How to cite this paper:

Gabi, D., Ismail, A S., Zainal, A., Zakaria, Z., & Al-Khasawneh, A. (2018). Hybrid cat swarm optimization and simulated annealing for dynamic task scheduling on cloud computing environment. *Journal of Information and Communication Technology*, *17*(3), 435-467.

HYBRID CAT SWARM OPTIMIZATION AND SIMULATED ANNEALING FOR DYNAMIC TASK SCHEDULING ON CLOUD COMPUTING ENVIRONMENT

[1]Danlami Gabi, [2]Abdul Samad Ismail, [2]Anazida Zainal, [2]Zalmiyah Zakaria & [3]Ahmad Al-Khasawneh

[1]Department of Kebbi State University of Science and Technology, Aliero, Nigeria

[2]Faculty of Computing, Universiti Teknologi Malaysia, Malaysia

[3]Faculty of Prince Al-Hussein bin Abdullah II of Information Technology, *Hashemite University, Zarqa, Jordan*

gabsonley@gmail.com; abdsamad@utm.my; anazida@gmail.com; zalmiyah@utm.my; akhasawneh@hu.edu.jo

ABSTRACT

The unpredictable number of task arriving at cloud datacentre and the rescaling of virtual processing elements can affect the provisioning of better Quality of Service expectations during task scheduling in cloud computing. Existing researchers have contributed several task scheduling algorithms to provide better QoS expectations but are characterized with entrapment at the local search and high dimensional breakdown due to slow convergence speed and imbalance between global and local search, resulting from lack of scalability. Dynamic task scheduling algorithms that can adjust to long-time changes and continue facilitating the provisioning of better QoS are necessary for cloud computing environment. In this study, a Cloud Scalable Multi-Objective Cat Swarm Optimization-based Simulated Annealing algorithm is proposed. In the proposed method, the

Received: 8 August 2017 **Accepted**: 10 April 2018 **Published:** 12 June 2018

. orthogonal Taguchi approach is applied to enhance the SA which is incorporated into the local search of the proposed CSM-CSOSA algorithm for scalability performance. A multi-objective QoS model based on execution time and execution cost criteria is presented to evaluate the efficiency of the proposed algorithm on CloudSim tool with two different datasets. Quantitative analysis of the algorithm is carried out with metrics of execution time, execution cost, QoS and performance improvement rate percentage. Meanwhile, the scalability analysis of the proposed algorithm using Isospeed-efficiency scalability metric is also reported. The results of the experiment show that the proposed CSM-CSOSA has outperformed Multi-Objective Genetic Algorithm, Multi-Objective Ant Colony and Multi-Objective Particle Swarm Optimization by returning minimum execution time and execution cost as well as better scalability acceptance rate of 0.4811−0.8990 respectively. The proposed solution when implemented in real cloud computing environment could possibly meet customers QoS expectations as well as that of the service providers.

Keywords: Cloud computing; multi-objective optimization; task scheduling; cat swarm optimization; simulated annealing.

INTRODUCTION

The evolution of cloud computing has reshaped Information Technology (IT) consumption through the provisioning of high-performance computing as well as massive resource storage that are continually channelled across a medium called the Internet. The paradigm permits the execution of large-scale applications, where distributed collaborative resources which are managed by several autonomous domains are made available (Khajehvand et al., 2014; Gabi, 2014). Trends toward the development of cloud computing have arisen far back when computers are connected and how networking among computers moved to distributed computing, which further led to cluster computing and from cluster computing to grid computing and eventually now, cloud computing (Rani et al., 2015). Presently, services provided by cloud computing are available at affordable cost, with high availability and scalability for all scales of businesses (Hassan et al., 2017). These services include: Software as a Service (SaaS); providing users with opportunities to run applications remotely from the cloud. The Infrastructure as a Service (IaaS); providing virtualize computer services that ensure better processing

power with reserved bandwidth for storage. The Platform as a Service (PaaS); providing operating systems and require services for a particular application (Furkt, 2010; Raza et al., 2015; Cui et al., 2017). All these services function within the delivery model of cloud computing such as Public cloud; that permit dynamic allocation of computing resource over the Internet through web applications. The Private clouds; built to provide full control over data, security as well as the quality of service. The Hybrid cloud; which controls the distribution of applications across both public and private cloud (Furkt, 2010). One of the fundamental challenges of cloud computing is the level of Quality of Service (QoS) satisfaction which has become insufficient to meet consumer and service provider expectations. The number of tasks arriving cloud datacentre are alarming and the recalling of virtual machines processing elements to meet each task expectations is a complex scheduling problem (Ibrahim et al., 2015). The cloud consumers sent tasks to cloud virtual resources (virtual machines). Each task is characterized with QoS objective(s) expected to be met. The cloud consumer demands their submitted task to be processed in a short time with less cost of execution. The service provider facilitates the provisioning of the required service that can meet this expectation while demanding for better pay. This problem can be referred to as a multi-objective NP-hard problem (Kalra & Singh, 2015). It has become necessary to develop task scheduling algorithm that considers dynamicity of cloud computing environment to facilitate efficient mapping of each task on a suitable resource and ordering the task on each resource to satisfy performance criteria (Monika & Jindal, 2016; Kalra & Singh, 2015; Zhang et al., 2014; Letort et al., 2015). Therefore, dynamic optimization algorithms are the potential solutions to distributing tasks amongst virtual machines at run-time as well as considering the current state of Virtual Machine (VM) information on its capacity to fast track next distribution decision (Gabi et al., 2015; Mustaffa et al., 2013; Ibrahim et al., 2016). To date, it is vital to design a low-complexity dynamic optimization algorithm to adapt the dynamicity of cloud tasks and resources while maintaining better QoS performance.

The Swarm Intelligence (SI) techniques are relatively new promising approaches for solving combinatorial optimization problems because of their ability in handling large scale problem and produce results in just one run. These techniques are inspired by the collective intelligence of social behavioural model of insects and other animals (Singh et al., 2017). With the SI technique, sharing of information is done easily among multiple swarms for co-evolution which learn in searching for solution space. The large-scale optimization becomes practical with this technique because it allows multiple agents to be parallelised easily (Singh et al., 2017; Mustaffa et al., 2015).

Some of the examples of SI techniques used by existing researchers to address task scheduling problem are; Particle Swarm Optimization (PSO) (Ramezaini et al., 2013; Awad et al., 2015; Jena, 2015), Ant Colony Optimization (ACO), (Shengjun et al., 2015; Anradha & Selvakumar, 2015), Artificial Bee Colony (ABC) (Kumar & Gunasekaram, 2014; Li & Pan, 2015; Gao et al., 2015), BAT algorithm (Gandomi & Yang, 2014; Jacob, 2014; George, 2015) & Cat Swarm Optimization (CSO) (Bilgaiyan et al., 2015; Gabi et al., 2016).

Cat Swarm Optimization (CSO) is one of the SI approaches introduced in (Chu & Tsai, 2007) to address continuous optimization problem. The technique converges faster than Particle Swarm Optimization (PSO) in terms of speed and convergence (Chu & Tsai, 2007). Exploring this technique to address a discrete optimization problem especially cloud task scheduling problem will be a potential solution. The CSO has both global and local search known as the seeking and tracing mode and a mixed ratio (MR) that determine the position of the cat (Gabi et al., 2016; Chu &Tsai, 2007). Its local search (tracing mode) can be enhanced to search for optimality in a multi-dimensional problem. Simulated Annealing (SA) is a type of local search and is easy to implement probabilistic approximation algorithm, as introduced in (Kirkpatrick et al., 1983) to solve the NP-hard optimization problem (Wang et al., 2016). It uses a neighbourhood function and a fitness function to avoid being trapped at the local optimal, thereby finding a solution closer to global optimum (Jonasson & Norgren, 2016; Abdullahi & Ngadi, 2016; Černý, 1985). The strength of the SA when searching for an optimal solution can as well be enhanced when method like orthogonal Taguchi is introduced (Taguchi et al., 2000). In this study, we proposed a Cloud Scalable Multi-Objective Cat Swarm Optimization based Simulated Annealing (CSM-CSOSA) algorithm to address cloud task scheduling problem in cloud computing. To determine the effectiveness of the algorithm, a multi-objective QoS task scheduling model is presented and solved using the proposed (CSM-CSOSA) algorithm.

Several contributions are made possible in this study, i.e. the development of a Multi-Objective model based on execution time and execution cost objectives for optimal task scheduling on cloud computing environment; the development of CSM-CSOSA task scheduling algorithm to solve the multi-objective task scheduling model; the implementation of the CSM-CSOSA task scheduling algorithm on CloudSim tool; the performance comparison of the proposed CSM-CSOSA task scheduling algorithm with multi-objective genetic algorithm (Budhiraja & Singh, 2014), multi-objective scheduling optimization method based on ant colony optimization (Zuo et al., 2015) and multi-objective particle swarm optimization (Ramezaini et al., 2013) based

on execution time, execution cost, QoS and percentage improvement rate percentage.

RELATED WORK

Several authors have put forward task scheduling optimization algorithms to solve task scheduling problem in cloud computing. Some of which are discussed as follows: Zuo et al. (2015) introduced a multi-objective optimization scheduling method based on an ant colony. The authors' aim is to optimise both the objective of performance and cost. The authors conduct some experiments via simulation to shows the effectiveness of their proposed algorithm. The result of the experiment shows that their method managed to achieve 56.6% increase in the best-case scenario as compared to other algorithms. However, local trapping is an issue regarding the ant colony method as they traverse toward solution finding. The updating process of pheromone can lead to long computation time. Besides, the number of tasks used for the experiment may not be significant enough to justify whether their proposed method is scalable to handle large task size. Similarly, Zuo et al. (2016) proposed a multi-objective task scheduling method based on Ant Colony Optimization (MOSACO). The objective of the study is to address deadline and cost in a hybrid cloud computing environment. The researchers have been able to measure the effectiveness of their proposed MOSACO algorithm using metrics of task completion time, cost, the number of deadline violations, and the degree of private resource utilization.

The results of the simulation show that their proposed MOSACO task scheduling algorithm can provide the highest optimality. However, scalability may be an issue due to the number of tasks used for the experiment, especially when considering the dynamicity of cloud computing. In another development, Dandhwani and Vekariya (2016) put forward a multi-objective scheduling algorithm for cloud computing environments. Their objective is to minimize the execution time and makespan of schedule tasks on computing resources. The authors reported that simulation results of their proposed method can minimize the execution time and makespan time effectively. However, the greedy approach may be insufficient to handle large scale tasks scheduling problem, especially in a dynamic cloud environment. Khajehvand et al. (2013) dwelled on heuristic scalable cost-time trade-off scheduling algorithm for grid computing environments to solve workflow scheduling problem. The study makes use of three scheduling constraints (i.e. the task sizes, task parallelism, and heterogeneous resources) to evaluate their proposed method. The authors revealed that simulation results show that their heuristic method has outperformed the comparison method based on performance and scalability with different workflow sizes. However, the heuristic based

approach can perform better when centralized scheduling environment is considered, where task arrival is known in advance and scheduling is done on the capacity of the virtual machines to handle the task demand. Besides, their performance in a dynamic cloud environment could be an issue due to the volume of tasks and heterogeneity of cloud computing resources. As a result, determining the right resource to execute the task demand will be a very complex decision. In another development, Lakra and Yadav (2015) introduced a multi-objective task scheduling algorithm to increase throughput and minimize resource execution cost. The experimental result via simulation shows that their proposed method can yield better performance in terms of cost and improves throughput. However, its application to address large size tasks in an elastic resource condition is still an issue that needs to be addressed. Yue et al. (2016) presented an improved multi-objective niched Pareto genetic (NPGA) method. The objective of the study is to minimize time consumption and financial cost of handling the users' tasks. The results of the experiment via simulation shows that their proposed algorithms can maintain the diversity and the distribution of Pareto-optimal solutions in cloud tasks scheduling under the same population size and evolution generation than the comparison algorithm. However, long computation time is bound to occur due to mutation process characterised by the genetic algorithm. Besides, the global solution finding merit of the genetic algorithm is insufficient to find an optimal solution due to the nature of its chromosome selection using the probability function. In their part, Budhiraja and Singh (2014) introduced a multi-objective task scheduling algorithm using the genetic technique. The objective of the study is to reduce the cost of execution, execution time and ensured scalability performance. The result of the simulation as stated by the authors shows that their method can obtain a better optimiser in terms of makespan and cost of resource usage. However, it is hard to draw a conclusion on their proposed algorithm, since comparison technique has not been considered.

Hua et al. (2016) presented a PSO-based adaptive multi-objective task scheduling algorithm for cloud computing environment. The objective of their study is to minimize processing time and the transmission time of scheduled tasks in cloud datacentre. The results of the experiment via simulation shows that their PSO-based AMTS algorithm can obtain better quasi-optimal solutions in task completion time, average cost, and energy consumption compared to the genetic algorithm. However, global search process of the PSO is insufficient to handle task scheduling optimization problem without incorporating any local search optimization technique. Besides, the number of iterations used in the experiments is insufficient to justify the performance of the proposed algorithm. On the other hand, Letort et al. (2015) presented a greedy-based scheduling algorithm that handles task scheduling problem based on resource and precedence constraints. The experimental results via simulation show a

significant increase in several numbers of cumulative constraints. However, the greedy approach can perform better when considering small scale network environment with small task size. Leena et al. (2016) proposed a bio-objective task scheduling algorithm based on genetic algorithm for hybrid cloud environments. The objective of the study is to minimize the execution time and execution cost of task schedule on computing resources. The authors make used of two single objective algorithms each for the execution time and execution cost to show the effectiveness of their proposed method. The result of the experiment via simulation shows that their proposed method can reduce the execution time and execution cost of all tasks scheduled on computing resources as compared to the single objective optimization algorithms. However, local entrapment can still be an issue with the genetic technique. Ramezani et al. (2013) introduced a multi-objective algorithm to solve three conflicting objectives; task execution/transfer time and task execution cost. The result of the experiment via simulation on CloudSim tool shows remarkable performance than other comparative algorithms. However, the PSO can easily get entrapped at the local optima region.

Findings from the Existing Method

Findings show that the heuristic (greedy) task scheduling algorithms are applicable to small size scheduling problems. Although some degree of success in addressing the NP-completeness of the scheduling of a task can be achieved by returning a feasible solution, but the dynamic nature of cloud computing environment lags the heuristic approach to satisfy scheduling optimization problems such as makespan and execution cost. The metaheuristic techniques are promising than the heuristic techniques. However, metaheuristic techniques used in the existing literature for multi-objective task scheduling problem exhibits both global and local search optimization process. The global search optimization alone cannot guarantee optimality and local search optimization often gets trapped at the local optimal. Hence, intensification and diversification will generate focus on exploring the search space in a local region using a combination of several methods to help achieve global optimality of both the execution time and execution cost objectives. This will as well increase the scalability to handle the dynamic changing task and resource condition (i.e. the virtual machine processing elements).

METHODOLOGY

Cat Swarm Optimization

Chu and Tsai (2007) introduce Cat Swarm Optimization (CSO) technique which mimics the common behaviour of a natural cat. As observed by the

author, the cats always remain alert while spending most of their time resting and move slowly when observing their environment. Two modes were actualized which represent the behaviour of cat (Gabi et al., 2016) i.e., the seeking mode and the tracing mode. The seeking mode is the global search process of the CSO technique. Four attributes were associated with this mode. The Seeking Memory Pool (SMP); which indicates the memory size sort by the cat, Seeking Range of selected Dimension (SRD); for selecting cat dimensions, Counts of Dimension to Change (CDC); used for disclosing how many dimensions according to cat number varied, and Self-Position Considering (SPC); represents a Boolean variable that unveil if the position at which the cat is presently standing can be chosen as the candidates' position to move into (Gabi et al., 2016). Algorithm 1 shows the procedure for the seeking mode (Chu & Tsai, 2007). The tracing mode is the local search procedure of the CSO technique. Algorithm 2 shows the pseudocode for the CSO tracing mode (Gabi et al., 2016).

Algorithm 1: Pseudocode for CSO seeking mode

Do

1. Generate N copies of cat,
1. Change at random the dimension of cats as per CDC by applying mutation operator
2. Determine all changed cats' fitness values.
3. Discover most suitable cats (non-dominant) based on their fitness values.
4. Replace the position of the *N* cat after picking a candidate at random

***Wh*ile** Stopping condition is not exceeded.

Algorithm 2: Pseudocode for CSO tracing mode

Begin

1. Compute and update cat velocity using the new velocity in Equation 1:

$$V_{K,d} = V_{K,d} + (c_1 \times r_1 \times (Xbest_d - X_{K,d})) \quad (1)$$
$$d = 1, 2, \ldots\ldots, M$$

Where *c*; the constant value of acceleration, *r*; is the uniform distributed random number in the range of [0, 1].

2. Add new velocity by computing the current (new) position of the cat using Equation 2

$$X_{k,d} = X_{k,d} + V_{k,d} \quad (2)$$

3. Calculate the fitness values of all cats.
4. Update and return best cats with the best fitness.

End

Limitations of Cat Swarm Optimization to Solve Cloud Task Scheduling Problem

Although the CSO technique has proven to be more efficient than PSO in both computation time and convergence speed (Chu &Tsai, 2007), its application in

cloud computing may require improvement to solve complex task scheduling optimization problem. The global search optimization process of the CSO is quite promising. However, this global search alone can not guarantee an optimal solution without the support of the local search optimization process. The CSO suffered local entrapment while its global solution finding merit is preserved. This is because the number of cats going into seeking mode (global search) all the time always exceed the ones with tracing mode (local search mode). This may cause the mutation process of the CSO at tracing (local search) mode to affect performance and may end up not achieving an optimal solution for cloud task scheduling optimization problem (Gabi et al., 2016). Similarly, for every iteration, the seeking (global search) mode and tracing (local search) mode of CSO were carried out independently, causing its position and velocity update to exhibit similar process. As a result, a very high computation time is bound to occur (Pradhan & Panda, 2012). Therefore, a local search optimization algorithm incorporated at the local search of the CSO is sufficient to address its limitations.

Simulated Annealing

Simulated Annealing (SA) is a local search probabilistic approximation algorithm introduced by Kirkpatrick et al. (1983). The algorithm uses a neighbourhood and a fitness function to avoid being trapped at the local optima (Jonasson & Norgre, 2016). The SA algorithm often begins with an initial solution X according to some neighbourhood function N with an updated solution X' created . As to how the particle tend to adopt a state which is an improvement over current one, the algorithm generates a solution when the fitness value X^* becomes lower than $f(X)$. However, assume has the higher fitness, it will occasionally be accepted if the defined probability shown in equation 3 is satisfied (Abdullahi & Ngadi, 2016).

$$P_{ro}(X, X^*, T) = \exp[\Big(-\big(f(X^*) - f(X)\big)\Big) * T^{-1}] \quad (3)$$

Where is the fitness evaluation functions and the current solutions of the neighbour accordingly; and represents the control parameter called the temperature. This parameter is determined according to the cooling rate used in (Abdullahi & Ngadi, 2016).

$$T = \sigma^i * T_O + T_{final} \quad (4)$$

Where: = temperature descending rate, ; the number of times which neighbour solutions have been generated so far; initial temperature; final temperature. When the initial value of the temperature is low, the algorithm

becomes limited in locating global optimal solution as the computation time of the algorithm is believed to be shorter (Jonasson & Norgre, 2016; Gabi et al. 2017b). At each iteration performed by the SA algorithm, the comparison between the currently obtained solution and a solution newly selected is carried out. A solution that shows improvement is always accepted (Moschakis & Karatza, 2015). The non-improving solutions are still accepted since there is a possibility that they may escape being trapped at local optima while searching for a global optimal solution. Based on the defined probability in equation 3, the acceptance of the non-improving ones is often determined by the temperature parameter (Nikolaev & Jacobson, 2010). This makes SA algorithm one of the most powerful optimization mechanism.

The basic SA procedure is represented in Algorithm 3.

Algorithm 3: SA pseudocode

INPUT: Initialize Temperature T_o, Final Temperature T_{final}, Temperature change counter $p = 0$, Cooling schedule σ, Number of iteration M_p

OUTPUT: Best Optimum Solution found

1. Generate an initial solution $X \in D$
2. **Repeat**
3. Initialize repetition counter $m \leftarrow 0$
4. Repeat
5. Generate a new solution $X^I \in N$, where N is the neighbourhood of X
6. Compute the P_{ro} according to **Equation 3**
7. If $0 < P_{ro} \ll 0$ decide whether to accept or reject a new solution based on $P_{ro}(X, X^*, T)$
8. Repeat counter $m \leftarrow m + 1$
9. Memorize the optimum solution so far found
10. Until $m = M_p$
11. $p \leftarrow p + 1$
12. **Until** stopping criteria is note exceeded

Limitation of Simulated Annealing to Cloud Task Scheduling

The SA has been regarded as a powerful local search probabilistic algorithm (Abdullahi & Ngadi, 2016), the SA iterates a number of times before finding an optimal or near optimal solution. The repeated number of iteration may affect the computational complexity of the algorithm in solving cloud task

scheduling problem thereby affecting the computational time. Similarly, the SA can get entrapped at the local optimal region especially when the problem size is very large. Its ability to enhance the local search region without the support of the global search may not guarantee optimality(Wang et al., 2016). Therefore, it can be a powerful local search optimization process when combined with a greedy method to overcome its weaknesses.

Orthogonal Taguchi Method

The Orthogonal Taguchi is a greedy-based method developed by Dr. Genichi Taguchi belonging to Nippon telephones and telegraph company in Japan (Gabi et al., 2016). One potential benefit of using the Taguchi method is its ability to solve complex problem while drastically reducing the computation time. The Taguchi method is used to address both single and multi-objective optimization problem (Tsai et al., 2012; Tsai et al., 2013). Taguchi proposed a general formula for establishing an orthogonal array with two levels of *Z* factors using equation 5 (Chang et al., 2015).

$$L_n(2^{n-1}), \tag{5}$$

Where, $n-1$ – symbolizes the column numbers in two-levels orthogonal array; $n = 2k$ – number of experiments corresponding to the n rows, and columns; number of required level for each factor *Z;* k – is a positive integer $(k > 1)$. According to Taguchi, for any column pairs, the combination of all factors at each level occurs at an equal number of times. Algorithm 4 shows the pseudocodes for the Taguchi optimization Method (Gabi et al., 2017a).

Algorithm 4: Taguchi Optimization Algorithm

Begin

1. Select two-level orthogonal array for matrix experiments such that $L_n(2^{n-1}) \forall\, n \geq N + 1$, and N represent task numbers.
2. Generate two sets of velocities $V_{set_{1_{k,d}}}(t)$ and $V_{\mathrm{set}_{2_{\mathrm{k,d}}}}(\mathrm{t})$ according to **Equation (6)**
3. Update the original velocity for every condition according to **Equation (7)**
4. Add new velocity by computing current (new) position of k-th cat using **Equation (8)**
5. Calculate cat fitness using **Equation (18)** such that; $QoS(X) = \theta * TTexe_{cost}(j) + (1 - \theta) * Texec(i,j)$ in accordance with the orthogonal array $L_n(2^{n-1})$.

End

Definition 1.1

Given as the solution search space, let $f: D \rightarrow \Re$ represents an objective function defined in the solution search space. Find $X^* \in D \ni f(X^*) << (X)\ \forall X \in D.$ Where

X is the vector of optimization variables $X = \{x1, x2, ..., x_n\}$ Therefore, each function associated with solution X is an optimal solution X^* that optimizes f.

The Cloud Scalable Multi-Objective Cat Swarm Optimization Based Simulated Annealing

Several swarm intelligence techniques get entrapped at the local optima (Habibi & Navimipour, 2016). The real CSO technique is no different. As rightly highlighted, the CSO has a control variable called the Mixed Ratio (MR) that defines the cat position (seeking or tracing mode). Assume the MR is set to 1, they allow 10% cats into tracing mode (local search) while 90% of the cats move into seeking (global search) mode. The number of cats that goes into seeking mode (global search) all the time always exceed that of tracing mode (local search mode) (Gabi et al., 2016). The mutation process of the CSO at tracing (local search) mode is bound to affect performance and this may end up not achieving an optimal for cloud task scheduling optimization problem (Gabi et al., 2016). Similarly, for every iteration, the seeking (global search) mode and tracing (local search) mode of CSO are carried out independently, causing its position and velocity update to exhibit similar process. As a result, a very high computation time is bound to occur (Pradhan & Panda, 2012). Although the chances of locating the global optima increased at the global search process, it may lose the ability to converge faster at tracing mode and that may have a significant effect to solution finding. Hence, a special mechanism is needed to incorporate in the tracing (local search) mode procedure of the CSO to improve its convergence velocity, scalability, and quality of solution (Abdullahi & Ngadi, 2016). As a powerful local search optimization algorithm, Simulated Annealing (SA) employs certain probability as prevention from being trapped at the local optima. Although it can iterate a number of times after which a near optimal solution can be found. To overcome this, a Taguchi experimental design procedure can be used to enhance its performance by reducing the number of iterations. With the combination of SA and Taguchi method in CSO, a CSM-CSOSA algorithm for scheduling independent non-preemptive task in cloud datacentre for the purpose of ensuring consumers QoS expectations is proposed. The methodology that describes this process is elaborated in the next subsection.

CSM-CSOSA SA Local Search with Taguchi Method

With the proposed CSM-CSOSA algorithm, the tracing (local) search process can now move out of the local optima region (Abdullahi & Ngadi, 2016). To control the performance of parameters of the proposed (CSM-CSOSA) algorithm, the tracing search procedure was further enhanced with the Taguchi

method and simulated annealing. Two sets of candidate velocities $V_{k,d1}(t)$ and $V_{k,d2}(t)$ (Gabi et al., 2016; Gabi et al., 2017a) were generated using the Taguchi method as shown in Equation 6. Details about Taguchi method can be found in (Taguchi et al., 2000). The velocities control the e□ciency and accuracy of the algorithm towards achieving an optimum solution.

$$V_{k,d}(t) = \begin{cases} V_{k,d1}(t) = V_{k,d}(t+1) + (c_1 \times r_1 \times (X_{gbset_d}(t+1) - X_{k,d}(t+1)) \\ V_{k,d2}(t) = V_{k,d}(t+1) + (c_1 \times r_1 \times (X_{lbest_d}(t+1) - X_{k,d}(t+1)) \end{cases} \quad (6)$$

$$d = 1, 2, \ldots\ldots, M$$

Where, $V_{k,d}(t)$ is the velocity of the cat; is the constant value of acceleration, r; is a random number in the range of [0, 1], t; symbolizes the iteration number. A non-dominant velocity among the generated velocities is selected to update the new position of the algorithm using the following rule:

$$V_{k,d}(t) = \begin{cases} V_{k,d1}(t), & if\ (V_{k,d1}) > \left(V_{k,d2}(t)\right) \\ V_{k,d2}(t), & otherwise \end{cases} \quad (7)$$

At each iteration, the comparison between the currently obtained solution and a solution newly selected is carried out. Hence, a solution that improves better is always accepted. The probability of accepting neighbour solution into a new generation of cats using SA is obtained using equation 11 (Abdullahi & Ngadi, 2016). The velocity set with best convergence speed is selected by the CSM-CSOSA algorithm to update the new position of the next cat provided the condition in equation 8 is satisfied (Zuo et al., 2016).

$$X_{k,d}(t) = \begin{cases} V_{k,d}(t) & if\ V_{k,d}(t) \neq 0 \\ r & otherwise \end{cases} \quad (8)$$

Where r; is an integer random number [0,1]. The position of the cat represents the solution of the cat. The cat with the best fitness is stored in an $n \times m$ archive at each run of the algorithm and is compared with the initial best solution in the archive based on dominant strategy. Assume i^{th} and j^{th} represent the positions of two cats in a D-dimensional search space as $X_i = (x_{i2}, x_{i3}, \ldots, x_{id}, \ldots. x_{iD})$ and $X_j = (x_{j2}, x_{j2}, \ldots, x_{jd}, \ldots. x_{jD})$ respectively. A non-dominant strategy is adopted to determine the best fitness when the conditions in equations 9 and 10 are satisfied (Abdullahi and Ngadi, 2016)

$$X_i = \begin{cases} X_i' & if \quad f(X_i') \succ f(X_i) \\ X_i & if \quad f(X_i') \preccurlyeq f(X_i) \end{cases} \quad (9)$$

$$X_j = \begin{cases} X_j' & if \quad f(X_j') \succ f(X_j) \\ X_j & if \quad f(X_j') \preccurlyeq f(X_j) \end{cases} \tag{10}$$

Where $f(.)$ denotes the fitness evaluation function. If the fitness value $f(X_i')$ is better than that of the $f(X_i)$. For minimization process, the new fitness is accepted for an update with the probability defined in equation 11.

$$P_{ro}(X, X', T) = \exp\left[\left(-\left(f(X_i') - f(X_i)\right)\right) * T^{-1}\right] \tag{11}$$

Where $f(X_i')$ and $f(X_i)$ denotes fitness functions of the cat and current solutions, represents the control parameter which is the temperature. The CSM-CSOSA algorithm is illustrated in Algorithm 5.

Algorithm 5: Proposed CSM-CSOSA Algorithm

Begin:

1. **Input**: *Initialize* cat parameters: create population of the cats as $X_i \; \forall i = \{1,2,3 \ldots . . n\}$, *initialize* X_i, *flag number, Initialize SA parameters: initial Temperature* T_O, *final Temperature* T_{final}, *rate of cooling* α.
2. ***Generate*** *an empty non-dominant archive of (n × m) size of uniform random number [0, 1]*
3. **Output:** Best solution with minimum total execution time and minimum total execution cost.
4. Identify the best optimal solution for the trade-off values as $X_{gbest} \in D \forall \boldsymbol{D} = \{dimension\}$
5. **Do** // apply **seeking mode process** to evaluate cat fitness.
   {
6. **If (flag← 1)**
7. **Execute** tracing mode process according to Algorithm 1.
8. Discover the X_{gbest} solution
9. **If** (X_{gbest} is not improved)
10. ***Else***
11. //*** apply **tracing mode process** to find the X_{gbest} using SA and Taguchi method***//
12. ***Call……. Algorithm 3*** *to execute the* ***SA Method***
    {
13. ***While*** ($T_{final} > T_o$) do
14. ***Call ……. Algorithm 4*** *to execute the* ***Taguchi method***
    {
15. Generate new solution X_i' in the neighbourhood of X_i using Equation 7 and Equation 8
16. Compute $f(X_i, X_i^i)$
17. $f = f(X_i') - f(X_i)$

(continued)

18. **If** $f \leq 0$ *or* $\exp(-fT^{-1}) > rand(0,1)$
    // rand (0, 1) is a uniformly random generated number between 0 and 1
19. Apply new fitness selection strategy based on Pareto dominance according to Equation 9 &10
20. Reduce the temperature using Equation 4
21. $X_i \leftarrow X_i'$,
22. **Endif**
    }
23. **Endwhile**
    }
24. **Endif**
    }
25. **While** condition is not reached.
26. Output optimization solution for the execution time and execution cost.

End.

Problem Description

In cloud computing, the attributes associated with the task scheduling problem are Cloud Information System (CIS), Cloud Broker (CB) and Virtual Machines (VMs). The tasks are referred to as cloudlets in cloud computing. The CIS receives cloudlets $\{c_1, c_2, c_3, \ldots \ldots .., c_n\}$ from the cloud consumers which are sent to CB. A query is generated from CIS−CB in each datacenter n the required service to execute the received cloudlets. Assume $\{v_1, v_2, v_3, \ldots \ldots ., v_m\}$ represent heterogeneous VMs (which varies in capacity in both speed and memory) for executing each cloudlet, then the time a cloudlet spends executing on VMs will determine the total cost per time quantum on all VMs. Therefore, the following assumptions are considered necessary for the scheduling: (i) two datacentres are considered sufficient for the task schedule; (ii) The two datacentres belong to the same service provider; (iii) Transmission cost is ignored; (iv) Cloudlets are dynamically assigned to VMs where each VM handles at most one cloudlet at a time and the total number of all possible schedules is considered to be $(n!)^m$ (Zuo et al., 2015) for the problems with n cloudlets and m VMs; (v) Pre-emptive allocation policy is not allowed; (vi) The cost of using VMs for a time quantum varies from one to another per hour (/hr). Hence, the Expected Time to Compute (ETC) and the Expected Cost to Compute (ECC) matrix will be used for the scheduling decision.

The modelling of the execution time and execution cost objective is as follows: Let $C_i \forall i = \{1, 2, \ldots ., n\}$ denotes set of cloudlets that are independent of one to the other schedule on virtual machine $V_j \forall j = \{1, 2, \ldots ., m\}$. The total execution time T_{exeij} for all cloudlets executed on V_j can be calculated using Equation 12 and the execution time of cloudlets $C_i \forall i = \{1, 2, \ldots ., n\}$ on is computed using equation 13.

$$T_{exe_{ij}} = \sum_{j=1}^{m} exe_{ij} \tag{12}$$

$$exe_{ij} = \sum_{i=1}^{n} x_{ij} * \frac{C_i}{npe_j \times V_{mips_j}}, \tag{13}$$

Such that:

$$x_{i,j} = \begin{cases} 1, & if\ cloudlet\ i\ is\ assigned\ to\ VM_j \\ 0, & otherwise \end{cases} \tag{14}$$

Where, exe_{ij} is the execution time of running cloudlets on one virtual machine; C_i is the set of cloudlets in Millions Instruction (MI) assigned on the virtual machine V_j; V_{mips_j} is the virtual machine speed in Million Instructions per Seconds (MIPs); is the number of the processing element (Gabi et al., 2016). Equation 15 is used to compute the cost of executing all cloudlets on all V_j if and only if the cost of a virtual machine per time quantum is given per hour (/hr) (Ramezani et al., 2013) while equation 16 computes the cost of executing cloudlets on V_j .

$$TTexe_{cost_{ij}} = \sum_{j=1}^{n} exe_{cost_{ij}} \tag{15}$$

Where $TTexe_{cost_{ij}}$ is the total cost of executing all cloudlets on V_j, $exe_{cost_{ij}}$ is the cost of executing cloudlets on V_j (Ramezaini et al., 2013).

$$exe_{cost_{ij}} = Vcost_j * \left(\frac{1}{3600} * \sum_{i=1}^{n} x_{i,j} * \frac{C_i}{npe_j * V_{mips_j}} \right) \tag{16}$$

$Vcost_j$, is the monetary cost of one unit V_j in US dollar per hour. A mathematical model for the multi-objective task scheduling problem can be expressed as follows:

$$Min\ F(X) = \{(Texe_{ij}, TTexe_{cost_{ij}}\}$$

Subject to:

$$\sum_{j=1} x_{ij} = 1, \ \forall\ i = 1,2, \ldots \ldots, n; \ \ x_{ij} \in \{0,1\}, \forall i, j \tag{17}$$

The fitness for the QoS when the trade-off factors for the time and cost for consumer service preference can be expressed as follows (Zuo et al., 2015; Beegom & Rajasree, 2015).

$$QoS(X) = \theta * TTexe_{cost_{ij}} + (1 - \theta) * Texe_{ij}$$
$$\forall\{(TTexe_{cost}, Texec) \in Archieve\} \quad (18)$$

Where, θ[0,1] is the control factor for selection of consumer service preference based on time and cost objectives.

Evaluation Metrics

The metrics used for evaluation are execution time, execution cost using the model presented in equation (12) and (15) and the QoS (fitness) model in equation (18) as well as the statistical analysis based on percentage improvement rate percentage (PIR%) using equation (19).

$$PIR\% = \frac{Execution\ time\ (other\ scheme) - Execution\ time\ (CSM-CSOSA)}{Execution\ time\ (other\ scheme)} \quad (19)$$

RESULTS AND DISCUSSION

The CloudSim simulator tool (Buyya et al., 2010) is used for the experiment. The CloudBroker policy of the CloudSim is used to implement the algorithm and run with two (2) different datasets. The settings for each algorithm are shown in Table 1. The comparison with multi-objective task scheduling algorithms discussed in the introduction were used, i.e. the Multi-Objective Genetic Algorithm (Budhiraja & Singh, 2014), Multi-Objective scheduling method based on Ant Colony Optimization (Zuo et al., 2016) & Multi-Objective Particle Swarm Optimization (Ramezaini et al., 2013).

Table 1

The parameter setting for the four task scheduling algorithms

Algorithm	Parameter	Value
MOPSO	Particle size	100
	Self-recognition coefficients (c1, c2)	2.0
	Uniform random number (*R1)*	[0,1]
	Maximum iteration	1000
	Variable Inertia weight(*W*)	90-40%
CSM-CSOSA	Cat size	100

(continued)

Algorithm	Parameter	Value
	Count Dimension to Change	5%
	Self-recognition coefficients (c1)	2.0
	Uniform random number (*R1)*	[0,1]
	Maximum iteration	1000
	Mixed ratio	2%
	Initial Temperature	10
	Final temperature	0.001
	Cooling rate	0.9
MOSACO	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
MOGA	Population size	100
	Maximal iteration	1000
	Crossover rate	0.5
	Mutation rate	0.1

The parameter setting for the datacentres as shown in Table 2 were based on Gabi et al. (2016) and, Abdullahi and Ngadi (2016).

Table 2

Parameter Settings for the Cloud Computing Datacentre

DATACENTER	Parameter	Values
	No. of datacentre	2
	No. of host in a datacenter	1
HOST	Host RAM	2GB
	Storage	ITB
	Bandwidths	10GB/s
	Accumulated host processing power	1000000 MIPS
CLOUDLETS	No. of Cloudlets	[100-1000]
	Lengths	[100, 1000] MIs
	File size	[200, 400] MB
	Output size	[300]
VIRTUAL MACHINE	VM id	[1-20]
	VMs Monitor	Xen

(continued)

DATACENTER	Parameter	Values
	Accumulated Ram	0.5GB
	Accumulated Storage	10GB
	Bandwidth	1GB/s
	VMs processing power	1000-10000 MIPS
	Number of processing elements	1 to 4
		5 to 50
	VM Policy	Time-shared
	Cost per unit VM	[0.17−1.25$/hour]
	Cost of using memory	0.05$/hour

The performance of the proposed CSM-CSOSA (on minimization of task execution time and execution cost) with the variation of its control parameters for consumer service selection preference is evaluated. The results are compared for the objective of execution time and execution cost as an extremely critical parameter for consumer QoS for varying number of tasks namely 100-1000 respectively. These experiments on two benchmark datasets, i.e., the normal distributed dataset & the HPC2N dataset (Abdullahi & Ngadi, 2016) where the experimental results are compared with three task scheduling algorithms (MOGA, MOSACO & MOPSO). Therefore, each algorithm runs 30 simulation times and the average value is taken as the comparison. In Tables 3 and 4, the conducted experiment shows the effectiveness of scheduling algorithms. The result of the experiments is summarized via an average value for a total of 30 simulation runs.

Table 3

Comparison on Execution time(sec) and Execution Cost(/hr)− Normal distributed dataset

Task	MOGA		MOSACO		MOPSO		CSM-CSOSA	
	Execution time	Execution cost	Execution time	Execution cost	Execution time	Execution cost	Execution time	Execution cost
100	6058.27	1332.82	88284.51	19422.59	4948.99	1088.77	4044.54	889.79
200	7059.25	1553.04	97392.78	21426.41	19383.69	4264.41	15879.07	3493.39
300	93137.01	20490.14	98499.76	21669.95	42941.44	9447.11	35153.95	7733.86
400	124651.76	27423.39	109194.01	24022.68	76178.27	16759.22	62431.91	13735.02
500	134063.75	29494.03	109298.76	24045.73	118307.78	26027.71	97050.19	21351.04
600	191371.01	42101.62	117094.76	25760.84	173652.70	38203.59	142451.62	31339.35

(continued)

Task	MOGA		MOSACO		MOPSO		CSM-CSOSA	
	Execution time	Execution cost	Execution time	Execution cost	Execution time	Execution cost	Execution time	Execution cost
700	223455.51	49160.21	183455.75	40360.26	234024.61		192413.25	42330.91
800	325969.03		323662.26	71205.69	308448.97		253340.68	55734.95
900	510860.71		425393.01	93586.46	389884.04		320557.41	70522.63
1000	707862.76		631498.76	138929.72	483529.84		397058.65	87352.90

Table 4

Comparison on Execution time(sec) and Execution Cost− HPC2N dataset

Task	MOGA		MOSACO		MOPSO		CSM-CSOSA	
	Execution time	Execution cost	Execution time	Execution cost	Execution time	Execution cost	Execution time	Execution cost
100	2149.55	472.90	8065.18	1774.33	852.106	187.46	1193.18	262.5
200	3176.66	698.86	8426.87	1853.91	2757.82	606.72	2893.53	632.57
300	4519.46	994.28	11721.81	2578.79	4847.71	1066.49	6975.66	1534.64
400	13868.47	3051.06	21323.50	4691.17	8261.82	1817.60	10411.32	2290.49
500	27376.30	6022.78	23722.67	5218.98	11905.35	2619.17	11458.76	2520.92
600	32964.88	7252.27	26875.12	5912.52	16063.94	3534.06	12764.87	2808.27
700	34756.75	7646.48	27772.62	6109.97	19253.01	4235.66	14023.48	3085.16
800	38861.56	8549.55	29721.44	6538.72	24622.87	5417.033	18513.75	4073.02
900	43382.02	9544.19	30152.42	6633.53	28902.72	6358.59	23278.9	5121.35
1000	45182.42	9940.13	36228.91	7970.36	27301.05	8206.23	26867.23	5910.79

According to this average value illustrated in Tables 3 and 4 precisely, it is clear that for the execution time and execution cost multi-objectives, the proposed CSM-CSOSA algorithm has balanced both the total execution time and total execution cost as consumer requirement which makes it superior compared to MOGA, MOSACO and MOPSO. In both tables (3 & 4) based on the two different datasets used, it can be seen that for CSM-CSOSA task scheduling, the execution time and execution cost spent to complete tasks

is very much minimal as compared to the execution time and execution cost spent to complete the tasks with MOGA, MOSACO and MOPSO algorithm. It is shown that the execution time obtained has an influence on the cost performance.

Moreover, to have a better sense of the performance of the algorithms, some figures are illustrated to show the performance of the algorithms more explicitly. As the task keeps increasing from 100-1000, all the four scheduling algorithms increase in terms of execution time and execution cost. Figures 1-4 are plotted for execution time and execution cost based on the two different datasets used respectively. According to these figures, as the number of tasks keep increasing, both the execution time and the execution cost increase as well. On the execution time and execution cost minimization, the proposed CSM-CSOSA task scheduling algorithm has a better operation and outperforms the MOGA, MOSACO and MOPSO task scheduling algorithms. The increase in task size and the performance obtained also show that the proposed CSM-CSOSA is scalable as well as capable of scheduling huge tasks with the lowest execution time in the heterogeneous environment. However, it also confirms that the CSM-CSOSA algorithm has shown to increase its quality of solutions by balancing task on the best virtual machine with minimum execution time and execution cost.

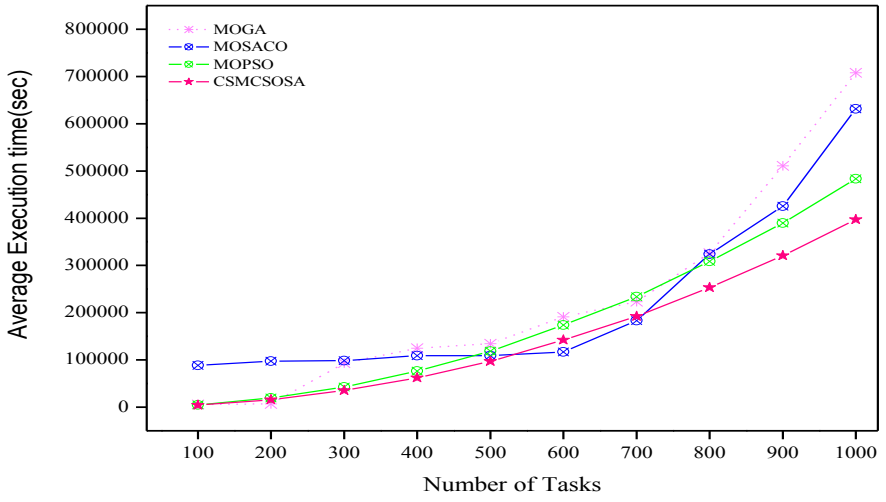

Figure 1. Average execution time-Normal distribution dataset.

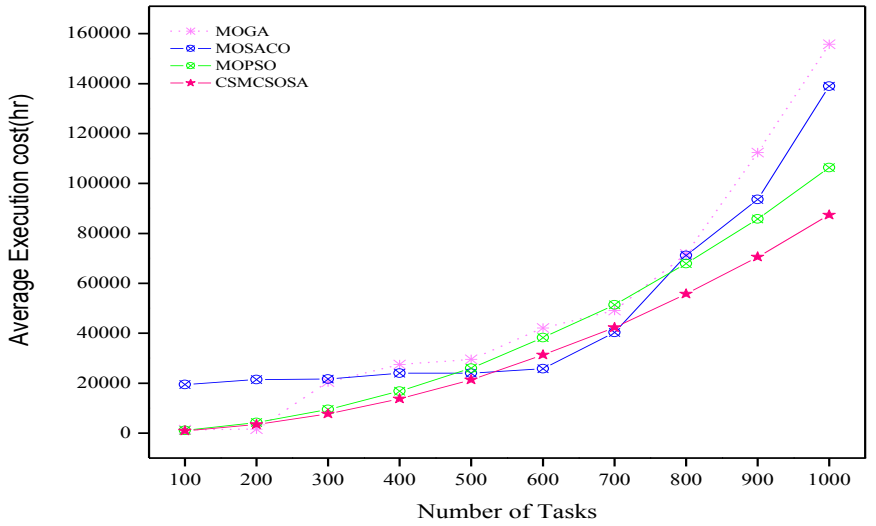

Figure 2. Average execution cost-Normal distribution dataset.

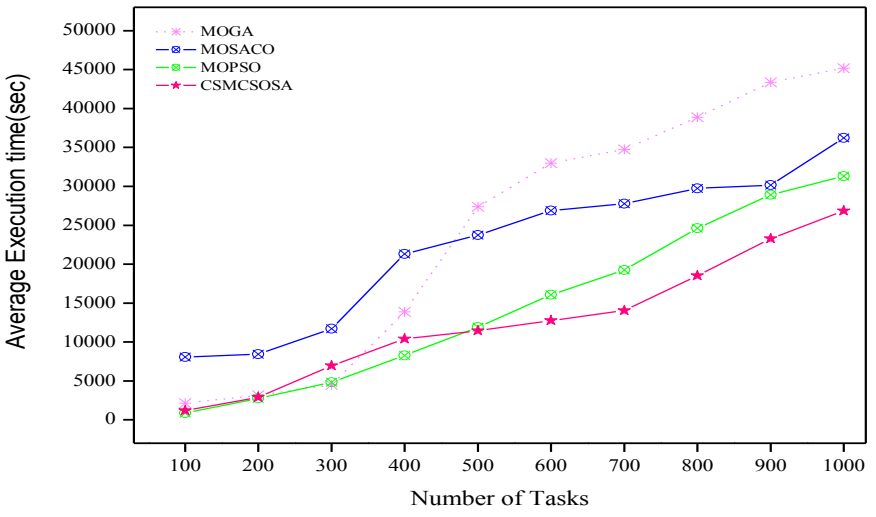

Figure 3. Average execution time-HPC2N dataset.

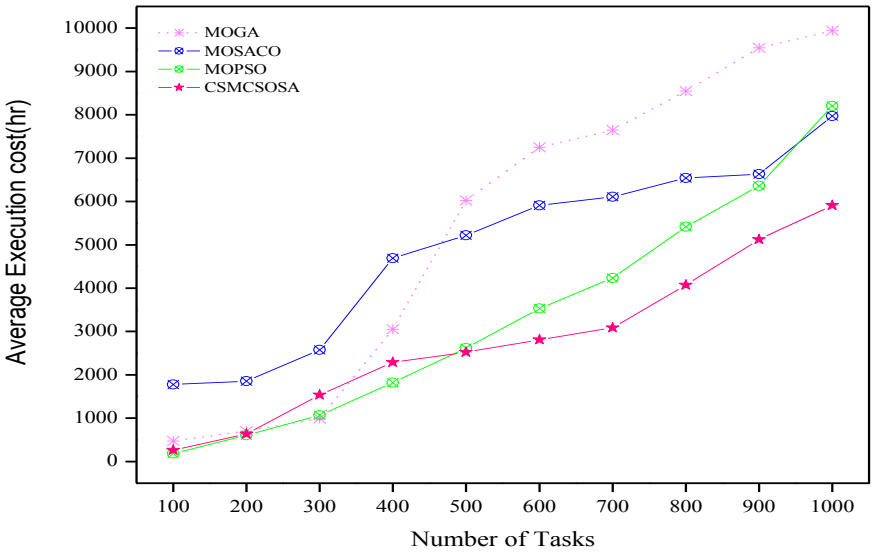

Figure 4. Average execution cost-HPC2N dataset.

In addition, the fitness (QoS) function formula in equation (18) is used to guide the target to optimize the performance of global best in the CSM-CSOSA algorithm and the MOGA, MOSACO and MOPSO task scheduling algorithm. The result is shown in Table 5. In all cases, the CSM-CSOSA shows the best performance.

Table 5

Comparison on Estimated total QoS Minimized

Datasets	**MOGA**	**MOSACO**	**MOPSO**	**CSM-CSOSA**
Normal distributed dataset	316359.91	218444.14	185186.51	152009.92
HPC2N Workload	24631.40	22407.86	15480.64	12842.05

In Table 6, the improvement of the proposed CSM-CSOSA algorithm over the three comparative scheduling algorithms using the normal distributed dataset shows our proposed algorithm has managed to achieve 34.59%, 30.37% and 17.87% in terms of total average execution time. A similar analysis is reported using the HPC2N dataset where the result is shown in Table 7 in

Table 6

Comparison on Improvement (%) based on Execution Time − Normal Distributed Dataset

	MOGA	MOSACO	MOPSO	CSM-CSOSA
Total Average execution time	2324489.06	2183774.36	1851300.33	1520381.27
PIR% over MOGA		6.05	20.35	34.59
PIR% over MOSACO			15.22	30.37
PIR% over MOPSO				17.87

Table 7

Comparison on Improvement (%) based on Execution Time −HPC2N Dataset

	MOGA	MOSACO	MOPSO	CSM-CSOSA
Total Average execution time	246238.07	224010.54	144768.40	128380.68
PIR% over MOGA		9.03	41.21	47.86
PIR% over MOSACO			35.37	42.68
PIR% over MOPSO				11.31

terms of execution time. In Table 7, the performance improvement achieved by the four scheduling algorithms is reported. The result of the analysis shows the CSM-CSOSA which was able to reduce the execution time by obtaining 47.86%, 42.68% and 11.31% compared to MOGA, MOSACO and MOPSO. The performance recorded by our proposed algorithm is due to the combination of Simulated Annealing (SA) and the Taguchi approach which is incorporated at the local search of the CSM-CSOSA that guides the algorithm toward position updating without affecting the computational complexity. This approach also helps our proposed algorithm returns local best solution as fast as possible which is also attributed to the significant choice of velocity. The CSM-CSOSA has shown to improve its quality of solutions at the latter stage of search procedure, making it more efficient for cloud task scheduling (Gabi et al., 2018).

Scalability Analysis of the Scheduling Algorithms

To further unveil performance of our proposed CSM-CSOSA task scheduling algorithm together with the three comparative algorithms, a scalability analysis

is conducted. This process enables us to gain insight on the scalability of the proposed algorithm towards scaling with large workloads and the changes on the number of virtual processing elements (Chen et al., 2008). Kumar and Kao (1987) put forward measuring criteria known as Isoefficiency metric to account for the scalability of a system. In the context of cloud computing, scalability can be seen as an algorithm-Virtual Machine (VM) combination. According to Sun and Rover (1994), scalability of an algorithm in relation to VM combination is when an average execution time exhibited remain constant even when a re-scaled in processing element and problem size occurs. Hence, considering the heterogeneity of cloud computing resources, an Isospeed-efficiency scalability metric $\psi(C, C')$ proposed in (Chen et al., 2008) for calculating the scalability of an algorithm based on machine dependance is adopted for the scalability investigation. In this study, the expected value for the scalability performance is considered to be in the ranges $0 < \psi < 1$. The Isospeed-efficiency scalability function $\psi\ (C, C')$ for computing the scalability is illustrated in equation 20 (Chen et al., 2008).

$$\psi(C, C') = \frac{C' * W}{C * W'} \tag{20}$$

Where, C^I is the initial execution time achieved by the algorithms based on configured number of processing elements on virtual machines, C is the scaled execution time when the processing element increases on virtual machine, W is the initial workload (tasks) assigned on virtual machine, W' is the rescaled workload (tasks) assigned on virtual machines. To compute the scalability of the proposed algorithm, one Parallel Workload, i.e., the HPC2N dataset with 527, 371 tasks were considered and 5000−14000 tasks instances drawn from the workload were used in the experiment. Processing elements from 5−50 are assigned to virtual machines. The results associated with each algorithm based on the obtained execution time is shown in Table 8, while computed scalability performance is reported in Table 9. The scalability computation for each algorithm is carried out using the following example for MOGA algorithm shown in equation 21.

$$\psi(C, C') = \frac{C' * W}{C * W'} = \frac{60749.54 * 8000}{734420.83 * 5000} = 0.1323 \tag{21}$$

Table 8

Average Total Execution Time(sec)−HPC2N Dataset

Workloads	Configured processing elements on VM0.195	MOGA	MOSACO	MOPSO	CSM-CSOSA
5000	**5**	60749.54	68224.11	98218.37	85691.89

(continued)

Workloads	Configured processing elements on VM0.195	MOGA	MOSACO	MOPSO	CSM-CSOSA
8000	**20**	734420.83	836672.18	403092.86	284977.74
10000	**30**	1908232.46	920394.46	611319.66	509939.30
12000	**40**	2021129.03	1864725.72	1909903.79	808999.24
14000	**50**	3490413.07	3143443.18	2077310.12	2020078.67

Table 9

Computed Scalability with− HPC2N Dataset

Scalability	Configured processing elements on VM	MOGA	MOSACO	MOPSO	CSM-CSOSA
	(5,20)	0.1323	0.1878	0.3898	0.4811
	(20,30)	0.4810	1.1363	0.8242	0.6986
	(30,40)	1.1329	1.0373	0.9287	0.8630
	(40, 50)	0.6755	1.0864	0.9238	0.8990

In the aforementioned Table 9, the proposed CSM-CSOSA algorithm is able to maintain better scalability performance by returning an acceptable value of 0.4811, 0.6986, 0.8630, 0.8990 for the HPC2N dataset compared to that of MOGA, MOSACO and MOPSO task scheduling algorithms. These values, however, shows that the proposed algorithm can respond to the dynamic changing cloud task and resource condition than the comparative algorithms under consideration.

CONCLUSION

The unpredictable number of task arriving at cloud datacentre and the rescaling of virtual machine processing elements during task scheduling affects the provisioning of better QoS expectations. Dynamic task scheduling algorithms are considered to be effective for addressing this kind of problem but are truly complex to develop. Previous authors have contributed immensely through the provision of several task scheduling algorithms but at the expense of scalability. In this study, we introduce Cloud Scalable Multi-Objective Cat Swarm Optimization based on Simulated Annealing (CSM-CSOSA) that considers the dynamicity of cloud computing environment to improve better QoS. The effectiveness of the algorithm is evaluated using a multi-objective model for the time and cost criteria. The novelty of the proposed method is

based on the use of SA and Taguchi method that enhance the local search procedure of the algorithm in exploring larger search space which eventually yield better optimum solutions. Comparison of the performance of CSM-CSOSA with some of the existing metaheuristics (MOPSO, MOSACO and MOGA) task scheduling algorithms is carried out with one dataset and one parallel workload. The results obtained shows that the proposed method has achieved a remarkable performance by returning good QoS as well as better scalability performance with 0.4811, 0.6986, 0.8630 and 0.8990 compared to the comparative algorithms. In the future, the study aims to look at privacy aware scheduling in such a way that protects the sensitive information associated with tasks.

ACKNOWLEDGEMENT

This work was sponsored by the Nigerian Tertiary Education Trust Fund (TETFund) in collaboration with Kebbi State University of Science and Technology Aliero, Nigeria.

REFERENCES

Abdullahi, M., & Ngadi, M. A. (2016). Hybrid symbiotic organisms search optimization algorithm for scheduling of tasks on cloud computing environment. *PLoS ONE, 11*(6), e0158229. doi: 10.1371/journal.pone.0158229, 2016.

Abubaker, A., Baharum, A., & Alrefaei, M. (2016). Multi-Objective particle swarm optimization and simulated annealing in practice. *Applied Mathematical Sciences*, *10*(42), 2087 – 2103.

Anuradha, M., & Selvakumar, S. (2015). ACO based task scheduling algorithm for hybrid cloud. *International Journal of Emergence Technology in Computer Science & Electronics (IJETCSE),13*(1), 373-377.

Awad, A. I., El-Hefnawy, N. A., & Andel_Kader, H. M. (2015). Enhanced particle swarm optimization for task scheduling in cloud computing environments. *Procedia Computer Science*, *65*, 920-929.

Beegom, A. A. & Rajasree, M. (2015). Genetic algorithm framework for bi-objective task scheduling in cloud computing systems. In R. Natarajan, G. Barua, Patra, M. R. (Eds.), Lecture notes in Computer Science (pp. 356-359). London, UK: Springer

Bilgaiyan, S., Sagnika, S., & Das, M. (2015). A multi-objective cat swarm optimization algorithm for workflow scheduling in cloud computing environment. *308*, 73–84.

Budhiraja, S., & Singh, D. (2014). An efficient approach for task scheduling based on multi-objective genetic algorithm in cloud computing environment. *IJCSC*, *5*(2), 110−115.

Buyya, R., Calheiros, R. N., Ranjan, R., Beloglazov, A., & De Rose, C. A. F. (2010). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software-Practice and Experience*, *41*(1), 23-50.

Chen, Y., Sun, X.-H., & Wu, M. (2008). Algorithm-system scalability of heterogeneous computing. *Journal Parallel Distribution Computer*, *68* (2008), 1403–1412.

Chang, H.-C., Chen, Y.-P., Liu, T.-K. & Chou, J.-H. (2015). Solving the flexible job shop scheduling problem with makespan optimization by using a hybrid Taguchi-Genetic algorithm. *IEEE Journals & Magazines*, (3), 1740-1754.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications, 45(1), 41–51*.

Chu, S.-C., & Tsai, P.-W. (2007). Computational intelligence based on the behavior of cats. *International Journal Innovation Computer Information Control*, *3*(2007), 163–173.

Cui, H., Liu, X., Yu, T., Zhang, H., Fang, Y., & Xia, Z. (2017). Cloud service algorithm research and optimization. *Security and Communication Networks,*1-7.

Dandhwani, V., & Vekariya, V. (2016). Multi-Objective task scheduling using K-mean algorithm in cloud computing. *International Journal of Innovative Research in Computer and Communication Engineering*, *4*(11), 19521−19524.

Furht, B. (2010). Cloud Computing Fundamentals. Furht, B., & Escalante, A. (Eds.). *Handbook of cloud computing*. Springer New York Dordrecht Heidelberg London, 3-19. Springer.

Gabi, D., Ismail, A.S., Zainal, A., Zakaria, Z., & Abraham, A. (2016). Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing. *Neural Computer & Application,* 1-19. doi:10.1007/s00521-016-2816-4

Gabi, D., Ismail, A.S., & Zainal, A. (2015). Systematic review on existing load balancing techniques in cloud computing. *International Journal of Computer Applications*, *125*(9), 16-24.

Gabi, D., Ismail, A.S., Zainal, A., & Zakaria, Z. (2017a). Solving task scheduling problem in cloud computing environment using Orthogonal Taguchi-Cat algorithm. *International Journal of Electrical and Computer Engineering (IJECE)*, *7*(3),1489-1497.

Gabi, D., Ismail, A. S., Zainal, A. Zakaria, Z. & Al-Khasawneh, A. (2017b). Cloud scalable multi-objective task scheduling algorithm for cloud computing using cat swarm optimization and simulated annealing. *Proceedings of the 8th International Conference on Information Technology (ICIT)*. 17-18 May. Amman, Jordan, 1007-1012.

Gabi, D., Ismail, A.S., Zainal, A., & Zakaria, Z. (2018). Quality of service (QoS) task scheduling algorithm with Taguchi orthogonal approach for cloud computing environment. In Saeed F., Gazem N., Patnaik S., Saed Balaid A., Mohammed F. (Eds.), *Recent Trends in Information and Communication Technology. IRICT 2017*. *Lecture Notes on Data Engineering and Communications Technologies*, *5*, Springer, Cham, 641-649.

Gabi, D. (2014). Surveillance on security issues in cloud computing: A view on forensic perspective. *International Journal of Scientific & Engineering Research*, *5*(5), 1246-1252.

Gandomi, A. H., & Yang, X. S. (2014). Chaotic bat algorithm. *Journal of Computational Science*, *5*(2), 224-232.

Gao, K. Z., Suganthan, P. N., Chua, T. J., Chong, C. S., Cai, T. X., & Pan, Q. K. (2015). A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion. *Expert Systems with Applications*, *42*(21), 7652-7663.

George, S. (2015). Hybrid PSO-MOBA for profit maximization in cloud computing. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *6*(2), 159–163.

Habibi, M., & Navimipour, N. J. (2016). Multi-Objective task scheduling in cloud computing using an imperialist competitive algorithm. *(IJACSA) International Journal of Advanced Computer Science and Applications*, *7*(5), 289−293.

Hassan, H., Nasir, M. H. M., Khairuden, N., & Adon, I. (2017). Factors influencing cloud computing adoption in small and medium enterprises. *Journal of Information and Communication Technology*, *16*(1), 21–41.

Hua, H., Guangquan, X., Shanchen, P., & Zenghua, Z. (2016). AMTS: Adaptive multi-objective task scheduling strategy in cloud computing. *China Communications*, *13*(4), 162–171.

Ibrahim, M., Ibrahim, H., Abdullah, A., & Latip, R. (2016). A high performance UCON And Semantic-Based Authorization Framework for Grid Computing. *Journal of Information and Communication Technology*, *15* (1), 183–202.

Ibrahim, A. O., Shamsuddin, S. M., & Qasem, S. N. (2015). Hybrid NSGA-II optimization for improving the three-term BP network for multiclass classification problems. *Journal of Information and Communication Technology, 14*, 21–38.

Jacob L. (2014). Bat algorithm for resource scheduling in cloud computing. *International Journal for Research in Applied Science & English Technolology*, *2*, 53–57.

Jena, R. K. (2015). Multi-objective task scheduling in cloud computing environment using nested PSO framework. *Procedia Computer Science*, *57*, 1219-1227.

Jonasson, J., & Norgren, E. (2016). *Investigating a genetic algorithm simulated annealing hybrid applied to university course time tabling problem*. KTH Royal Institute of Technology School of Computer Science and Communication. Degree Project Technology: Stockholm Sweden.

Kalra, M., & Singh, S. (2015). A review of metaheuristic scheduling techniques in cloud computing. *Egyptian Informatics Journal*, *16*, 275–295.

Kumar, R. S., & Gunasekaran, S. (2014). Improving task scheduling in large scale cloud computing environment using artificial bee colony algorithm. *International Journal of Computer Applications*, *103*(5), 29-32.

Kumar, V., & Rao, V.N. (1987). Parallel depth-first Search on multi-processors part II: Analysis. *International Journal of Parallel Programming*, *16*(6), 501-519.

Kirkpatrick, S., Jr. Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.

Khajehvand, V., Pedram, H., & Zandieh, M. (2014). Multi-objective and scalable heuristic algorithm for workflow task scheduling in utility grids. *Journal of Optimization in Industrial Engineering*, *14* (2014), 27−36.

Khajehvand, V., Pedram, H. & Zandieh, M. (2013). SCTTS: Scalable cost-time trade-off scheduling for workflow application in grids. *KSII Transactions On Internet and Information Systems*, *7*(12), 3096- 3117.

Lakra, A. V., & Yadav, D. K. (2015). Multi-objective *tasks scheduling algorithm for cloud computing throughput optimization*. *Procedia Computer Science*, *48* (2015), 107–113.

Leena, V. A., Ajeena, B. A. S., & Rajassree, M. S. (2016). Genetic algorithm based bi-objective task scheduling in hybrid cloud platform. *International Journal of Computer Theory and Engineering*, *8*(1), 7-13.

Letort, A., Carlsson, M., & Beldiceanu, N. (2015). Synchronized sweep algorithms for scalable scheduling constraints. *Constraints*, *20*, 183–234.

Li, J. Q., & Pan, & Q. K. (2015). Solving the large-scale hybride flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. *Information Sciences*, *316*, 487-502.

Mustaffa, Z., Yusof, Y., & Kamaruddin, S. S. (2013). Enhanced ABC-LSSVM FOR Energy Fuel Price Prediction. *Journal of Information and Communication Technology*, *12*, 73–101.

Mustaffa, Z., Sulaiman, M. H., & Yusof, Y. (2015). An application of grey wolf optimizer for commodity price forecasting. *Applied Mechanics and Materials*, *785*, 73-478. doi:10.4028/www.scientific.net/AMM.785.473.

Moschakis, I. A., & Karatza, H. D. (2015). Multi-criteria scheduling of Bag-of-Tasks applications on heterogeneous interlinked clouds with simulated annealing. *Journal of Systems and Software*, *101*, 1–14.

Monika, & Jindal, A, (2016). Optimization of task scheduling algorithm through QoS parameters for cloud computing. *MATEC Web of Conferences*, *57*, 02009 (2016). doi: 10.1051/matecconf/2016570.

Nikolaev, A. G., & Jacobson, S. H. (2010). Simulated annealing. In Gendreau, M., & Potvin, J.-Y. (Eds.), International series in operations research & management science. *Handbook on Metaheuristic*, *146*: Second Edition: Springer Science+Business Media, 1−39.

Pradhan, P.M., & Panda G. (2012). Solving multi-objective problems using cat swarm optimization. *International Journal Expert System with Application*, *39*(2012), 2956–2964.

Ramezani, F., Lu, J., & Hussain, F. (2013). Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization. *LNCS 8274*, 237–251.

Ramezani, F., Lu, J., Taheri, J., & Hussain, F. K. (2015). Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments. *World Wide Web*, *18*, 1737–1757.

Rani, K. B., Rani, P. B., & Babu, V. A. (2015). Cloud computing and inter-clouds-types, topologies and research issues. *Procedia Computer Science*, *50* (2015), 24-29.

Raza, H. M., Adenola, F. A., Nafarieh, A., & Robertson, W. (2015). The slow adoption of cloud computing and IT workforce. *Procedia Computer Science*, *52* (2015), 1114-1119.

Shengjun, X., Mengying, L., Xiaolong, X., & Jingyi, C. (2014). An ACO-LB algorithm for task scheduling in cloud computing environment. *Journal of Software*, *9*(2), 466-473.

Singh, P., Dutta, M., & Aggarwal, N. (2017). A review of task scheduling based on meta-heuristics approach in cloud computing. *Knowledge and Information Systems*, *52*(1), 1-51.

Sun, X., & Rover, D.T. (1994). Scalability of parallel algorithm-machine combinations. *IEEE Transactions on Parallel and Distributed Systems*, 5(6), 599-613.

Tsai, P.-W., Pan, J.-S., Chen, S.-M., & Liao, B.-Y. (2012). Enhanced parallel cat swarm optimization based on the Taguchi method. *Expert Systems with Applications*, *39* (2012), 6309–6319.

Taguchi, G., Chowdhury, S., & Taguchi, S. (2000). *Robust engineering*. New York: McGraw-Hill.

Tsai, J-T., Fang, J.-C., & Chou, J.-H. (2013) Optimized tasks scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. *Computers & Operations Research*, *40*, 3045–3055.

Wang, J., Zhou, B., & Zhou, S. (2016). An improved cuckoo search optimization algorithm for the problem of chaotic systems parameter estimation. *Computational Intelligence and Neuroscience*, 1-8. doi: http://dx.doi.org/10.1155/2016/2959370.

Yue, P., Shengjun, X., & Mengying, L. (2016). An improved multi-objective optimization algorithm based on NPGA for cloud task scheduling. *International Journal of Grid and Distributed Computing*, *9*(4), 161-176.

Zuo, L., Shu, L., Dongy, S., Chen, Y., & Yan, L. (2016). A multi-objective hybrid cloud resource scheduling method based on deadline and cost constraints. *IEEE Access*. doi: 10.1109/ACCESS.2016.2633288.

Zuo, L., Shu, L., Dong, S., Zhu, C., & Hara, T. (2015). A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. *IEEE Access*, 2687−2699.

Zhang, F., Cao, J., Li, K., Khan, S. U. & Hwang, K. (2014). Multi-objective scheduling of many tasks in cloud platforms. *Future Generation Computer Systems*, *37*, 309–320.