2015, pp. 991–995, 2015.

*Corresponding author