Research Article

International Journal of Advanced Computer Research, Vol 9(43)
ISSN (Print): 2249-7277 ISSN (Online): 2277-7970
http://dx.doi.org/10.19101/IJACR.PID29

Non-preemptive chaotic cat swarm optimization scheme for task scheduling on cloud computing environment

Danlami Gabi[1*], Nasiru Muhammad Dankolo[2], Abdul Samad Ismail[3], Anazida Zainal[4] and Zalmiyah Zakaria[4]
Lecturer, Kebbi State University of Science and Technology, P.M.B 1144, Aliero, Kebbi State, Nigeria[1]
Assistant Lecturer, Kebbi State University of Science and Technology, P.M.B 1144, Aliero, Kebbi State, Nigeria[2]
Professor, School of Computing, Universiti Teknologi Malaysia, P.M.B. 81310, Skudai, Johor, Malaysia[3]
Senior Lecturer, School of Computing, Universiti Teknologi Malaysia, P.M.B. 81310, Skudai, Johor, Malaysia[4]

Received: 9-September-2018; Revised: 28-November-2018; Accepted: 30-January-2019

Abstract
With exponential growth in the number of customers accessing the cloud services, scheduling tasks at cloud datacenter poses the greatest challenge in meeting end-user's quality of service (QoS) expectations in terms of time and cost. Recent research makes use of metaheuristic task scheduling techniques in addressing this concern. However, metaheuristic techniques are attributed with certain limitation such as premature convergence, global and local imbalance which causes insufficient task allocation across cloud virtual machines. Thus, resulting in inefficient QoS expectation. To address these concerns while meeting end-users QoS expectation, this paper puts forward a non-preemptive chaotic cat swarm optimization (NCCSO) scheme as an ideal solution. In the developed scheme, chaotic process is introduced to reduce entrapment at local optima and overcome premature convergence and Pareto dominant strategy is used to address optimality problem. The developed scheme is implemented in the CloudSim simulator tool and simulation results show the developed NCCSO scheme compared to the benchmarked schemes adopted in this paper can achieve 42.87%, 35.47% and 25.49% reduction in term of execution time, and also 38.62%, 35.32%, 25.56% in term of execution cost. Finally, we also unveiled that a statistical significance on 95% confidential interval has shown that our developed NCCSO scheme can provide a remarkable performance that can meet end-user QoS expectations.

Keywords
Cloud computing, Cat swarm optimization, Chaotic process, Pareto dominance.

1.Introduction
Cloud computing is a consumable technology of our time that allow sharing of resources (e.g. virtual machines, storage, bandwidth) to meet the exponential demand of cloud end-users [1−4]. Three service models associated with cloud computing environment include software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS). The SaaS layer of cloud computing allows cloud customers to run applications remotely, where application delivery is carried out via the Internet and manage by a third-party vendor. The cloud customers often interact with SaaS layer to get their task submitted to the IaaS layer (e.g. datacenter) and later receives its processed results via the SaaS layer.

The SaaS layer function with the support of PaaS layer, providing interactive mechanisms for both cloud customers and service providers [5] while PaaS allow cost-efficient development and deployment of applications [6,7]. On the other hand, the IaaS layer provides services for cloud customers in terms of infrastructure (e.g. virtual machine) as a service. The IaaS provide a pool of resources of varied types that can be leased by cloud customers according to their computing requirements.

Currently, due to escalating number of end-users accessing the cloud services, providing efficient scheduling to meet their QoS expectations has become a greater concern. Although, high-level research has been conducted in unveiling the supremacy of metaheuristic techniques toward mitigating these concerns [8], however, the metaheuristics are attributed to global and local

*Author for correspondence

imbalanced and slow convergence speed which leads to insufficient task schedule on cloud virtual machines. Thus, affecting the provisioning of customers QoS expectation. The need to improve metaheuristic scheduling schemes to provide better QoS that can meet end-user's expectation is paramount. The conventional cat swarm optimization (CSO) is a metaheuristic technique put forward in [9]. This technique mimics the behavior of natural cat and has relatively proven better in term of both global and local convergence than particle swarm optimization (PSO) [9, 10].

To ensure the conventional CSO algorithm become suitable for cloud task scheduling, we adopted the use of chaotic process and Pareto dominance and developed an NCCSO scheme. In the developmental scheme, the chaotic and Pareto dominant strategy is used to overcome the problem of local and global imbalanced and slow convergence speed. A multi-objective QoS task scheduling model based on execution time and execution cost is then proposed upon which our NCCSO scheme was used to solve the model on CloudSim simulator tools. Experimental results via simulation has shown that we developed NCCSO scheme had an outstanding performance compared to that of the benchmarked schemes.

The contribution of this study is as follows:

- Development of a multi-objective task scheduling model for cloud computing.
- An improvement is developed for the CSO using chaotic and Pareto-dominance approach and the resultant solution in solving task scheduling problem is cost optimal and minimum computation time.
- Development of an NCCSO task scheduling scheme to addressing customers QoS expectations.

2.Related work

Task scheduling has become a significant research topic with the objective of ensuring that every computing resources (e.g. virtual machines) are equitably distributed on cloud tasks to meet customers QoS expectations. Recent research has shown that a set of scheduling strategies (based on heuristic and metaheuristic techniques) can solve multi-objective task scheduling problem with the aim of ensuring customers' QoS expectation. Although metaheuristics techniques can handle the large tasks scheduling problem and converges faster than the heuristic techniques, their incorporation with chaotic process/or some greedy based techniques can further increase its performance towards addressing cost and time objectives. Researchers that explored the concept of incorporating and hybridizing these techniques in addressing the concerns of the cloud task scheduling problem are discussed in the following:

In Zuo et al. [8], a Multi-Objective Ant Colony Optimization (MOACO) algorithm is developed to minimize the makespan time and customer budget cost. According to the researchers, their proposed cost model reflects the relationship between customer 's resource cost and budgetary cost. Their simulation results have shown the proposed algorithm can achieve the optimum solution for both performance and cost. In Ramezani et al. [11], the researchers developed a multi-objective particle swarm optimization (MOPSO) algorithm. The results of the experiment via simulation with CloudSim simulator shows their method can find the optimal solution in a reasonable amount of time. Although hybridization, can help improve better quality of solution for end-user's service preferences during task scheduling. In Gabi et al. [10], the researchers put forward an orthogonal Taguchi-based cat swarm optimization (OTB-CSO) algorithm to improve the performance of cloud computing systems in term of makspan. The researchers incorporated Taguchi Orthogonal approach in the local search of the conventional CSO to increase its convergence speed which later reduces local trapping that led to minimum makespan time. However, the algorithm only handles a single objective optimization problem. Improving this algorithm can help to address a multi-objective task scheduling problem.

Liu et al. [12] developed an improved min-min algorithm for cloud computing environment. The researchers aimed at addressing three basic objectives (quality of service, dynamic priority model and cost of service) for their scheduled task. The simulation results from their proposed algorithm when compare with the traditional min-min algorithm unveiled that it can increase resource utilization rate and execute longer task at reasonable times. In another development, Xu et al. [13] put forward a multi-objective genetic optimization algorithm (MOGA) to minimize average completion time, total completion time. In their scheduling process, a complex, large task was divided into multiple sub-tasks where allocation of task of research is carried out based on chromosomes encoding. The researchers designed three different fitness function to evaluate the fitness of each chromosome line with their task scheduling

objectives. The simulation results as unveiled by the researchers has shown that their developed MOGA algorithm can produce faster convergence speed which led to better performance. Khajehvand et al. [14] put forward a hybrid first-fit cost-time trade-off (FCTT) and workflow planning cost-based (WPC) model. Their objective is to minimize task runtime and execution cost. In the schedule method adopted by the researchers, each task is assigned a rank using a bottom-up traversal technique which allows child tasks to first be assigned a rank prior to the parent task assignment. The tasks are then sorted by ranking them in a non-increasing manner. The researchers then adopted the FCTT scheduling algorithm to select task whose execution of all parents' tasks is completed. The experimental results via simulation shows their proposed FCTT can reduce task runtime and allocation cost compared with MOGA and best effort (BE) algorithms. However, task updating method by WPC can lead to long computation time, since the performance of the algorithm depends upon it update process.

From the review conducted on related work, metaheuristic techniques usually exhibit certain limitations such as high dimensional complexity, slow convergence speed, local trapping and imbalance between global and local optima. All these limitations can lead to inefficient task schedule. Therefore, an improvement is required to provide an ideal solution to cloud task scheduling process in cloud computing. This study addresses the concern of scheduling of task at the IaaS layer of cloud computing with a focus on multi-objective task scheduling.

3.Cat swarm optimization and the need for improvement

In [9], the researchers introduced CSO technique that mimics common behavior of natural cat. As put forward by the researchers, their proposed algorithm operates in two modes; the resting (seeking) and chasing (tracing) mode (detail about these modes can be found in [9]. A control factor known as the mixed ratio (MR) is used to discover if the position at which the cat is currently standing is in either seeking or tracing mode. The position of the cat signifies possible solution sets, while the velocity of the cat is associated with a dimension and a fitness value. According to [9], as the cat progresses closer to the solution (fitness), the cat updates itself with best results in the memory continuously until all cats achieve the best solution (also known as the fitness) [15].

Although, the conventional CSO exhibit better performance in term of convergent speed than PSO [9], however, certain limitations are associated with the conventional CSO. The global search may not always provide a superior solution when its search space increases exponentially. Besides, the number of cats that usually goes into the global search region of the conventional CSO always exceeds that of the local search. These can lead to poor convergence speed, causing tasks entrapment at the local optimal. On the other hand, operation executed by the conventional CSO at both global and local searches is independently carried out for each iteration. This, likewise, causes velocity and position update of the conventional CSO to perform a similar process, thus, leading to long computation time. Similarly, imbalance between global and local search is another concern exhibited by the conventional CSO. Therefore, there is a need to improve on the conventional CSO technique so that efficient scheduling can be achieved in a cloud computing environment [16].

4.The Pareto dominance and chaotic process

For any task scheduling problem, the chances of locating an optimal solution that will ensure minimum execution time and cost is becoming harder in a large search space like the cloud environment [17]. Due to multi-criteria associated with scheduling of task in the cloud, the concept of optimality needs to be achieved. Multi-objective optimization is characterized with trade-off factors, where each of the trade-off that serves as a solution correspond to a specific order of importance of the objectives [18].

In another development, dynamic and nonlinear systems usually exhibit certain processes that look similar to a deterministic chaotic process. The chaotic process contains a very high sensitivity to initial condition and parameter change. It is derived from the term chaos known as randomness of a simple dynamic system. This motivates its use as a source of randomness in optimization theory on various fields. Recent research, e.g., in [19] have shown the adoption of chaotic sequences in stochastic optimization techniques for providing population diversity in the search space which ensured global convergence and avoidance of local optima entrapment. The equation of the chaos is shown in Equation (1) and the logistic course adopted in [20] will be used in this paper due to its successful application to address optimization problems.

$$y_{n+1} = \lambda\big(y_n(1 - y_n)\big); n = 1,2,3, \ldots \quad (1)$$

where$\{y_n\}$ ... 1,2,3 is the sets of numbers that are generated from logistic chaotic map; $\lambda \in [0,4]$ is the control parameter of logistic equation; $y_n \in (0,1)$ is the n^{th}chaotic number and $y_0 \in (0,1)$ and $y_0 \notin \{0.0, 0.25, 0.50, 0.75, 1.0\}$.

The incorporation of chaotic approach provides better tasks mapping sequence and overcome premature convergence to reduce task execution time [19]. In another development, Pareto-Optimization can be used to provide cloud end-users with many non-dominant solutions as possible, by allowing sets of trade-offs between execution time and execution cost [21, 22]. These will guide the end-users to select their service preferences. Since the actual cloud end-users' service preference is very difficult to predict in cloud computing, cloud end-user's attention can be restricted on trade-off points *P** known as Pareto front, where cloud customers select their service preference in terms of the virtual machine that guarantees minimum execution time and cost [14]. Since in cloud computing environment, end-users often affected by their budget constraint, [15], several issues may still surface when several cloud end-users request cloud resources at the same time. The cloud end-users not only have to manage conflicting requirements under its budget constraint, but also have to manage trade-off between time and cost in such a way that could guarantee the execution of their task in minimum time [23]. The main goal of this research is to make sure that the Pareto optimal set are discovered for all tasks schedule based on the proposed scheduling technique. This study used the following definition in solving our multi-objective task scheduling problem:

Definition 1.1 Multi-objective Optimization problem
A typical multi-objective optimization problem can be expressed as a minimization of a K components of a vector function f_i in the form [11]:

$$\mathrm{M}\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x}), \dots f_k(\vec{x})], \quad (2)$$

where, vector of decision variable is given as $\vec{x} \; \forall \; \vec{x} = \{x_1, \dots,\}$ such that $f_i: R^n \to R, i = \{1,2, \dots, k\}$ are the objective functions in a universe U. $\vec{f}(\vec{x})$ is the multi-objective function.

5.Problem description

The problem is first represented by considering a set of independent tasks to be schedule on sequence of heterogeneous virtual machines for processing. $V = \{ v_k \mid m \geq k \geq 1 \}$, are the sets of virtual machines and m is the number of virtual machines. $T = \{ t_i \mid n \geq i \geq 1 \}$ represents the tasks groups and n is the overall number of tasks [22]. The goal of the scheduling problem is to dynamically assign each tasks $t_i \; \forall \; i = \{1,2, \dots, n\}$ to appropriate cloud virtual machines $v_k \; \forall \; k = \{1,2, \dots, m\}$ in order to determine the sequence as well as the timing of task executed and the amounts of cost for executing the task. This is followed with a task scheduling model that accommodate various objectives, such as the minimization of execution time and minimization of the execution cost (if tasks incur a cost) for a specified demand. We assume the following descriptions for the scheduling problem formulation: (i) Two datacentres are considered for the schedule; (ii) The datacentres belong to the same service provider, where the cost of transmission is ignored; (iii) Tasks are assigned to virtual machine sequentially and the total number of all possible schedules is $(n!)^m$ for the problem with n number of task and m number of virtual machine; (iv) Pre-emptive scheduling allocation policy is not allowed; (v) The cost of using a virtual machine for a time quantum varies from one virtual machine to the other. By considering an entry such as the Expected Time to Compute (ETC) matrix in Equation (3), our goal is to make sure each tasks are dynamically assign to virtual machine $v_k$with the right computing capacity in order to find the optimum value of the total expected execution time and total expected execution cost incurred in executing all tasks.

$$ETC = \begin{bmatrix} t_1v_1 & t_1v_2 & \cdot & \cdot & \cdot & t_1v_k \\ t_2v_1 & . & \cdot & \cdot & \cdot & . \\ . & . & \cdot & \cdot & \cdot & . \\ . & . & \cdot & \cdot & \cdot & . \\ t_nv_1 & . & \cdot & \cdot & \cdot & t_nv_k \end{bmatrix} \quad (3)$$

Let $\{t_1, t_2, \dots, t_n\}$ denote set of tasks independent of one to another and $\{v_1, v_2, \dots, v_k\}$ denote the set of heterogeneous virtual machines. Suppose $t_i \forall i = \{1,2, \dots, n\}$ is scheduled on a v_k, the execution time $exec_k$ of all tasks processed on a v_k is computed using Equation (4) (Ramezani et al., 2013). The total execution time $Texe_k$ of all tasks t_i processed on all virtual machines $v_k \; \forall k = \{1,2, \dots, m\}$ is computed using Equation (5) [24].

$$exec_k = \sum_{i=1}^{n} x_{ik} \times \frac{t_i}{npe_k \times v_{kmips}}, \quad (4)$$

$$Texec_k = \sum_{k=1}^{m} \sum_{i=1}^{n} x_{ik} \times \frac{t_i}{npe_k \times \mathrm{v_{kmips}}} \quad (5)$$

Where, $exec_k$ is the execution time of running tasks on one virtual machine; x_{ik} is equal to one 1 if task t_i is assign to v_k otherwise, $x_{ik} = 0$; t_i is the task whose length is given in Millions Instructions (MI); v_{kmips} is the v_k speed whose unit is in Million Instructions Per Second (MIPS); npe_k is the number of processing element of a virtual machine v_k

The execution cost model is a multi-objective model that incorporates the execution time model as shown in Equation (5). This model is based on pay-as-you-go basis, where cloud customers are charged according to time quantum, they have used the virtual machines. The time quantum [25], is the smallest discrete unit to compute cost of using a virtual machine. In this paper, we assume the cost of memory and central processing unit (CPU) are all included in the monetary cost of using a virtual machine. For instance, assume for every one-minute *N* of using a virtual machine, the price specified by the service provider is 0.5 dollars per hour(/hr), then, for a time period in minutes of using a virtual machine, the cost will be computed as $(N * 0.5)/60$ dollars.

Therefore, virtual machines with highest execution time will always return the lowest execution cost and vice versa. Assume the cost of executing tasks on a virtual machine per hour (hr) is known, Equation (6) hold for cost exe_{cost_k} of executing tasks t_i on a virtual machine v_k per time quantum in second [11].

$$exe_{cost_k} = \times \frac{1}{3600} \sum_{i=1}^{n} v_{kcost} \times x_{ik} \times \frac{t_i}{npe_k * V_{mips_k}} \quad , \quad (6)$$

where, v_{kcost} is the monetary cost of one unit v_k in US dollar per hour.

Such that:

$$x_{ik} = \begin{cases} 1, & if\ task\ i\ is\ assign\ on\ v_k \\ 0, & otherwise \end{cases} \quad (7)$$

When more than one $v_k\ \forall k = \{1,2,\dots,m\}$ are used by a service provider to execute several tasks, the total tasks execution cost, $TTexe_{cost_k}$, by all virtual machines in a datacenter iscomputed using Equation (8).

$$TTexe_{cost_k} = \frac{1}{3600} \sum_{k=1}^{m} \sum_{i=1}^{n} vcost_k \times x_{ik} \times \frac{t_i}{npe_k * V_{mips_k}} \quad (8)$$

The multi-objective task scheduling mathematical model can therefore by expressed as follows:

$$Min\ \text{F (X)} = \left[f(Texec_k), f\left(TTexe_{cost_k}\right)\right] \quad (9)$$

Subject to:

$$\sum_{k=1} x_{ik} = 1, \quad \forall\ i = 1,2,\dots\dots,n; \quad x_{ik} \in \{0,1\}, \forall i,k$$

Equation (9) is the multi-objective optimization model to be solved by applying our proposed multi-objective scheme.

6.The developed task scheduling scheme

The proposed NCCSO scheme consists of two phases (global and local search) that combined and solved the optimization problem. The following attributes were first considered to arrive at an optimal solution; the tasks number, budget costs, the number of virtual machines and other relevant parameters such as count dimension to change (CDC), seeking range of selected dimension (SRD), the seeking memory pool (SMP) and self-position considering (SPC) of the cat. Each cat symbolizes the choice of a virtual machine used for the task schedule. This is encoded in [1 × n] vector, with *n* belonging to a number of tasks. We assumed that each virtual machine in a datacenter has different cost per time quantum (heterogeneous). Based on the expected time to compute (ETC) matrix, when the tasks are schedule by the NCCSO scheduler, each task is assigned a cat (where the cat represents the virtual machine). Every cat has dimension *D* with *n* tasks assigned and the model associated with each cat are based on two objective function; the total execution time $\left(f(Texec_k)\right)$ and total execution cost $\left(f\left(TTexe_{cost_k}\right)\right)$. When a cat traverses all tasks, the cat formed a feasible solution to the problem. Each cat has both position and velocity vector. The position of the cat symbolizes the solution that is attained by the cat. A Mixed Ratio (MR) is used to specify two group of cats (seeking and tracing). The cats are moved into either seeking or tracing mode at random using value specified by the *MR*. When the cats reach their desired target, their fitness value are computed based on defined objective of the scheduling problem ($Texec_k$ and $TTexe_{cost_k}$).

A dominant strategy is used to compare the optimum solution and is stored at the archive, where the final velocity that formulates the latest velocity is selected. This velocity returns optimal solution which is used to compute the new position of the cat as shown in Equation (10).

$$\vec{X}_{k,d} = \vec{X}_{k,d} + \vec{V}_{k,d} \quad (10)$$

Where, $\vec{X}_{k,d}$ is the position of the Cat; $\vec{V}_{k,d}$ is the velocity attain by the cat. In order to ensure the

quality of the solution, avoid being trapped at the local optima, the quality of the feasible solution is evaluated using the fitness function. The fitness function is set based on an optimization model of the scheduling problem. As earlier reported, the optimal solution of each cat is stored at the archive. Every cat is assessed with a value of fitness functions and all Pareto optimal solutions that are stored at the archive. Hence, the fitness function $QoS(\vec{X})$ using Equation (11) was used for the evaluation:

$$QoS(\vec{X}) = \sum_{j=1}^{m} W_j f_j(\overrightarrow{X_\iota}), \{\forall \overrightarrow{X_\iota} \epsilon\ Archieve\} \quad (11)$$

Where, m is the number of objective functions and W_j is the preference weight for every objective function $(f_j(\overrightarrow{X_\iota}))$. Algorithm 1 shows the pseudocode for the developed NCCSO task scheduling algorithm.

Algorithm 1: NCCSO-based algorithm

INPUT: Task number, lengths, *initialize* mixed ratio *MR*; *virtual machine (VM) number and their required attributes (number of processing elements, unit cost of using one VM);*
Generate *an empty non-dominant archive of (n × m) size of uniform random number [0, 1]*
Compute all cats according to defined objective (Fitness) functions
OUTPUT: Optimal Task Schedule
Identify best optimal solution for the trade-off values.
Compare fitness functions of all cats, keep position with best fitness value into the archive
1. **While** condition is not reached
2. increment_iteration_number ← t + 1
3. **DO**
4. **If** $(flag \leftarrow 1)$
5. Generate y $(where\ y = SMP)$ copies of $k - th$ cat
6. Change at random the dimension of cats as per CDC using $+\ or\ -$ mutation operation
7. Determine the fitness of changed cats.
8. Discover suitable cats based on their fitness values.
9. **Else**
10. ***Chaos (): chaotic generated number according to Equation (1)***
11. Calculate the fitness values of the n experiments using Equation (11) and store Pareto optimal value in the archive
12. ***Endif***
13. ***If*** $(X_{ibest} \succ X_{jbest})$
14. $X_{ibest} = X_{gbest}$
15. $X_{cbest} = G_{best-id}$ *// current best position becomes the global best position*
16. ***If (termination condition reached)***
17. **Output** position of best minimum total execution time
18. **Output** position of the best minimum total execution cost
19. ***Else***
20. *Go to step 2*
21. ***Endif***
22. ***Endif***
23. ***EndWhile***

Simulation environment

CloudSim 3.0.3 [26] tool was installed on Eclipse-Java-Luna-SR2-win32-x86-64 for the simulation. The developed NCCSO task scheduling scheme is compared with Multi-Objective Ant Colony Optimization (MOACO) [8], Multi-Objective Particle Swarm Optimization (MOPSO) [11] and Min-Min [12] task scheduling schemes. The choice of properties for the datacenter host, task and virtual machines were selected as used in [16]. The estimated cost (0.17$ + 0.05$=0.22$/hr) of using a unit virtual machine for a time quantum is based on (Ramezani et al., 2015), while the selected values for inertia weight and coefficient factors $(c_1 c_2)$ for the MOACO, MOPSO, and NCCSO were based in [27]. The parameter settings for the scheduling algorithms are; 1:) MOPSO (Particle size:100, Self-recognition coefficients $(c_1 c_2)$: 2.0, Uniform random number (R_1): [0,1], Maximum iteration:1000, Inertia weight (W): 90-40%, Mixed ratio:2%); 2:) NCCSO (Cat size:100, Count Dimension to Change: 5%, Self-recognition coefficients (c_1): 2.0, Uniform random number (R_1): [0,1], Maximum iteration:1000, Inertia weight (W): 90-40%, Mixed ratio: 2%) and 3:) MOACO (Pheromone persistence α: 0.3, Importance

of pheromone (γ):1, Importance of resource innate attribute (β):1, Pheromone evaporation value (ρ): 0.3, Iteration number:1000, Number of ant m: 100).

7.Simulation results and discussion

In order to provide a concise explanation on the results obtained, we scheduled task instances from 20 to 100 on 20 heterogeneous virtual machines. Ten (10) independent simulation runs were conducted and the efficiency of the developed NCCSO was revealed. *Table 1* shows the results of the average value of the simulation runs. *Table 2* shows improvement gained by the developed NCCSO task scheduling scheme compared to the benchmarked schemes. The performance improvement is computed using Equation (12) [10]. To further reveal how significant our developed NCCSO over the benchmarked algorithm.

$$\text{PIR}(\%) = \frac{Total\ execution\ time(\ other\ scheme) - total\ execution\ time\ (NCCSO)}{Total\ execution\ time(\ other\ scheme)} * \times 100 \quad (12)$$

In *Table 1* precisely, Min-Min scheduling scheme showed reduction in execution time and execution cost compared to MOACO, MOPSO and NCCSO schemes when 20 to 50 task instances are scheduled on virtual machines. As the task scheduling intervals changes over time (see task instances from 70 to 100), its performance degrades further. This is because, unlike the metaheuristic, the heuristic techniques usually performe better with small task instances. When the task sizes become larger, their performance is seen degrading often at times. This makes the Min-Min scheduling scheme not suitable for scheduling tasks on cloud computing environment where tasks instances are unpredictable in numbers as they arrive the cloud datacenters. The MOACO task scheduling scheme likewise performed better with 20 to 40 tasks instances scheduled on heterogeneouse virtual machines. Its performance suddenly degrades with increase in task sizes.

This can be attributed to the traversing process of the ant colony technique as its convergence speed tend to be slower, due to tasks increase in the scheduling process which normally causes its entrapment at the local region. In another development, the MOPSO task scheduling scheme shows better improvement when 50 to 100 tasks are schedule on heterogeneous virtual machines compared to MOACO and Min-Min scheduling schemes. Although, Min-Min scheduling schemes shows better improvement with 20 to 30 tasks instances scheduled on 20 heterogeneous virtual machines, the results revealed that, as task instances gets larger, our developed NCCSO scheduling scheme gets better compared to all the benchmarked schemes in term of minimum execution time and execution cost. This shows that the developed scheme can adapt fluctuating tasks sizes, making it suitable for cloud computing environments. To further unveil the advantages of our developed scheme over the benchmarked schemes, *Table 2* shows the results on performance improvement gained. In the overall performance, our proposed NCCSO task scheduling scheme was able to minimized the total tasks execution time of 42.87%, 35.47% and 25.49% compared to Min-Min, MOACO and MOPSO task scheduling schemes, and also outperformed the benchmarked algorithm in term of execution cost with 38.62%, 35.32% and 25.56% improvement.

From the simulation results, it has been shown that our developed NCCSO task scheduling scheme has ability to find the best virtual machines with minimum execution of time and cost to execute end-user's tasks compared to the benchmarked schemes. *Figures 1* and *2* is used to further illustrated the performance of our proposed NCCSO scheme over the benchmarked schemes. This performance is attributed to the incorporation of chaotic and Pareto based approaches at the local search procedure of the conventional CSO. This helps in traversing all cats (virtual machines) and find best virtual machines suitable for the cloud end-users by overcoming premature convergence at the local search process of the metaheuristic technique. This performance can also be attributed to the fact that, our developed NCCSO task scheduling scheme exhibits an intelligent updating of positions while reaching at best solutions.

Table 1 Comparison of execution time and execution cost

Task	Improved Min-Min		MOACO		MOPSO		NCCSO	
	Execution time (s)	Execution cost ($)/hr	Execution time (s)	Execution cost ($)/hr	Execution time (s)	Execution cost ($)/hr	Execution time (s)	Execution cost ($)/hr
20	21.46	4.72	27.54	6.05	26.86	5.90	25.40	5.47
30	39.16	8.61	56.32	12.39	70.29	15.45	53.58	11.78
40	60.44	13.29	87.56	19.26	92.29	20.59	53.92	11.86
50	66.62	14.65	163.49	35.96	147.28	32.40	129.19	28.42
60	209.82	46.15	243.89	53.65	206.50	45.43	162.77	35.80
70	318.81	70.13	312.78	68.10	287.21	63.18	240.12	52.82
80	505.02	111.10	407.54	89.65	334.99	73.69	312.28	68.70
90	768.78	169.13	674.89	148.47	538.09	118.37	331.62	72.94
100	1096.40	241.21	754.90	166.07	663.34	145.93	454.52	99.98

Table 2 Performance improvement rate (PIR%)

	Improved min-min	MOACO	MOPSO	NCCSO
Total execution time	3086.51	2732.94	2366.85	1763.40
PIR (%) over Improved Min-Min		11.455	23.31	**42.87**
PIR (%) over MOACO			13.39	**35.47**
PIR (%) over MOPSO				**25.49**
Total execution Cost	631.84	599.60	520.94	387.77
PIR (%) over Improved Min-Min		5.10	17.55	**38.62**
PIR (%) over MOACO			13.11	**35.32**
PIR (%) over MOPSO				**25.56**

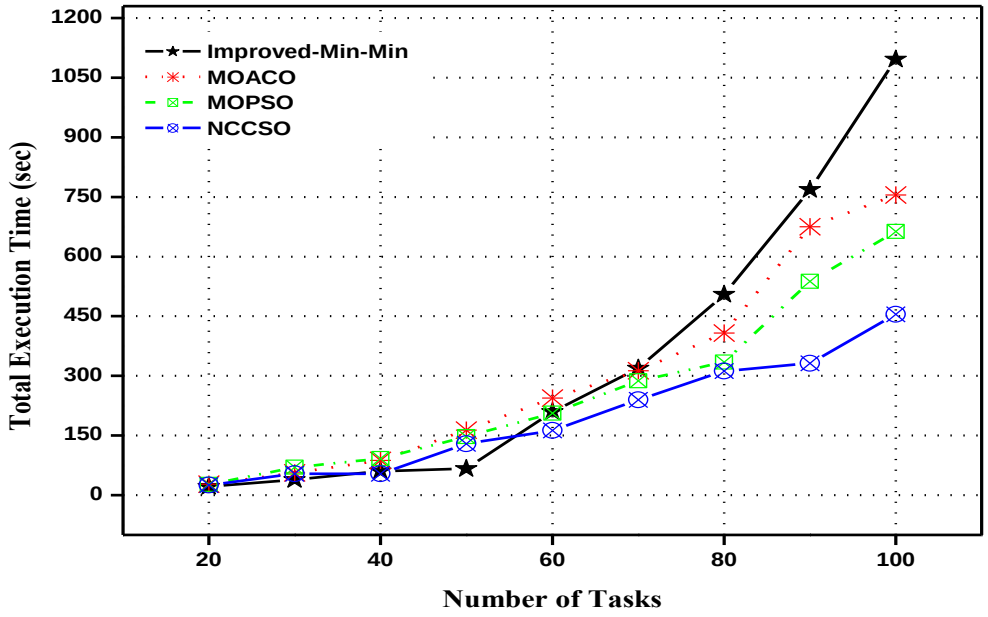

Figure 1 Comparison on the total execution time

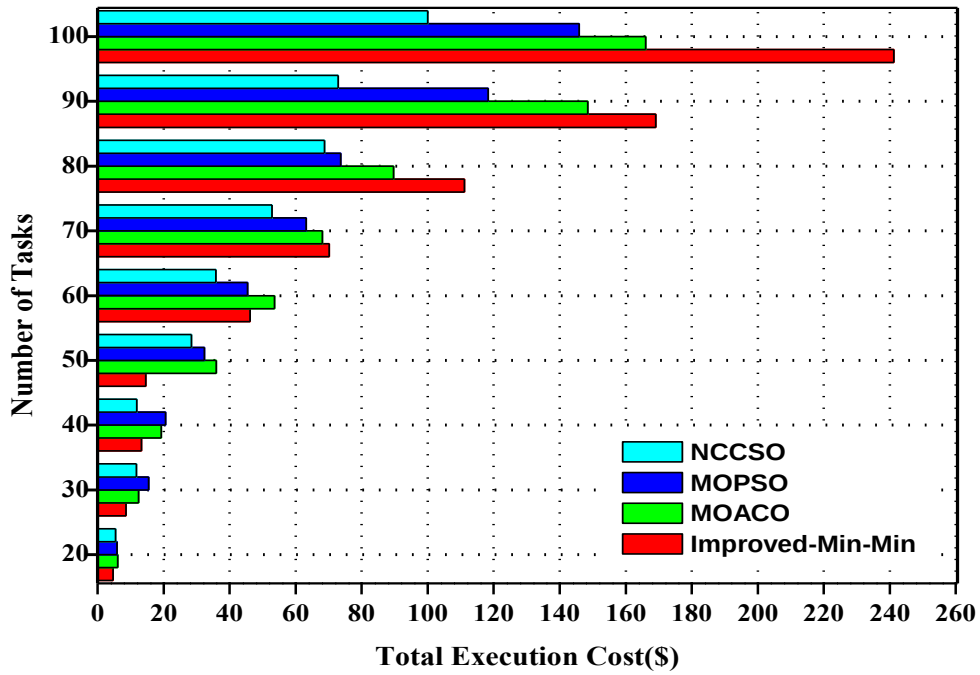

Figure 2 Comparison on the total execution cost

8.Statistical significance based on confidential interval

To further elaborate performance of our developed NCCSO task scheduling scheme, a computed statistical significance on the scheduling schemes based on 95% confidential intervals are provided in *Table 3*. The computed value is derived using Equation (13) [28].

$$\text{Confidential Interval (CI)} = \bar{x} \pm t\frac{s}{\sqrt{n}}, \qquad (13)$$

where, $\bar{x}$ is the mean; t represent the t-distribution that is derived from the t distribution table; s is the standard deviation of the sample data derived after running the task instances on a virtual machine and n represent the number of samples. The computed confidential intervals on the scheduling schemes will help justify how significant our scheme for the scheduling performance in term of tasks execution time. The smaller the value of the confidence intervals shows how precise our estimate compared to the benchmarked schemes. *Table 3* shows the results on 95% confidential interval computed after 20 simulation runs. The results are to show whether execution time obtained by our developed NCCSO scheduling scheme is significantly less compared to the benchmarked schemes for all task instances**.** It has been shown in *Table 3* that; the value obtained by our developed NCCSO is significantly less considering the benchmarked schemes. These means there is a significant difference between the performance of our developed NCCSO scheduling scheme and that of the benchmarked schemes in term of execution time. It can be concluded that our developed NCCSO scheduling scheme can provide better service to the cloud end-users and meet cloud providers satisfactions.

Table 3 Comparison of execution time

	Improved MIN-MIN	**MOACO**	**MOPSO**	**NCCSO**
Degree of freedom	9	9	9	9
Confidence level	0.025	0.025	0.025	0.025
t-distribution	2.262	2.267	2.267	2.262
Mean	342.94	303.21	262.98	195.93
Standard deviation	378.38	264.49	218.04	148.25
Lower bound	72.28	114.02	107.01	90.04

	Improved MIN-MIN	MOACO	MOPSO	NCCSO
Upper bound	613.60	492.40	418.95	254.14
95% confidential interval	(72.28,613.60)	(114.02,492.40)	(107.01,418.95)	(90.04,254.14)

9.Conclusion

In this paper, we unveiled the efficiency of our developed NCCSO scheduling scheme on the CloudSim simulator tool. The obtained results had shown that our developed NCCSO scheduling scheme is promising in providing the minimum execution time and execution cost compared to the benchmarked schemes. This was as a result of incorporation of chaotic process and Pareto-dominance techniques within the local search process of the conventional cat swarm optimization. Causing NCCSO scheduling scheme to overcome global and local imbalanced as well as premature convergence, which led to its best performance in term of the minimum execution time and execution cost. The paper further revealed how significant the performance of the NCCSO scheme. As a result, computation based on 95% confidential intervals to show the statistical significance of the developed NCCSO was revealed. It was concluded that, the developed NCCSO scheduling scheme can meet cloud customers QoS expectations. In the future, we intend to investigate the scalability of the proposed scheme using larger computing workloads.

Acknowledgment

We hereby acknowledged the financial support of the Tertiary Education Trust Fund (TETFund) Nigeria.

Conflicts of interest

The authors have no conflicts of interest to declare.

References

[1] Gui Z, Yang C, Xia J, Huang Q, Liu K, Li Z, et al. A service brokering and recommendation mechanism for better selecting cloud services. PloS one. 2014; 9(8): e105297.

[2] Adebisi AA, Adekanmi AA, Oluwatobi AE. A study of cloud computing in the university enterprise. International Journal of Advanced Computer Research. 2014; 4(15):450-8.

[3] Soni A, Hasan M. Pricing schemes in cloud computing: a review. International Journal of Advanced Computer Research. 2017; 7(29):60-70.

[4] Gabi D, Ismail AS, Zainal A, Zakaria Z. Solving task scheduling problem in cloud computing environment using orthogonal Taguchi-cat algorithm. International Journal of Electrical and Computer Engineering (IJECE). 2017; 7(3):1489-97.

[5] Meena M, Bharadi VA. Performance analysis of cloud-based software as a service (SaaS) model on public and hybrid cloud. In symposium on colossal data analysis and networking (CDAN) 2016 (pp. 1-6). IEEE.

[6] Gabi D, Ismail AS, Zainal A. Systematic review on existing load balancing techniques in cloud computing. International Journal of Computer Applications. 2015; 125(9):16-24.

[7] Zhang Y, Qian C, Lv J, Liu Y. Agent and cyber-physical system based self-organizing and self-adaptive intelligent shopfloor. IEEE Transactions on Industrial Informatics. 2017; 13(2):737-47.

[8] Zuo L, Shu L, Dong S, Zhu C, Hara T. A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. IEEE ACCESS. 2015; 3:2687-99.

[9] Chu SC, Tsai PW. Computational intelligence based on the behavior of cats. International Journal of Innovative Computing, Information and Control. 2007; 3(1):163-73.

[10] Gabi D, Ismail AS, Zainal A, Zakaria Z, Abraham A. Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing. Neural Computing and Applications. 2018; 30(6):1845-63.

[11] Ramezani F, Lu J, Hussain F. Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization. In international conference on service-oriented computing 2013 (pp. 237-51). Springer, Berlin, Heidelberg.

[12] Liu G, Li J, Xu J. An improved min-min algorithm in cloud computing. In proceedings of the international conference of modern computer science and applications 2013 (pp. 47-52). Springer, Berlin, Heidelberg.

[13] Xu Z, Xu X, Zhao X. Task scheduling based on multi-objective genetic algorithm in cloud computing. Journal of Information & Computational Science. 2015; 12(4):1429-38.

[14] Kahejvand V, Pedram H, Zandieh M. Multi-objective and scalable heuristic algorithm for workflow task scheduling in utility grids. Journal of optimization in industrial engineering. 2014; 7(14):27-36.

[15] Pradhan PM, Panda G. Solving multiobjective problems using cat swarm optimization. Expert Systems with Applications. 2012; 39(3):2956-64.

[16] Gabi D, Ismail AS, Zainal A, Zakaria Z, Al-Khasawneh A. Hybrid cat swarm optimization and simulated annealing for dynamic task scheduling on cloud computing environment. Journal of ICT. 2018; 17(3):435-67.

[17] Saule C, Giegerich R. Pareto optimization in algebraic dynamic programming. Algorithms for Molecular Biology. 2015; 10(1):1-20.

[18] Kalyanmoy D. Multi-objective optimization using evolutionary algorithms: an introduction. KanGAL Report. 2011(2011003).

[19] Li X, Xu J, Yang Y. A chaotic particle swarm optimization-based heuristic for market-oriented task-level scheduling in cloud workflow systems. Computational Intelligence and Neuroscience. 2015; 2015:81.
[20] Abdullahi M, Ngadi MA, Dishing SI. Chaotic symbiotic organisms search for task scheduling optimization on cloud computing environment. In ICT international student project conference (ICT-ISPC) 2017 (pp. 1-4). IEEE.
[21] Chang HC, Chen YP, Liu TK, Chou JH. Solving the flexible job shop scheduling problem with makespan optimization by using a hybrid Taguchi-genetic algorithm. IEEE Access. 2015; 3:1740-54.
[22] Liu J, Pacitti E, Valduriez P, De Oliveira D, Mattoso M. Multi-objective scheduling of scientific workflows in multisite clouds. Future Generation Computer Systems. 2016; 63:76-95.
[23] Farahabady MR, Lee YC, Zomaya AY. Pareto-optimal cloud bursting. IEEE Transactions on Parallel and Distributed Systems. 2014; 25(10):2670-82.
[24] Ramezani F, Lu J, Taheri J, Hussain FK. Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments. World Wide Web. 2015; 18(6):1737-57.
[25] Pachorkar N, Ingle R. Affinity aware VM colocation mechanism for cloud. International Journal of Advanced Computer Research. 2014; 4(17):956-60.
[26] Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and experience. 2011; 41(1):23-50.
[27] Eberhart RC, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization. In proceedings of the congress on evolutionary computation. CEC00 2000 (pp. 84-88). IEEE.
[28] Hosmer DW, Lemeshow S. Confidence interval estimation of interaction. Epidemiology (Cambridge, Mass.). 1992; 3(5):452-6.

Danlami Gabi received his Ph.D. degree in Computer Science from Universiti Teknologi Malaysia. He is currently a Lecturer at Kebbi State University of Science and Technology Aliero, Nigeria. His research interests include Complex Algorithm Design for Distributed Systems, Cloud Computing, Mobile Edge Computing and Management beyond the edge.
Email: gabsonley@gmail.com

Nasiru Muhammad Dankolo received his MSc. degree in Computer Science at Universiti Teknologi Malaysia in 2018. He currently serves as an Assistant Lecturer at Kebbi State University of Science and Technology Aliero, Nigeria. His research interests are Machine Learning, Data Mining and Deep Learning.
Email: nasirdankolo@gmail.com

Abdul Samad Ismail received his Ph.D. degree in Computer Science from Aston University, Birmingham UK. His current position is Professor at the Department of Computer Science, School of Computing, Universiti Teknologi Malaysia. His research interests are in Wireless Networking, Cloud Computing, and Network Security.
Email: abdsamad@utm.my

Anazida Zainal received her Ph.D. degree in Computer Science from Universiti Teknologi Malaysia, Skudai Johor, Malaysia. Her current position is Senior Lecturer at Department of Computer Science, School of Computing, Universiti Teknologi Malaysia. Her research interests are in Network Security, Intrusion Detection System, Intrusion Prevention System and Soft Computing.
Email: anazida@gmail.com

Zalmiya Zakaria received her Ph.D. degree in Computer Science from Universiti Teknologi Malaysia, Skudai Johor, Malaysia. Her current position is a Senior Lecturer at the Department of Computer Science, School of Computing, Universiti Teknologi Malaysia. Her research interests are in Optimization in Planning, Scheduling and Timetabling, Machine Learning and Automated Reasoning, Intelligent System Design and Development, Artificial Intelligence Programming (Prolog and LISP).
Email: zalmiyah@utm.my