International Journal of Electrical and Computer Engineering (IJECE)
Vol. 7, No. 3, June 2017, pp. 1489~1497
ISSN: 2088-8708, DOI: 10.11591/ijece.v7i3.pp1489-1497

Solving Task Scheduling Problem in Cloud Computing Environment Using Orthogonal Taguchi-Cat Algorithm

Danlami Gabi[1], Abdul Samad Ismail[2], Anazida Zainal[3], Zalmiyah Zakaria[4]
[1,2,3,4] Department of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia
[1] Department of Computer Science, Kebbi State University of Science and Technology, Nigeria

Article Info

Article history:

Received Jan 9, 2017
Revised Mar 15, 2017
Accepted Apr 8, 2017

Keyword:

Cat swarm optimization
Cloud computing
Makespan
Taguchi optimization
Task scheduling

ABSTRACT

In cloud computing datacenter, task execution delay is no longer accidental. In recent times, a number of artificial intelligence scheduling techniques are proposed and applied to reduce task execution delay. In this study, we proposed an algorithm called Orthogonal Taguchi Based-Cat Swarm Optimization (OTB-CSO) to minimize total task execution time. In our proposed algorithm Taguchi Orthogonal approach was incorporated at CSO tracing mode for best task mapping on VMs with minimum execution time. The proposed algorithm was implemented on CloudSim tool and evaluated based on makespan metric. Experimental results showed for 20VMs used, proposed OTB-CSO was able to minimize makespan of total tasks scheduled across VMs with 42.86%, 34.57% and 2.58% improvement over Minimum and Maximum Job First (Min-Max*)*, Particle Swarm Optimization with Linear Descending Inertia Weight (PSO-LDIW) and Hybrid Particle Swarm Optimization with Simulated Annealing (HPSO-SA) algorithms. Results obtained showed OTB-CSO is effective to optimize task scheduling and improve overall cloud computing performance with better system utilization.

Corresponding Author:

Danlami Gabi,
Departement of Computer Science, Faculty of Computing,
Universiti Teknologi Malaysia,
81310, Skudai, Johor, Malaysia.
Email:gdanlami2@live.utm.my

1. INTRODUCTION

Cloud computing is an internet-based computing that offers different services (Infrastructure, platform, and software as a service) in form of pay-as-you-go [1]−[9]. The existence of enterprises prompt competitiveness in term of revenue generation, hence, task scheduling is quite necessary for avoiding some level of revenue loss, performance degradation, and breach of service level agreement (SLA) [10],[11]. Therefore, dynamic solutions are needed to schedule tasks across heterogeneous virtual machines (VMs), where the performance of these solutions can be measured based on i.e., makespan [12]−[15]. Makespan minimization is used to measure how effective a task scheduling algorithm in reducing task execution delay. To effectively optimize task scheduling in order to minimize task execution time is however considered as NP-hard optimization problem [1],[16]. Previous researchers have proposed task scheduling solutions that minimized task execution delay, revenue loss and maximized system performance using artificial intelligence (AI) techniques [6],[13],[17]−[24]. Cat swarm optimization (CSO) is one of the swarm intelligence (SI) techniques that have proven faster than particle swarm optimization in term of speed and convergence [25], [26]. CSO can be a potential solution when improved to address task scheduling problem in cloud computing.

In this study, we incorporated Orthogonal Taguchi based approach [27] at local search (tracing mode) of CSO and increased its convergence speed for task mapping on VMs with minimum execution time

[6]. In former term, a task scheduling model for the task execution time was proposed. As a result, Orthogonal Taguchi Based-Cat Swarm Optimization (OTB-CSO) is proposed that minimized the model [28].

Contributions for the proposed work are summarized as follow:

- Development of *makespan* model for optimal task scheduling for cloud computing environment;
- Hybridization of *Taguchi* optimization approach with CSO for optimal task scheduling in cloud computing environment;
- Implementation of the proposed algorithm on CloudSim tool
- Performance comparison based on makespan metric with percentage improvement.

The rest of this article is organized as followed: Related works on task scheduling based on Taguchi approach in Section 2. Section 3 discussed modelling of the scheduling problem. Cat swarm optimization is discussed in section 4. Section 5 discussed the proposed tracing mode process of CSO algorithm. Taguchi optimization with proposed OTB-CSO is discussed in Section 6. Section 7 discussed the experimental setup. Results obtained are presented in Section 8. Section 9 discussed the results of simulation and Section 10 concluded the paper.

2. RELATED SCHEDULING WORKS BASED TAGUCHI APPROACH

We reported few existing researchers that explored Taguchi based approach to solved optimization problems as followed: [29], used Taguchi method in conjunction with simulation modeling of new applications for robotic flexible assembly cells (RFACs) to minimized total tardiness and number of tardy jobs. In [8], an improved differential evolutionary algorithm (IDEA) using Taguchi based approach that optimized task scheduling and resource allocation problem were presented. In [30], an optimum scheduling algorithm based Taguchi approach was presented. In [31], a hybrid no-wait flexible flow shop scheduling algorithm that combined non-static genetic algorithm (NSGA-II) with variable neighborhood search (VNS) was presented based on Taguchi method and minimized makespan and mean tardiness of jobs. In [32], an improved effective genetic algorithm using Taguchi approach was presented that minimized total order completion time (makespan). In [33], an algorithm that studied the effect associated with scheduling rules based on the performance of a dynamic scheduling in flexible manufacturing systems was presented using Taguchi approach. In [34], a Taguchi-based genetic algorithm (TBGA) that solved the problem of job shop scheduling was presented. Based on existing works, this study explored the method used to designed TBGA as proposed in [34] for the design of our proposed OTB-CSO algorithm that addressed independent task scheduling and achieved minimum makespan of total tasks scheduled across VMs for a dynamic cloud environment**.**

3. MATHEMATICAL MODEL OF THE SCHEDULING GOAL

The formulation of the objective model for the task scheduling problem is based on [35] and [36] as follow: Let $TL(i) = \{T_1, T_2, \ldots.., T_n\}$ denote the set of cloudlets that are independent of each other scheduled on virtual machines (VM) $V(j) = \{V_1, V_2, \ldots\ldots, V_m\}$. Suppose a cloudlet $TL(i)$ is scheduled on a VM $V(j)$, execution time $exec(i,j)$ of a cloudlet executed by one VM $V(j)$ is calculated using Equation 1 [9].

$$exec(i,j) = \frac{TL\,(i)}{npe(j) \times V_{mips}(j)}, \qquad (1)$$

$$\forall\, i \in Task, i = \{1\}, \qquad j \in Vm, \qquad j = \{1\}$$

Where: $exec(i,j)$ is the execution time of running a single cloudlet on one virtual machine; $TL\ (i)$ is the length of a cloudlet in million instruction (MI); $V_{mips}(j)$is the VM processing speeds in million instructions per second (MIPS); $npe(j)$is the number of processing elements. When several VMs are involved in executing set of cloudlets, the total execution time $Texec(i,j)$ of all cloudlets executed on all VMs is calculated using Equation 2.

$$Texec(i,j) = \sum\left(\frac{TL(i)}{npe(j) \times V_{mips}(j)}\right) \qquad (2)$$

$$\forall\, i = \{1,2,\ldots,\mathrm{n}\}, j = \{1,2,\ldots\ldots,\mathrm{m}\}$$

$$makespan = min\left\{\max \sum\left(\frac{TL(i)}{npe(j) * V_{mips}(j)}\right)\right\} \qquad (3)$$

$\forall\, i = \{1,2,\dots,\mathrm{n}\}, j = \{1,2,\dots\dots,\mathrm{m}\}$

4. CAT SWARM OPTIMIZATION(CSO)

CSO is one among swarm optimization technique added to the family of swarm intelligence (SI) by [17]. The interesting behavior of cat enabled them to observe that cat has both resting and chasing behavior. This behavior is modeled into seeking and tracing mode. A control variable called the mixed ratio (MR) is used to define the position of the cat i.e., seeking or tracing mode.

4.1. Seeking Mode

The seeking mode, being a global search aspect of CSO defined cat behavior as per resting, looking around, at the same time deciding next position to move to [37]. This mode is shown in Algorithm 1.

Algorithm 1: Seeking Mode Process [9]

```
1. Generate Y (where Y = SMP) copies of k-th cat, i.e $Z_{qd}$ where (1≤ q≤Y) and (1≤d≤D) where D is
   the overall dimension.
2. Change at random the dimension of a cat by applying mutation operator to Zqd.
3. Determine the fitness values of all changed cats.
4. Discover best cats (non-dominant) based on their fitness values.
5. Replace the position of the k-th cat after picking a candidate at random from Y.
```

4.2. Tracing Mode

The tracing mode corresponds to local search [9],[37]. It is however presented as follow:

i. Compute and update cat k-*th* velocity using new velocity in Equation 4:

$$V_{K,d} = V_{K,d} + (c_1 \times r_1 \times (Xbest_d - X_{K,d})) \tag{4}$$

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

Where *c*; the constant value of acceleration, *r*; is the uniformly distribution random number in the range of [0, 1]. For each iteration, Equation 5 will be used to update the velocity.

ii. Add new velocity by computing the current (new) position of the *k-th* cat using Equation 5:

$$X_{k,d} = X_{k,d} + V_{k,d} \tag{5}$$

iii. Determine fitness values of all cats.

iv. Updates achieve contents with best cats.

5. PROPOSED CSO TRACING MODE (LOCAL SEARCH)

Our objective is to minimize makespan of total tasks scheduled across VMs in order to reduce task execution delay. As a result, an optimum task scheduling algorithm based on Taguchi is proposed [8],[13],[32]. On the other hand, CSO global search (seeking mode) and the local search (tracing mode) are carried out independently and that require a very high computation time [2], [9]. In order to overcome this, the tracing mode needs to be modified. The tracing mode operation of cat swarm is re-structured by applying Taguchi method as follow:

i. Generate two velocity sets:

$$Vo_{k,d}(t) = \begin{cases} Vset_{1_{K,d}}(t) = V_{K,d}(t-1) + (c_1 \times r_1 \times (Xgbest_d(t-1) - X_{K,d}(t-1)) \\ Vset_{2_{K,d}}(t) = V_{K,d}(t-1) + (c_1 \times r_1 \times (Xlbest_d(t-1) - X_{K,d}(t-1)) \end{cases} \tag{6}$$

Such that: $$Vo_{k,d}(t) = \begin{cases} Vset_{1_{k,d}}(t), & if\ orthogonal\ array\ element\ is\ \text{"1"} \\ Vset_{2_{k,d}}(t), & otherwise \end{cases} \tag{7}$$

Where: $Vo_{k,d}$ represents two candidate's velocity sets; d is dimension of the solution space; $Xgbest_d$ represents global best the position of the cat; $Xlbest_d$ is the local best position of the cat; r_1 represent uniform random number in the range of [0,1], c_1 is a constant value of acceleration; $X_{K,d}$ represent position of the cat and t, is the number of iteration. The size of orthogonal array is determined according to size of task from the generated two velocities, one is chosen to update the original velocity $V_{k,d}$ each time there is a run of the experiment according to Equation 8:

$$V_{k,d} = \begin{cases} max\ v\ ,\ if\ \left[V_{k,d} + Vo_{k,d}(t)\right] > maximum\ velocity, \\ V_{k,d} + Vo_{k,d},(t)\quad otherwise \end{cases} \tag{8}$$

ii. Add new velocity by computing the current (new) position of k-*th* cat using Equation 9.

$$X_{k,d} = X_{k,d} + V_{k,d} \tag{9}$$

iii. Determine fitness value of each cat.
iv. Sum the fitness of all cats according to their generated velocities, compare and select the final velocity to formulate the latest velocity.

6. TAGUCHI TASK SCHEDULING OPTIMIZATION

Orthogonal array based Taguchi approach is a good optimization method. The detail pseudocode of Taguchi method for our matrix experiment is presented in Algorithm 2 [32].

Algorithm 2: Taguchi Optimization Algorithm [9]

```
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 $\mathrm{Vset}_{1_{K,d}}(t)$ and $\mathrm{Vset}_{2_{K,d}}$ (t) according to Equation (6)
3. Determine dimension of scheduling problem which corresponds to task number N.
4. Calculate the fitness values of n experiments in accordance to the Orthogonal $L_n(2^{n-1})$ array.
```

The above algorithm is applied at tracing mode of cat swarm optimization (CSO) for minimization of makespan.

6.1. OTB-CSO Based Task Scheduling Algorithm

We developed our OTB-CSO based algorithm to solve the proposed task scheduling problem presented at Equation 3 using Algorithm 3 below.

Algorithm 3: OTB-CSO Algorithm

```
Start
    1.  Initialize associated position, cats' parameters; MR, mixed ratio; Y, the position of cats, velocity
        of cats and flag of every cat to distinguish cat into seeking and tracing mode.
    2.  Determine all require attributes such as virtual machine number, the number of processing
        elements, processing power to calculate cats' fitness function.
    3.  Compute all cats according to defined objective (Fitness) functions in Equation (3)
    4.  Compare fitness function of all cats and keep position with best fitness value.
    5.  Do
    6.        increment_iteration_number ← t + 1
    7.      If (seeking flag← 1)
    8.          Call algorithm 1 by applying seeking mode process
    9.  Else
    10.         Call algorithm 2 by applying tracing mode process based Taguchi approach
    11. Endif
    12.     Choose current best member as $Xl_{best}$ and corresponding best position as $X_{pbest}$
    13. If ($Xl_{best} > X_{gbest}$)
    14.     $Xl_{best} = X_{gbest}$
    15.     $X_{pbest} = G_{best\_id}$ // current best position becomes the global best position
    16.     Compute and update the new velocity and position according to (Equation (8) and (9))
    17. If (termination condition reached)
    18.   Output the position as the best task scheduling pattern (task sequence) that returns the best
        fitness (makespan).
    19. Else
    20. Go to step 6.
    21. Endif
    22. Endif
End.
```

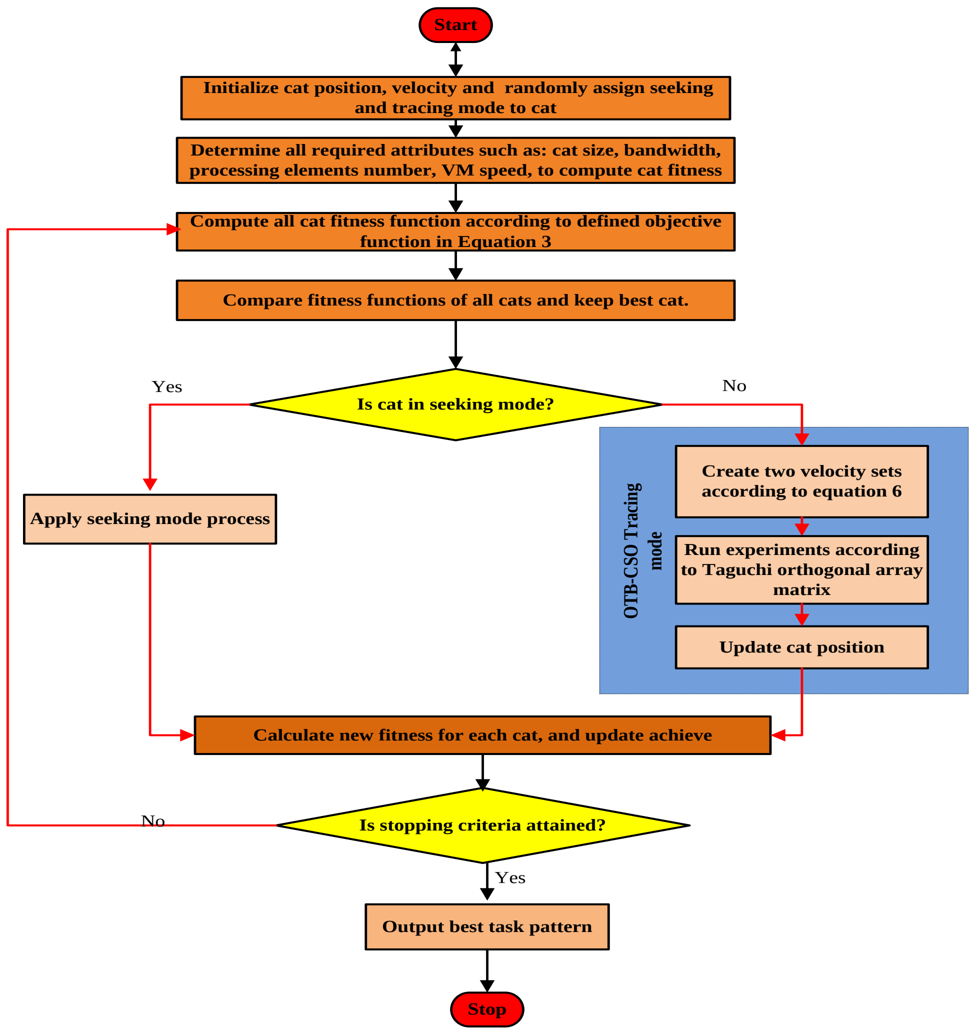
Start
Initialize cat position, velocity and randomly assign seeking and tracing mode to cat
Determine all required attributes such as: cat size, bandwidth, processing elements number, VM speed, to compute cat fitness
Compute all cat fitness function according to defined objective function in Equation 3
Compare fitness functions of all cats and keep best cat.
Is cat in seeking mode?
Yes
No
Apply seeking mode process
OTB-CSO Tracing mode
Create two velocity sets according to equation 6
Run experiments according to Taguchi orthogonal array matrix
Update cat position
Calculate new fitness for each cat, and update achieve
Is stopping criteria attained?
No
Yes
Output best task pattern
Stop

Figure 1. Flowchart of the OTB-CSO [9]

7. EXPERIMENTAL SETUP

The experiment conducted was carried out based on the following computer specifications (Processor: Intel® Core™ i5-5200U CPU@3.60 GHz 3.60GHz; System type: Window 10 (64-bit) x64-based processor; Memory: GB DDR3L RAM; Hard Disk: 1000 GB (1TB) SATA-3G HDD) and utility software (Eclipse-java-luna-SR2-win32-x86-64; Simulation tool: CloudSim 3.0.3) [9]. The choice of properties for both virtual machines (VMs), host, and tasks used for the experiment are based on [12] and [17], where Datacenter (No. of datacentre: 2; No. of host in a datacentre:1; Host RAM: 2GB; Storage: 1TB; Bandwidths:10GB/s; total host processing power: 1000000 MIPS), Cloudlets (Lengths: 100-1000 MIs; No. of Cloudlets: 10-100) and VMs (VMs: 20; VMs Monitor: Xen; Ram: 0.5GB; Storage: 10GB; Bandwidth: 1GB/s; VMs processing power: 1000-10000 MIPS; Processing element: 1 to 2; VM Policy: Time-shared) were used. The values of inertia weight and constriction factors (c1, c2) used are based on [38] as shown in Table 1.

Table 1. Parameter setting for PSO and CSO

Algorithm	Parameter	Value
PSO	Particle size	100
	Self-recognition coefficients (c1, c2)	2
	Uniform random number (*R1*)	[0,1]
	Maximum iteration	1000
	Inertia weight(*W*)	0.9-0.4
	Mixed ration	2%
CSO	Count Dimension to Change	5%

8. RESULTS

Ten (10) independent simulation runs with 1000 iterations were carried out on minimum and maximum job first (*Min-Max*) [15], particle swarm optimization with linear descending inertia weight (*PSO-LDIW*) [13], hybrid particle swarm optimization with simulated annealing (*HPSO-SA*) [12] and OTB-CSO algorithm for same size of input cloudlets(tasks) and 20VMs. The best, worst and average makespan was obtained and tabulated as shown in Table 2. Performance improvement (*PIR*) on makespan [10],[39] was illustrated in Table 3.

Table 2. Comparison of makespan obtained using 20 VMs

Task	Min-Max			PSO-LDIW			HPSO-SA			OTB-CSO		
	Best	Worst	Average	Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
10	14.77	47.07	30.66	14.23	32.90	20.80	6.60	21.52	14.90	7.62	21.49	13.53
20	31.17	59.74	46.68	23.04	37.20	35.01	12.59	48.39	34.26	10.01	38.94	29.15
30	48.07	72.02	65.32	42.18	82.51	51.04	26.54	60.57	39.79	32.76	40.76	37.57
40	79.35	147.54	99.41	51.09	104.95	89.66	49.66	66.86	54.70	45.55	59.76	54.04
50	104.57	264.13	149.74	111.29	212.71	148.73	65.76	111.53	95.16	62.45	101.14	88.96
60	186.75	275.98	224.15	176.53	366.20	222.56	97.79	164.09	132.97	89.51	165.51	129.06
70	249.68	414.22	319.13	146.84	435.72	307.02	138.45	241.01	176.21	109.71	183.81	174.98
80	386.22	686.22	447.44	158.80	477.95	331.60	186.96	322.03	203.18	154.84	252.37	201.69
90	452.25	831.66	495.27	252.16	519.89	367.36	245.29	443.03	264.53	213.74	289.62	252.48
100	530.29	998.66	691.39	384.23	942.97	669.88	367.54	573.86	489.27	302.54	605.31	486.57

Table 3. OTB-CSO performance improvement (%) based on Makespan

	Total Average makespan	Min-Max	PSO-LDIW	HPSO-SA	OTB-CSO
		2569.19	2243.66	1504.97	1468.03
20 VMs	*PIR (%)Over I-Min-Max*		12.67	41.42	42.86
	PIR (%) Over PSO-LDIW			32.92	34.57
	PIR (%) Over HPSO-SA				2.45

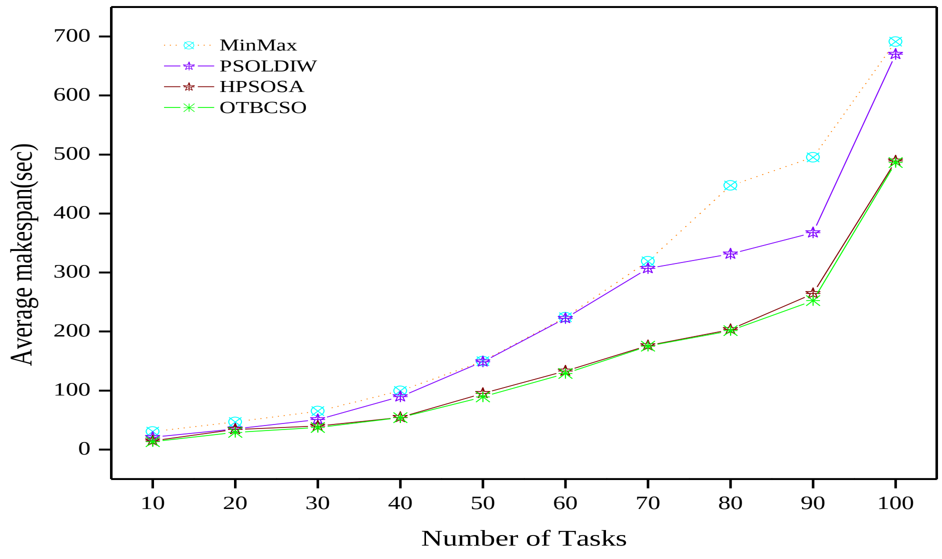

Figure 2. Graph of makespan obtained

9. DISCUSSION

For each problem size with 1000 iterations, 10 independent simulation runs were carried out based on input tasks 10-100 on Min-Max, PSO-LDIW, HPSO-SA, and OTB-CSO algorithm. The makespan and performance improvement rate obtained by the four algorithms are reported in Table 2 and Table 3. Figure 2 showed a graph of average makespan achieved by proposed OTB-CSO algorithm as compared to the three existing algorithms. Table 3 unveiled performance improvement achieved using 20 VMs. The performance improvements obtained based on makespan are 42.86%, 34.52%, 2.45% over Min-Max, PSO-LDIW and HPSO-SA algorithms. Although HPSO-SA outperformed Min-Min and PSO-LDIW, it showed some robustness, where OTB-CSO outperformed the HPSO-SA with only 2.45%. The outlined performance of OTB-CSO over existing algorithms was attributed to the incorporation of Taguchi Orthogonal approach at CSO tracing mode. The Taguchi method is a facilitator, which enables search process within tracing mode of the CSO traversed the best solution regions. It helps in guiding our algorithm toward achieving a good solution. However, our proposed OTB-CSO tracing mode has actually utilized an Orthogonal array of Taguchi method to return search results more suitable by enabling searching efficiency improved [23],[35]. Obtained results have significantly shown how our proposed algorithm outperformed better by achieving minimum makespan. The improvement (%) showed the reduction in execution time achieved by our proposed algorithm. This shows OTB-CSO is effective to optimize task scheduling with better quality of service provisioning in cloud computing.

10. CONCLUSION

In this paper, we presented an Orthogonal Taguchi Based-cat swarm optimization (OTB-CSO) algorithm for optimum task scheduling that reduced task execution delay in a dynamic cloud computing environment. Our proposed OTB-CSO explored local search ability of Taguchi optimization method to improve the speed of convergence and quality of solution by achieving minimum makespan. The experimental results showed proposed OTB-CSO outperformed Min-Max, PSO-LDIW, and HPSO-SA in minimizing the task makespan on VMs. A more study of other computation-based and network-based parameters is required with the integration of more advanced concepts such as virtual machine migration, energy consumption and to optimize further the algorithm in order to scale with larger workloads is require for further confirm the performance of the proposed algorithm.

ACKNOWLEDGEMENTS

The author will like to acknowledge the support of the Co-authors for their significant contributions toward making this research a success and also the financial support rendered by Tertiary Education Trust Fund (TETFund) Nigeria in carrying out this research.

REFERENCES

[1] G. Aceto, *et al.*, "Cloud monitoring: A survey," *Computer Networks*, vol/issue: 57(2013), pp. 2093-2115, 2013.
[2] S. Banerjee, *et al.*, "Development and Analysis of a New Cloudlet Allocation Strategy for QoS Improvement in Cloud," *Arab Journal of Science and Engineering*, vol/issue: 40(5), pp. 1409-1425, 2015.
[3] K. B. Bey, *et al.*, "Balancing Heuristic for Independent Task Scheduling in Cloud Computing," *in proceedings of 12th International Symposium on Programming and Systems (ISPS)*, pp. 1– 6, 2015.
[4] G. S. Domanal and G. R. M Reddy, "Optimal Load Balancing in Cloud Computing by Efficient Utilization of Virtual Machines," in *proceedings of the Sixth International Conference on Communication Systems and Networking (COMSNETS)*, pp. 1-4, 2014.
[5] F. Durao, *et al.*, "Systematic Review on Cloud Computing," *Journal of Supercomputing*, vol. 68, pp.1321-1346, 2014.
[6] A. V. Lakra and D. K. Yadav, "Multi-Objective Task Scheduling Algorithm for Cloud Computing Throughput Optimization," *Procedia Computer Science Journal*, vol. 48, pp. 107-113, 2015.
[7] V. A. Leena, *et al.*, "Genetic Algorithm Based Bi-Objective Task Scheduling in Hybrid Cloud Platform," *International Journal of Computer Theory and Engineering*, vol/issue: 8(1), pp. 7-13, 2016.
[8] Z. Zhou and H. Zhigang, "Task Scheduling Algorithm based on Greedy Strategy in Cloud Computing," *The Open Cybernetics & Systemics Journal*, vol. 8, pp. 111-114, 2014.
[9] D. Gabi, *et al.*, "Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing," *Neural Comput & Applic*, 2016.
[10] M. Abdullahi, *et al.*, "Symbiotic Organism Search optimization based task scheduling in cloud computing environment," *Future Generation Computer Systems*, vol/issue: 56(2016), pp. 640-650, 2016.
[11] C. W Tsai, *et al.*, "A Hyper-Heuristic Scheduling Algorithm for Cloud," *IEEE Transactions on Cloud Computing*, vol/issue: 2(2), pp. 236–250, 2014.
[12] H. S. Al-Olimat, *et al.*, "Cloudlet Scheduling with Particle Swarm Optimization," in *proceedings of the 2015 Fifth International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 991 – 995, 2015.
[13] U. Bhoi and N. Ramanuj, "Enhanced Max-min Task Scheduling Algorithm in Cloud Computing," *International Journal of Application or Innovation in Engineering and Management*, vol/issue: 2(4), pp. 259-264, 2013.
[14] R. Xu, *et al.*, "Makespan Minimization on Single Batch-processing Machine via Ant Colony Optimization," *Computers & Operations Research*, vol. 39, pp. 582-593, 2012.
[15] Z. Zhou and H. Zhigang, "Task Scheduling Algorithm based on Greedy Strategy in Cloud Computing," *The Open Cybernetics & Systemics Journal*, vol. 8, pp. 111-114, 2014.
[16] M. R. Garey and D. S. A Johnson, "Guide to the Theory of NP-Completeness," New York, WH Freeman, 2016.

[17] S. C. Chu and P. W. Tsai, “Computational intelligence based on the behavior of cats,” *International Journal of Innovative Computing, Information, and Control*, vol/issue: 3(2007), pp. 163-173, 2007.
[18] A. I. Awad, *et al.*, “Dynamic Multi-Objective Task Scheduling in Cloud Computing Based on Modified Particle Swarm Optimization,” *Advances in Computer Science: An International Journal*, vol/issue: 4(5), pp. 110-117, 2015.
[19] B, L. D. Dhinesh and P. V. Krishna, “Honey Bee Behavior Inspired Load Balancing of Tasks in Cloud Computing Environments,” *Journal of Applied Soft Computing*, vol/issue: 13(5), pp. 2292-2303, 2013.
[20] A. B. El-Sisi, *et al.*, “Intelligent Method for Cloud Scheduling Based on Particle Swarm Optimization Algorithm,” in *proceedings of the International Arab Conference On Information Technology (Acit2014)*, pp. 39-44, 2014.
[21] R. K. Jena, “Multi-objective Task Scheduling in Cloud Environment Using Nested PSO Framework,” *Procedia Computer Science Journal*, vol/issue: 57(2015), pp.1219-1227, 2015.
[22] F. Ramezani, *et al.*, “Evolutionary algorithm-based Multi-Objective Task Scheduling Optimization Model in Cloud Environments,” *Springer Science-Business-Media, New York*, vol. 18, pp. 1737-1757, 2015.
[23] S. Singh and M. Kalra, “Scheduling of Independent Tasks in Cloud Computing Using Modified Genetic Algorithm,” in *proceedings of the Sixth International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 565-569, 2014.
[24] Z. Wu, *et al.*, “A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling,” in *proceedings of the International Conference on Computational Intelligence and Security (CIS)*. 11-14 December. Nanning, China, pp. 184-188, 2010.
[25] S. C. Chu and P. W. Tsai, “Computational intelligence based on the behavior of cats,” *International Journal of Innovative Computing, Information, and Control*, vol/issue: 3(2007), pp. 163-173, 2007.
[26] P. M. Pradhan and G. Panda, “Solving Multi-objective problems using cat swarm optimization,” *An international Journal of Expert System with Application*, vol. 39, pp. 2956-2964, 2012.
[27] G. Taguchi, *et al.*, “Robust Engineering,” New York, McGraw-Hill, 2000.
[28] U. Bhoi and N. Ramanuj, “Enhanced Max-min Task Scheduling Algorithm in Cloud Computing,” *International Journal of Application or Innovation in Engineering and Management*, vol/issue: 2(4), pp. 259-264, 2013
[29] K. Abd, *et al.*, “Simulation modelling and Analysis of Scheduling in Robotic Flexible Assembly Cells using Taguchi Method,” *International Journal of Production Research*, vol/issue: 52(9) pp.2654-2666. 2013.
[30] G. R. Cavory, *et al.*, “A genetic approach to the scheduling of preventive maintenance tasks on a single product manufacturing production line,” *International journal of Production Economics*, vol. 74, pp. 135-146, 2001.
[31] H. Asefi, *et al.*, “A Hybrid NSGA-II and VNS for Solving a Bi-Objective No-Wait Flexible Flowshop Scheduling Problem,” *International Journal of Advance Manufacturing Technology*, vol. 75, pp. 1017–1033, 2014.
[32] H. C. Chang, *et al.*, “Solving the Flexible Job Shop Scheduling Problem with Makespan Optimization by Using a Hybrid Taguchi-Genetic Algorithm,” *IEEE Journals & Magazines*, vol. 3, pp. 1740-1754, 2015.
[33] R. Caprilhan, *et al.*, “Evaluation of the Impact of Information Delays on Flexible Manufacturing Systems performance in Dynamic Scheduling Environments,” *The International Journal of Advanced Manufacturing Technology*, vol/issue: 67(1), pp. 311-338, 2013.
[34] J. T Tsai, *et al.*, “An Improved genetic algorithm for job-shop scheduling problems using a Taguchi-based crossover,” *International Journal of Advance Manufacturing Technology*, vol. 38, pp. 987–994, 2008.
[35] S. Bilgaiyan, *et al.*, “A Multi-Objective Cat Swarm Optimization Algorithm for Workflow Scheduling in Cloud Computing Environment,” *International Journal of Soft Computing*, vol/issue: 10(1) pp. 37-45, 2015.
[36] F. Ramezani, *et al.*, “Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization. *International Journal of Parallel Programming*, vol/issue: 42(2014), pp. 739-754, 2013.
[37] R. Shojaee, *et al.*, “A New Cat Swarm Optimization based Algorithm for Reliability-Oriented Task Allocation in Distributed Systems,” in a *symposium on Sixth International Telecommunications (IST)*, pp. 861-866, 2012.
[38] R. C. Eberhart and Y. Shi, “Comparing Inertia Weights and Constriction factors in particle swarm optimization,” in *proceedings of the IEEE Conference on Evolutionary Computation, ICEC*, pp. 84-88, 2000.
[39] M. S. Abdulhamid, *et al.*, “Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm,” *Neural Comput & Applic*, 2016.

BIOGRAPHIES OF AUTHORS

DANLAMI GABI is currently pursuing his Ph.D. in Computer Science at Universiti Teknologi Malaysia. He obtained his BSc in Computer at Usmanu Danfodio University, Sokoto, Nigeria. Later, he proceeded to obtain his MSc. in Information Security and Computer Forensics at the University of East London, United Kingdom. He is a lecturer in Kebbi State University of Science and Technology Aliero. His research interests include complex algorithm design for distributed systems, cloud computing, and energy efficient computation.

ABDUL SAMAD ISMAIL received Ph.D. in Computer Science from Aston University, Birmingham UK, MSc Computer Science in Central Michigan University Mt. Pleasant, Michigan, USA, and BSc. Math and Computer Science at University of Wisconsin Superior, Wisconsin, USA. His research interests are related to Wireless Computer, Cloud Computing, and Network Security. Currently, he is a Professor at Universiti Teknologi Malaysia and Dean of the Faculty of Computing.

ANAZIDA ZAINAL received her Ph.D. in Computer Science from Universiti Teknologi Malaysia, Skudai Johor, Malaysia. MSc. in Computer Science from Universiti Teknologi Malaysia, and BSc. Computer Science at Rutgers University, New Jersey, US. Her research interests are Network Security, Intrusion Detection System, Intrusion Prevention System and Soft Computing. Currently, she is a lecturer at Universiti Teknologi Malaysia.

ZALMIYA ZAKARIA received her Ph.D. in Computer Science from Universiti Teknologi Malaysia, Skudai Johor, Malaysia. MSc.in Computer Science from Universiti Teknologi Malaysia and BSc. Computer Science from Universiti Teknologi Malaysia. Her research interests are Optimization in planning, scheduling and timetabling, Machine learning and automated reasoning, Intelligent system design and development, Artificial Intelligence Programming (Prolog and LISP). Currently, she is a lecturer at Universiti Teknologi Malaysia.