ISSN (Print): 2476-8316
ISSN (Online): 2635-3490
Dutse Journal of Pure and Applied Sciences (DUJOPAS), *Vol. 6 No. 3 September 2020*

A Look-Ahead Cat Swarm Optimization Workflow Scheduling Algorithm for Satisfying Cloud Consumer QoS Requirements

[1]Yahaya Saidu, [2] Faruq Umaru Ambursa, [2]Abdulwahab Lawan, [4]AbdulHakeem Ibrahim

[1]Department of Mathematical Sciences,
Faculty of Science,
Taraba State University, Jalingo

[2] Department of Information Technology,
Faculty of Computer Science and Information Technology,
Bayero State University,
Kano

[3]Katsina State Institute of Technology and Management,
Katsina State,
Nigeria

Email: yahayasaidu17@gmail.com

Abstract

As the world is advancing towards faster and more efficient computing paradigms, cloud computing has evolved as an efficient and cheaper solution to meet such increasing and demanding requirements. Cloud computing is a paradigm that enables users to access any amount of resources and other services anytime, anywhere over the Internet. Workflow scheduling (WFS) is a very important aspect in cloud computing which involves the process of mapping inter-dependent tasks on the available resources such that workflow application is able to complete its execution within the user's specified Quality of service (QoS) requirements. Generally, workflow applications come with various QoS objectives and constraints which are mostly conflicting in nature. Therefore, satisfying the QoS requirements for cloud consumers involves minimizing violation of the user defined QoS constraints. This has posed serious problem to research community. Recently, many researches have been reported in the literature to have addressed the challenge. However, most of the existing techniques provide low QoS satisfaction and violate the user-defined QoS constraints especially in highly tight constraints setting scenarios. This paper proposed a new mechanism based on cat swarm optimization (CSO) and a min-max heuristic strategy to produce better QoS satisfaction with minimize level of violation. The proposed approach has been implemented using Java programming language and experimental evaluation was conducted through simulation with two sample Workflows of different

**Author for Correspondence*

scales. Both the proposed Look-Ahead CSO (LACSO)and the benchmark Look-Ahead PSO (LAPSO) algorithms were examined in terms of Cumulative Violation Rate (CVR). Results from the experiments with respect to the metric demonstrated that LACSO algorithm significantly outperformed the benchmark LAPSO with about 39% decrease in terms of VR.

Keywords: Cloud Computing, Multi-Objective, Workflow Scheduling, QoS requirements, Cat Swarm Optimization (CSO)

INTRODUCTION

Cloud is a computing paradigm that allows users to access any amount of resources and other services anytime, anywhere via the internet (Verman & Kaushal, 2017).It is an internet-based computing platform that allowed buy of computing resources in form of infrastructures, platforms and software as subscription-based services in a pay-as-you-go model to consumers, in the same way that public utilities are. Today, most business and scientific processes conducted on the cloud platform are represented in terms of workflows. A workflow is an application composed of a collection of tasks with interdependencies which can be used to complete scientific, business or engineering processes. However, huge increase in computational capacities and the growing of computational density and data volume, typical workflow tasks involve transferring terabytes of data to high execution supercomputers performing complex computations and running huge data analysis and visualization, and handling the storage of the output results (Rodriguez, & Buyya, 2014).

The process of mapping inter-dependent tasks on the available resources such that workflow application is able to complete its execution within the user's specified Quality of service (QoS) objectives (time, cost, reliability, availability, security and reputation, etc.) and constraints such as deadline and budget is known as Workflow scheduling(Ambursa, Latip, Abdullah & Subramaniam, 2016).For instance, users might specify one or more preferred QoS parameter as the optimization objective(s). The QoS on the other hand, is a set of parameters that specify the properties of the cloud services. Thus, it is the ability to provide different priority to different workflow tasks and to guarantee certain level performance to the cloud resources(Kaur, Hans, & Kaur, 2018).

Generally, cloud users and service providers negotiate and delineate a binding agreement in terms of the QoS to be delivered and amount to be paid for the cloud services. This agreement is formally stipulated in a formal document known as service level agreement (SLA), which is a contract file signed between the cloud provider and the consumer that determines the set of QoS metrics that are used to measure services and penalties in case of violations (So & Jenkins, 2013). Therefore, satisfying the QoS requirements for cloud consumers involves minimizing violation of the user defined QoS constraints and improving the QoS satisfaction. Moreover, some of the QoS constraints such as deadlines, budgets, minimum reliability, etc. imposed by cloud users are conflicting in nature making the workflow scheduling more difficult task.This has posed serious challenge to research community because it involves a form of complex trade-off between the QoS constraints and objectives.

Recently, various scalable and dynamic heuristic, meta-heuristic and hybrid based algorithms have been proposed for finding optimal or sub-optimal solutions to the NP complete cloud workflow scheduling problem keeping in mind the QoS objectives and user constraints(Verman & Kaushal, 2017; Bilgaiyan, Sagnika & Das., 2015; Bousselmi, Brahmi, Gammoudi, 2016; Durillo, Prodan & Barbosa, 2015; Tao, Chang, Yi, Gu, & Li, 2011; Wang, Zhu, & Ramamohanarao, 2015; Ambursa *et al.,* 2016, Aron Chana & Abraham,2015; Sridhar,& Babu,2015; Chen, Liu, Wen, Chen, & Zhou, 2015; Choudhary, Gupta, Singh & Jana, 2018; Singh, Dutta & Aggarwal, 2017; Sun, Xiuo, Xu, 2018). Unlike heuristics or metaheuristics approaches, which focused on one or two objective optimization, hybrid algorithms are capable of handling multiple and conflicting QoS objectives that need to be satisfied in cloud environment. The hybrid algorithms join the advantages of both heuristics and metaheuristics due their mutually beneficial guided search approach

To the best of our acknowledge, only few among the approaches considered up to six user QoS objectives (Tao *et al.,* 2011; Ambursa *et al.,* 2016). In both cases, Particle Swarm Optimization (PSO) (Eberhart & Kennedy, 1995) was used with combination of other heuristics to solve the trading off multi-objectives problem. To date, Look-Ahead PSO algorithm which is a synergy between PSO and Min-Max based heuristic achieved good solutions in various degrees of QoS constraints and preferences defined by a cloud consumer (Ambursa *et al.,* 2016). However, the techniquealso violates the user-defined QoS constraints especially in highly tight constraints setting scenarios.

In this paper, a new multi-objective cat swarm optimization and min-max based workflow scheduling algorithm for satisfying cloud consumer requirements is proposed. Some of the contributions of the paper include:

i. To propose a multi-objective hybrid algorithm based on cat swarm optimization (CSO) and Min-Max heuristics strategy.
ii. To apply the proposed technique in minimizing the degree of violation of user QoS requirements in cloud environment.
iii. Implement and substantiate the performance of the proposed algorithm with other existing state-of-art approach via simulation experiment evaluation using workflow applications.

METHODOLOGY

A. Problem Formulation

Workflow Design

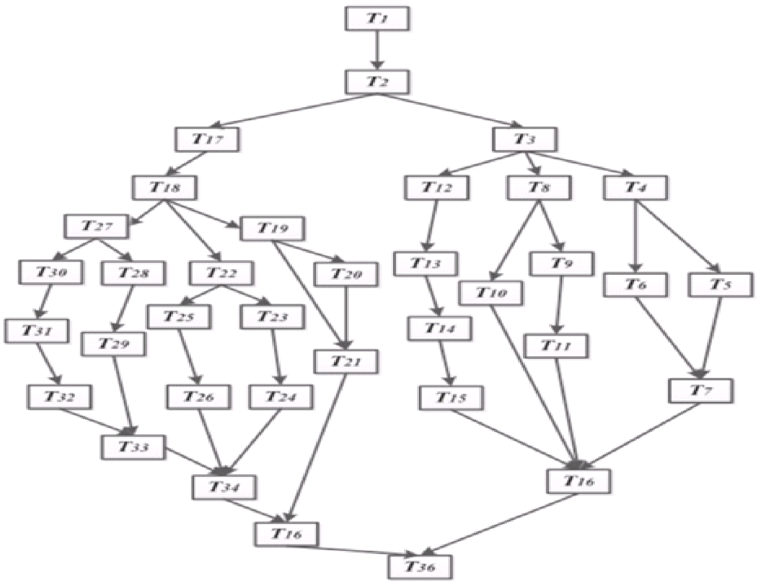

Figure 1.0: e-Business Workflow Application

A workflow application is modelled as a Directed Acyclic Graph (DAG), defined by $G = (T, E)$,where $T = \{t_1, t_{2\,,} \dots\ t_n\}$and represents the set of n tasks of the Workflow application, and E is a set of e edges which represent the dependencies or data communication between psairs of workflow tasks. Supposed that all tasks in T are compute-intensive and each t_i has a size $s_t$in

millions of instructions (MI) then, $e_{x,y} \in E,$ represents a communication from task$x\ to\ y$. Each $e_{x,y}$ link is denoted as (x, y) where x is the child (predecessor) task of $y\ and\ y$ is the parent (successor) task of x.The workflow task graph involves a relationship in which the execution of a child task cannot be started before its parent finishes its execution (Verma & Kaushal, 2017). A task with no parent is known as an entry task and a task with no children is known asexit task. Conventionally, a task graph has single entry and a single exit task. However, some task graphs possess more than one entry task. In such a situation, since task-scheduling algorithm may require a single entry and a single exit task, hence a pseudo task-entry with zero-cost edges is used and doing so does not affect schedule (Topcuoglu, Hariri, & Society, 2002). Figure 1.0 illustrated a samplee-Business workflow application used in this research.

In this study, the satisfaction of hard requirements is ensured by evaluating the feasibility of a scheduling solution based on meeting user constraints in every QoS dimension. The User-defined *hard* requirements are set of constraints that must be fulfilled to consider a scheduling solution feasible. To check whether a solution is feasible or not, a constraint violation detection model is used which can be described as a penalty function for the cumulative QoS function. It indicates the violation rate of each scheduling solution. The model is mathematically expressed as follows:

$$\text{minimize } \nu_\Psi = \delta_{\pi_{\text{dln}}} + \delta_{\pi_{\text{bgt}}} + \delta_{\pi_{\text{rel}}} + \delta_{\pi_{\text{avl}}} + \delta_{\pi_{\text{sec}}} + \delta_{\pi_{\text{rep}}}\ \text{s.t}\ \nu_\Psi = 0 \tag{1}$$

where ν_Ψ, represents the cumulative QoS violation rate (CVR) of the schedule Ψ, $\delta_{\pi_{\text{dln}}}$, $\delta_{\pi_{\text{bgt}}}$,$\delta_{\pi_{\text{rel}}}$, $\delta_{\pi_{\text{avl}}}$, $\delta_{\pi_{\text{sec}}}$, $\delta_{\pi_{\text{rep}}}$ are violations of the deadline, budget, reliability, availability, security and reputation respectively. The lower value of ν_Ψ the better, since the low mean the solution is approaching a feasible side. Thus, a solution is feasible if it attains 0% CVR. The solution selection process is demonstrated in algorithm 1.

Algorithm 1: QoS Constraints-aware Solution Selection

```
Input: $a$ and $b$
Output: $best$
  1. Compute $\nu_a$ and $\nu_b$ using equation 1
  2. $best \leftarrow a$
  3. If $\nu_{best} > 0$ & $\nu_b > 0$ then // both schedules are infeasible
  4.     If $\nu_b < \nu_{best}$ then
  5.         $best \leftarrow b$
  6. Else If $\nu_{best} = 0$ & $\nu_b = 0$ then // both schedules are feasible
  7. Else If $\nu_b = 0$ & $\nu_{best} > 0$ then // b is feasible and a is not
  8.     $best \leftarrow b$
  9. Return $best$
```

B. Classic Cat Swarm Optimization (CSO) Algorithm

CSO is a new optimization algorithm in the field of swarm intelligence (Chu, Tsai, & Pan, 2007) and was developed by Chu and Tsai (2007) based on the behavior of cats. Cats naturally have high alertness and curiosity about their surroundings and moving objects in their environment. CSO models the behavior of the cats into two modes: 'Seeking mode' and 'Tracing mode' to find the global and local solutions of optimization problems respectively. Most swarm-based algorithms were inspired to simulate intelligent behavior of biological creatures such as birds, ants, fish, cats, frogs, wolfs, bees etc. therefore, each swarm is made of initial population composed of particles to search in the solution space. Here, cats are used as particles for solving the problems. In CSO, every cat has its own position composed of D dimensions, velocities for each dimension, a fitness value, which represents the accommodation of the cat to the fitness function, and a flag to identify whether the cat is in seeking mode or tracing mode. The final solution would be the best position of one of the cats. The CSO keeps the best solution until it reaches the end of the iterations (Ouroskhani et al., 2013). CSO has two modes in order to solve optimization problems which are described below: CSO has four essential parameters:

i. **Seeking Memory Pool (SMP):** the number of copies (replicas) for each cat that are to be made during the seeking process. One among them will replace the original cat later.
ii. **Seeking Range of the selected Dimension (SRD):** the maximum difference between the new and old values in the dimension selected for mutation
iii. **Counts of Dimension to Change (CDC):**The number of allotments in each copy that will be modified. Each copy will be evaluated, and one of the best will be *selected for replacement.*
iv. **Self Position Consideration (SPC):**this is Boolean variable which indicates the current position of the cat as a candidate position movement. SPC can not affect the value of SMP.

Seeking Mode

The seeking mode is used to model the cat during a period of resting but beingalert-looking around its environment for its next move.The process of seeking mode is describes as follow:

Step1: Make j copies of ʽP_{cat_k}, where j = SMP. If the value of SPC is true, let j = (SMP-1), then retain the present ʽP_{cat_k}as one of the candidates. Where ʽP_{cat_k}denotes cat position in M-dimensional space

Step2: For each copy, according to CDC, randomly plus or minus SRD percent of the present values and replace the old ones.

Step3: Calculate the fitness values $f(cat_k)$of all candidate cats.

Step4: If all $f(cat_k)$are not exactly equal, calculate the selecting probability of each candidate cat, otherwise set all the selecting probability of each candidate to be 1.

Step5: Randomly pick ʽ$P_{cat_k}$to move to from the candidate ʽ$P_{cat_k}$and replace ʽ$P_{cat_k}$of the original cat_k.

Step1: compute the new velocityV_i^{t+1}, for every dimension according to:

$$V_i^{t+1} = \omega V_i^t + \mathrm{cr}(\mathrm{X_{best}} - X_i^t) \qquad (2)$$

Where ω represents inertia weight, V_i^t previous velocity, c acceleration constant, r random number in the range between 0 and 1. X_i^t, denotes the current location and $\mathrm{X_{best}}$, the best location

Step2: Update the position of cat according to (3).

$$X_i^{t+1} = X_i^t + V_i^{t+1} \qquad (3)$$

Step3: check if the position of the cat goes beyond the specified range. If so, assign the boundary value of the cat.

Step4: evaluate the fitness value of the cats and update the solution set with the best positions of the current iteration.

C. The Proposed Look-ahead Cat Swarm Algorithm (LACSO)

The LACSO algorithm is a synergy between CSO algorithm and constraint handling algorithms(CSS and auxiliary Min-Max heuristic). The LACSO procedure is depicted in algorithm 5. The base line CSO metaheuristic optimizes scheduling solution by iteratively searching for and producing good solutions. In order to combine its two modes, a mixture ratio (MR) which indicates the rate of mixing of seeking mode and tracing mode is defined. This parameter decides how many cats will be moved into seeking mode or tracing mode process. For example, if the population size is 60 and the MR parameter is equal to 0.65, there should be 60×0.65 = 39cats move to seeking mode and 21 remaining cats move to tracing mode in this iteration.

First of all, N cats are created and then, their positions and velocities are initialized. Also, flags are assigned to each of the cats to indicate which mode they belong and then the fitness value of each of the cats is evaluated according to the objective function defined in equations (1), the best cat is then stored in the memory.

In next step, according to cat's flag, a seeking mode or tracing mode process is activated by a cat. The descriptions of how the two modes operate are provided in algorithm 2 and algorithm 3 respectively. After completing the related processes in the two modes, the fitness values of all cats are re-evaluated to determine the best cat (solution) and store in the memory. Algorithm 4is called to appraise and choose good scheduling solutions from the pool of solutions generated by CSO during the optimization process at the end of each iteration.

At the end, termination condition is checked, if satisfied, the program is terminated and the global best solution is returned. However, if termination criteria are not satisfied, then cats are redistributed into seeking and tracing modes and the process is restarted all over again. Note that $lBest_i^t, gBest_i^t$ and $gBest_i^{t+1}$ denote the local best, global best and improved global best schedule solutions respectively.

Algorithm 2: Cat's Seeking Mode

```
1. Start
2. Foreach cat in seeking mode
3.     Make j copies $\mathrm{cat_k}$ where j = SMP.// copies of the present position of a cat
4.     If the value of SPC is true, let j = (SMP-1), then
5.             retain the present $P_{\mathrm{cat_k}}$ as one of the candidates.
6.     For each j, according to CDC, randomly $+ \ or \ -$SRD% from the present values and replace the old ones.
7.     Calculate the $f(\mathrm{cat_k})$ of all candidate cats using Eq. (1)
8.     Invoke algorithm 3.1
9.     Randomly pick $P_{\mathrm{cat_k}}$ to move to from the candidates $P_{\mathrm{cat_k}}$ and replace $P_{\mathrm{cat_k}}$ of original $\mathrm{cat_k}$.
10. End for
11. End
```

Algorithm 3: Cat's Tracing Mode

```
1. Start
2. Foreach cat in tracing mode
3.     compute $V_i^{t+1}$, for every dimension using Eq. (2)
4.         update $X_i^{t+1}$ for cat according to Eq. (3)
5.     If $X_i^{t+1} > [0,1]$ then
6.         $X_i^{t+1} = 1$
7.     evaluate $f(\mathrm{cat_k})$ using Eq. (1) and
8.         update $\mathrm{lBest}_i^t$ using Eq. (3)
9. End for
10. End
```

The Fitness function modeled the (six) QoS constraints (deadline, budget, minimum reliability, minimum availability, average security and average reputation) and isused to tracks cumulative violation levels of scheduling solutions with the aim of meeting user hard requirements. It is used to determine and sum up all the violations (if any) of the user-defined constraints.

D. Min-Max Heuristic (or Look-Ahead QoS-aware Task-Remapping- LATR) Algorithm

The LATR is the core of LACSO algorithm aims to improve the global best scheduling solution found by the baseline CSO through deterministically altering one of the task service mappings of the best solution based on a min–max heuristic. Algorithm 3 represents the LATR. For the detail description of definitions and operations of the denotations used in the algorithm refer to the reference (Ambursa *et al.,* 2016).

Algorithm 4: Look-ahead QoS-aware Task-Remapping Algorithm (LATR)

```
    Input: $\phi^{t}_{best}$
    output: $\phi^{+}_{best}$

 1. $\sigma_{inferior} \leftarrow \mathrm{searchO}_{QoSSet_{best}}$
 2. $\phi_{\tau}^{\sigma_{inferior}} \leftarrow \mathrm{search}(\sigma_{inferior})_{MapSet_{best}}$
 3. Counter $\leftarrow 1$ // First attempt to improve $\phi_{\tau}^{\sigma_{inferior}}$
 4. $\phi_{determ} \leftarrow \phi^{t-1}_{best}$
 5. $\phi_{\tau}^{\sigma_{inferior}} \leftarrow \mathrm{search}(\sigma_{inferior})_{MapSet_{determ}}$
 6. If $\phi_{\tau}^{\sigma_{inferior}} > \phi_{\tau}^{\sigma_{inferior}}$ then
 7. $AS \leftarrow S_{\tau}^{h}$
 8. $temp \leftarrow copyof\,\phi^{t}_{best}$
 9. $temp \leftarrow Remap\ (h, AS)$ //using algorithm (1)
10.     Calculate $\lambda_{temp}$ using equation (1)
11. If $\lambda_{best} = 0$ then // best solution is feasible
12. If $\lambda_{temp} = 0$ then
13. Calculate $\mathcal{W}_{temp}$ using equation (3.3)
14. If $\mathcal{W}_{temp} > \mathcal{W}_{best}$ then
15. $\phi^{+}_{best} \leftarrow$ Remap $(h, AS)$
16. Else // $\lambda_{best} > 0$ $(best\ is\ infeasible)$
17. If $\lambda_{temp} < \lambda_{best}$ then
18. $\phi^{+}_{best} \leftarrow$ Remap $(h, AS)$ using PSO algorithm
19. Else
20.     If Counter $\leftarrow$ 1then
21. Counter ++ // second attempt to improve$\phi_{\tau}^{\sigma_{inferior}}$
22. $\Phi_{determ} \leftarrow$ rand(current PSO swarm)
23. Go to line 6
24. Return $\phi^{+}_{best}$
```

In summary, the goal LATR is to enable robust performance in terms of ability to handle different kinds of QoS requirements settings. Consequently, a task with minimum QoS value is selected for remapping(using a determinant scheduling solution ϕ_{determ} as a guide) to an alternative service that can maximize its QoS. Line 1-18 of algorithm 3 showed how the remapping process is performed in first attempt. If, however, the condition of line 6 is not satisfied, then the algorithm makes a second attempt by choosing a random determinant solution from the current CSO solutions and retries (line 19–23). Detail explanation of how each step of the algorithm operates can be found in the reference (Ambursa et al., 2016).

Algorithm 5: Look-ahead CSO (LACSO) Algorithm

```
1. Start
2. initialize N; MR; SMP; SRD; CDC;
3. Generate an empty M-dimensional solution space array of (N × M) size
4. Set all $CanSerSet_{QoS}$// set candidate services sets and their corresponding QoS attributes
5. Initialize $x_i^t$ and $v_i^t$
6. According to the value of MR assign each cat a flag to fit them into seeking or tracing modes
7. iter ← 10000
8. Foreach iter do
9.    Calculate $f(cat_k)$of each cat according to Eq. (1)
10.     Compare $f(cat_k)$for all cats, and
11.       $f(cat_k) \leftarrow f(cat_k)_{best}$// store the position with best fitness value (FS) in memory.
12.    If (flag = 1)
13.         Invoke algorithm 2 //Seeking mode process
14.    Else
15.         Activate algorithm 3 // Tracing mode process
16.    Re-evaluate $f(cat_k)$ for all cats using Eq. (1)
17.    Update $gBest_i^t$ using Eq. (3)// global-best cat
18.     Invoke algorithm 4 to improve the schedule solution
19.    If (termination condition is not reached?)
20.       counter + +
21.       Go to step 6.
22.    Else
23.       Return the $gBest_i^{t+1}$ solution
24. End for
25. End
```

1. Experimentation

The experimentation study was conducted by means of simulation using NetBeans IDE for Java Programming environment. The simulation was performed on a computer with a Intel(R) Core(TM) Duo CPU (2.17 GHz, 2 cores) and a 2GB RAM with 32-bit MS Windows 7 Operating System. The results were illustrated with excel files.The assumptions and experimental settings are adopted from the benchmark work (Ambursa et al., 2016)

In this simulation experiment two workflow applications a 15-task e-Protein (scientific workflow) and 36-tasks e-business workflow were used to evaluate the performances of both the proposed and the benchmark algorithm as used in reference (Tao, *et al.,* 2009; Tao, *et al*., 2011; Ambursa *et al.,* 2016). To each of the tasks in the workflow, depending on the test case, a medium scale was randomly assigned – in the range 60 to 120- or a large scale in the range 60 to 150 – set of service instances as shown in table 1. The values for the six QoS parameters of each service instance were also randomly generated (Q. Tao, *et al.,* 2011; Ambursa *et al.,* 2016), but the follow the rule that for the same task the service cost depend on the values of the other parameters (Chen & Zhang, 2009).

The algorithms were assessed in four (4) different test instances as depicted in table 1. The test instances demonstrated a range of different scheduling scenarios with different levels of complications – from moderate to tight situations. For each test case 20 independent runs were carried out with 10000 iterations in each for both the two algorithms and the resultsof comparison with respect to the cumulative violation of constraints Rate were recorded. For the purpose of graphical illustration LAPSO and LACSO are used to denote the results of the two algorithms. The initial population chosen for both algorithms is 60. PSO parameters are adopted from and used according to (Tao, *et al.,* 2011; Ambursa *et al.,* 2016). Also, the CSO parameters setting used in this work are as follows:$MR = 0.65 CDC = 0.66, SMP = 7, c = 2.05$,,r in the range [0,1]. These configurations are chosen in accordance with recommendations provided in the references (So & Jenkins, 2013; Pradhan & Panda, 2012; Orouskhani *et al.,* 2013;Bilgaiyan *et al.,* 2015; Gabi *et al.,* 2016).

Table 1: *Test Instances*

Test Case	**Workflow size**	**Service scale**	**Constraints**					
Case			Budget	Deadline	Min. Availability	Min. Reliability	Mean Security	Mean Reputation
1	15	60-120	75	40	0.8	0.8	5.4	4.5
2	15	60-120	70	42	0.9	0.92	5.7	5.6
3	36	60-150	165	35	0.8	0.8	5.5	4.8
4	36	60-150	95	35	0.8	0.8	5.3	4.7

2.Results and Discussion

The experimentation study was carried out by means of simulation using NetBeans IDE for Java Program development. The two schemes were tested using two workflow applications (scientific and business workflow applications) with 15 and 36 tasks respectively as used in reference (Tao, *et al.,* 2009; Tao, *et al.,* 2011; Ambursa *et al.,* 2016). To each of the tasks in the workflow, depending on the test case, a medium scale was randomly assigned – in the range 60 to 120- or a large scale in the range 60 to 150 – set of service instances as shown previously in table 1. The values for the six QoS parameters of each service instance were also randomly generated (Q. Tao, *et al.,* 2011; Ambursa *et al.,* 2016), but the follow the rule that for the same task the service cost depend on the values of the other parameters (Chen & Zhang, 2009) as cited in (Ambursa *et al.,* 2016).

Test case 1: Moderate Constraints (medium scale Cloud)

In this experiment, all the user-defined hard constraints, including budget are set to moderate condition and the cloud scale to medium. From the simulation resultsin Fig. 2.0, LACSO managed to fulfill all the user-defined *hard* constraints with no violation (0%violation). However, in case of LAPSO 5.63% violation has been recorded. In addition, LACSO achieved faster convergence than LAPSO in that the scheme attained 0% violation at iteration 168. Contrariwise, LAPSO attained its minimum violation rate of 5.63% at iteration 7452.

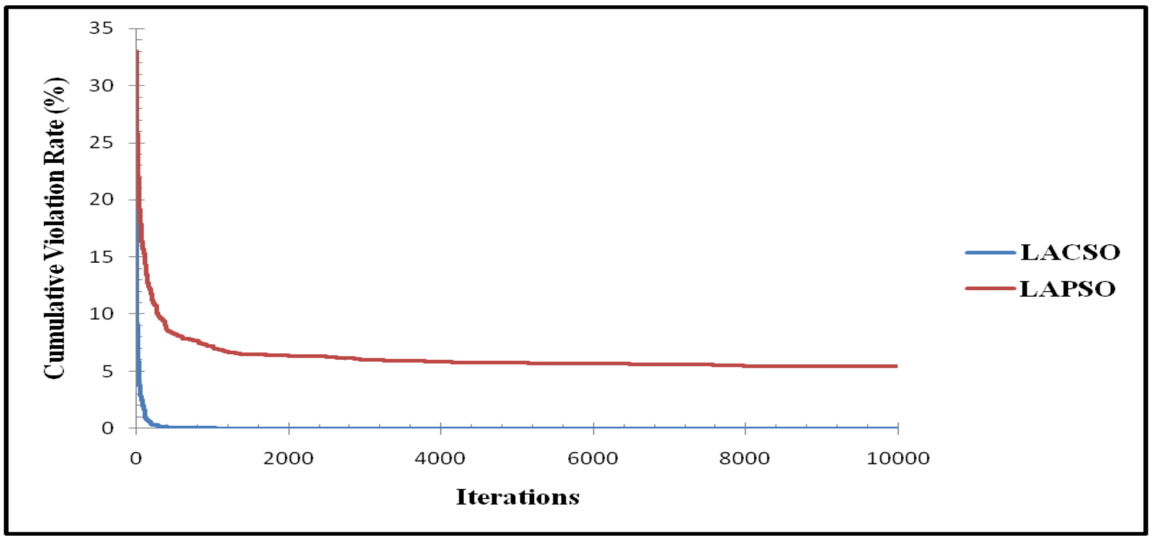

Fig.2: CVR under moderate user constraints and medium scale cloud

Test case 2: Moderate Budget & tight other Constraints (medium scale Cloud)
In this experiment, budget is set to moderate and all the other user-defined hardconstraints to tight settings and the cloud scale to medium. From the simulation results, Fig.3.0 showed the performances of the two schemes. With respect to the CVR, LACSO has managed to achieve 2.59% violation at iteration 1037.On the hand, LAPSO has recorded violations of 13.23% at iteration 6493. The result has showed that LACSO has upper hand in handling user hard constraints over LAPSO.

Test case 3: Moderate Constraints (Large scale Cloud)

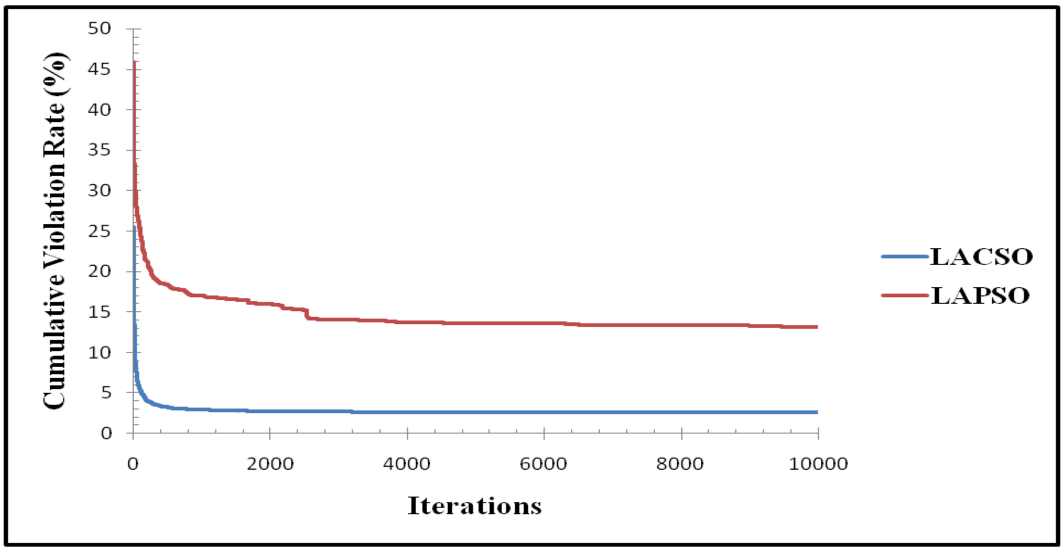

Fig. 3: CVR under moderate budget, tight other constraints & medium scale cloud

In this experiment, budget and all other user-defined hard constraints, are set to moderate situations and the cloud scale to large. Fig.4.0 showed the performances of the two schemes. With regard to CVR, LAPSO has recorded 6.18% violation of user-defined hard constraints at iteration 5367. Conversely, LACSO had zero or no violation at iteration 2622. In view of this, LACSO achieved faster convergence than LAPSO.

Test case 4: Tight Budget & moderate other Constraints (large scale Cloud)

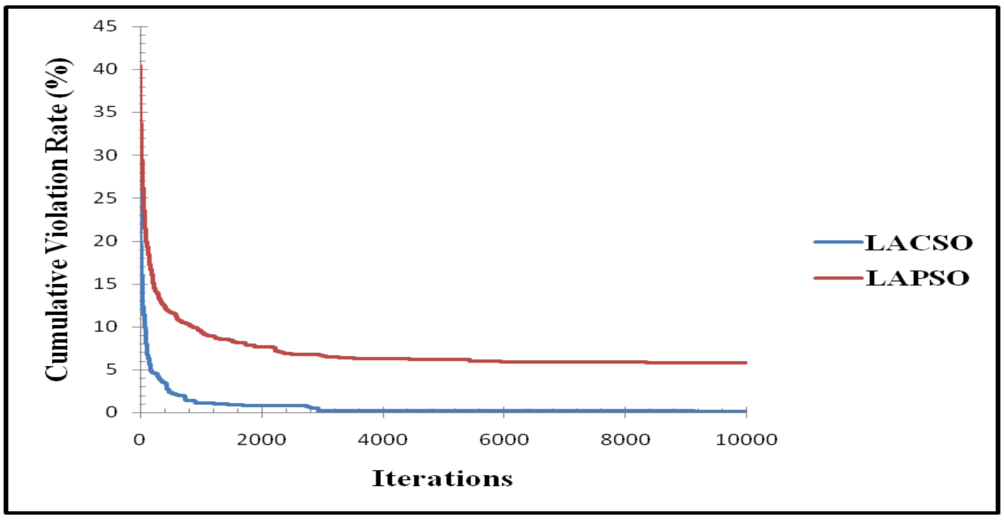

Fig. 4.0: CVR under moderate budget and other user constraints, a large scale cloud

The budget is set to tight and all the other user-defined hard constraints to moderate situations and the cloud scale to large. faster convergence than LAPSO, in that LACSO attained the 5.50% violation at iterations 3714 whereas, LAPSO at iterations 7885.This shows the advantage of LACSO over LAPSO. Fig. 4.0 below illustrated the result.

4.1 Results Analysis

In respect to CVR, Figs. 2, 3, 4 and 5 presented the results of comparison between LACSO and LAPSO. In most of the test instances under the moderate constraints, both algorithms can find feasible solutions with a much minimal level of violation, because large number of feasible solutions exists in the solution space. Hence, the algorithms can search and easily find feasible solutions. However, as tight constraints are imposed, finding feasible solutions become much difficult for LAPSO algorithm. However, the constraint violation detector assisted both algorithms in detecting and handling violations in any of the user constraints and selecting the best solution based on the CVR when the solutions are in the infeasible region.

From the Figures, it can be noticed that LACSO achieves better solutions result with minimum violation level compared to the LAPSO. The two approaches used different baseline metaheuristic. The former used CSO and the latter PSO. The main difference in the

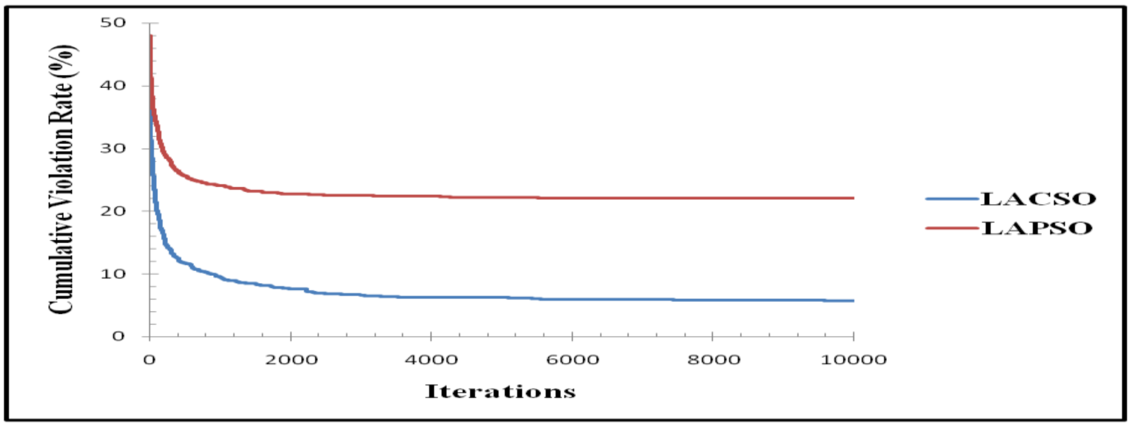

Fig. 5: CVR under tight budget, moderate other constraints and large scale cloud

learning rule between the PSO and CSO is that the local and global searches for *pBest* and *gBest*, respectively, in PSO affect all particles' velocity and position that decide the step size of the particle in the next generation. On the other hand, the CSO has different strategies for global and local searches. The update of velocity and position is used only for the global search cats in seeking mode, not for the local search cats in tracing mode. For the cats in seeking mode, the step size of them is selected by the SRD value chosen, not affected by the velocity and position. Without the large change in the velocity and position of seeking mode cats, they are possible to focus on an exhaustive search for the local area. Thus, the reason for the better performance of LACSO over LAPSO is attributed to the inherent CSO.

On the whole, the simulation results showed the robustness and efficiency of the proposed LACSO in satisfying various users' QoS requirement settings, delivering solutionswith minimal violation. Consequently, with respect to the metric CVR which is the focus of this research used in the comparison between the algorithms, LACSO significantly outperform med the benchmark algorithm LAPSO.

CONCLUSION

In this paper, a multi-objective hybrid algorithm based on cat swarm optimization (CSO) and Min-Max heuristics for scheduling workflow applications on cloudhas been presented. The proposed algorithm has been implemented using Java programming language. Results from simulation experiment conducted using two workflow applications (scientific and business workflow applications) with 15 and 36 tasks demonstrated that the proposed strategy is effective enough to minimize the violation of QoS requirements for cloud consumers' resources. Hence, LACSO significantly outperforms the state-of-the-art algorithm (LAPSO) in terms of mininizing various users' QoS constraints in a heavily tight situations. In the future, we intend to evaluate the LACSO algorithm on real cloud environment with real time infrastructures such as Amazon EC2 or some other IaaS platforms. Moreover, a selection strategy that will allow LACSO algorithm to schedule multiple workflows at a time instead of single shall be exploited.

REFERENCES

Ambursa, F. U., Latip, R., Abdullah, A. & Subramaniam, S. (2016). A particle swarm optimization and min–max-based workflow scheduling algorithm with QoS satisfaction for service-oriented grids. *Journal of Supercomputing*.73:2018. doi:10.1007/s11227-016-1901-x

Aron, R., Chana, I., & Abraham, A. (2015). A hyper-heuristic approach for resource provisioning based scheduling in grid environment. *Journal of Supercomputing,*1427–1450. http://doi.org/10.1007/s11227-014-1373-9

Bilgaiyan, S., Sagnika, S., & Das, M. (2015). A Multi-Objective Cat Swarm Optimization Algorithm for Workflow Scheduling in Cloud Computing Environment. *International* Journal of Soft Computing 10(1): 37-45

Bousselmi, K.,Brahmi, Z., & Gammoudi, M. M. (2016, March). QoS-aware scheduling of workflows in cloud computing environments. In *Advanced Information Networking and Applications (AINA), 2016 IEEE 30th International Conference on* (pp. 737-745). IEEE.

Chen, C., Liu, J., Wen, Y., Chen, J. Zhou, D. (2015). A hybrid genetic algorithm for privacy and cost aware scheduling of data intensive workflow in cloud. In: International

Conferenceon Algorithms and Architectures for Parallel Processing, pp. 578–591. Springer

Chen, W. N.,&Zhang, J. (2009). An Ant Colony Optimization Approachto a Grid WorkflowScheduling Problem with Various QoS Requirements.*IEEE Trans. on Systems, Man, and Cybernetics,Part C: Applications and Reviews,* 39, 29-43.

Choudhary, A., Gupta, I., Singh, V., Jana, P.K. (2018). A GSA basedhybrid algorithm for bi-objective workflow scheduling in cloudcomputing. *Future Gener. Comput. Syst.* **83**, 14–26

Chu, S.C.,& Tsai, P.W. (2007). "Computational Intelligence Based On TheBehavior Of Cats". International Journal of Innovative ComputingInformation and Control, 3(1), 163-173.

Chu, S.C., Tsai, P.W., & Pan, J.S. (2006). Cat swarm optimization, Proc. Of the 9th Pacific Rim International Conference on Artificial Intelligence, LNAI 4099, pp.854-858.

Durillo, J .J., Prodan, R., & Barbosa, J. G. (2015). Pareto tradeoff scheduling of workflows on federated commercial clouds. Simulation Modelling Practice and Theory, 58(February), 95-111. http://dio.org /10.1016/j.simpat.2015.07.01

Eberhart, R., & Kennedy, J. (1995, October). A new optimizer using particle swarm theory. In Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on (pp. 39-43). IEEE.

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. Journal of Neural Comput & Applications. DOI 10.1007/s00521-016-2816-4

Gabi D., Ismail, A.S., Zainal, A., Zakaria, Z. & Abraham. A. (2017). Cloud Scalable Multi-Objective Task Scheduling Algorithm for Cloud Computing Using Cat Swarm Optimization and Simulated Annealing. EEE 2017 8th International Conference onInformation Technology (ICIT)

Kaur, S., Bagga, P., Hans, R., & Kaur, H. (2018). Quality of Service (QoS) Aware Workflow Scheduling (WFS) in Cloud Computing: A Systematic Review. Arabian Journal for Science and Engineering, 44(4), 2867-2897. https://doi.org/10.1007/s13369-018-3614-3

Orouskhani, M., Orouskhani, Y., Mansouri M., & Teshnehlab M. (2013). A Novel Cat Swarm Optimization Algorithm for Unconstrained Optimization Problems. I.J. Information Technology and Computer Science, 11(2013), 32-41. DOI: 10.5815/ijitcs.2013.11.04

Pradhan, P. H., & Panda,G. (2012). Solving multi-objective problems using cat swarm optimization. Expert Systems with Applications, 39 (2012), 2956–2964.

Singh, P., Dutta, M., & Aggarwal, N. (2017). A review of task scheduling based on meta-heuristics Approach in cloud computing. Knowledge Information System DOI: 10.1007/s10115-017-1044-2

So, J., & Jenkins, W. K. (2013). Comparison of Cat Swarm Optimization with particle swarm optimization for IIR system identification. 2013 Asilomar Conference on Signals, Systems and Computers.doi:10.1109/acssc.2013.6810419.

Sridhar, M., Babu, G.R.M. (2015). Hybrid particle swarm optimizationscheduling for cloud computing. In: Advance Computing Conference(IACC), 2015 IEEE International, pp. 1196–1200. IEEE

Sun, T., Xiao, C., & Xu, X. (2018). A scheduling algorithm using sub-deadline for workflow applications under budget and deadline constrained.Cluster Computing https://doi.org/10.1007/s10586-018- 1751-9

Tao, Q., Chang, H. Y., Yi, Y., Gu, C. Q., &. Li, W. J (2011). A rotary chaotic PSO algorithm for trustworthy scheduling of a grid workflow. Computers and Operations Research., 38(5), 824–836. http://doi.org/10.1016/j.cor.2010.09.012

Tao, Q., Chang, H. Y., Yi, Y., Gu, C. Q., &. Yu Y. (2009). QoS Constrained Grid Workflow Scheduling Optimization Based on Novel PSO Algorithm. In Eighth International Conference on Grid and Cooperative Computing (pp.153–159). IEEE. http://doi.org/10.1109/GCC.2009.39

Topcuoglu, H., Hariri, S., & Society, I.C. (2002). Performmance-Effective and Low-Complexity. Parallel and Distributed Systems. IEEE Transactions on13(3), 260-274.

Verma, A., & Kaushal, S. (2017). A Hybrid Multi-Objective Particle Swarm Optimization for Scientific Workflow Scheduling. Parallel Computing. doi:10.1016/j.parco.2017.01.002

Wang, M., Zhu, L., & Ramamohanarao, K. (2015). Reasoning task dependencies for robust service selection in data intensive workflows. Computing, 97(4), 337–355. http://doi.org/10.1007/s00607-013-0381-6

Zhang Q. Cheng L. & Boutaba R. (2010). Cloud Computing: State-of-the-Art and Research Challenges. Journal of Internet Services and Application, Springer, 7-18