Journal of Intelligent Systems and Internet of Things
Vol. 12, No. 01, PP. 45-56, 2024
DOI: https://doi.org/10.54216/JISIoT.120104

Energy Efficient Task Scheduling Strategy using Modified Coot Optimization Algorithm for Cloud Computing

Kandan M.[1,*], M. Mutharasu[2], Siva Satya Sreedhar P.[3], S. Thenappan[4], G. Nagarajan[5]

[1]Department of Computing Technologies, School of Computing, Faculty of Engineering and Technology, SRM Institute of Science and Technology, Kattankulathur 603203, Tamil Nadu, India
[2]Department Of C.S.E (Cyber Security), Madanapalle Institute Of Technology & Science, Kadiri Road, Angallu Madanapalle, Andhrapradesh, 517325, India
[3]Department of Information Technology, Seshadri Rao Gudlavalleru Engineering College, Gudlavalleru, Krishna District, Andhra Pradesh, 521356, India
[4]Department of Electronics and Communication Engineering, Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology, Chennai, India
[5]Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education, Krishnan Koil, Tamil Nadu, India.
Emails: kandan.win25@gmail.com; mutharasum@mits.ac.in; sivasatyasreedhar@gmail.com; drthenappans@veltech.edu.in; nagarajanite@gmail.com
[*]Corresponding Author: kandan.win25@gmail.com

Abstract

Cloud computing (CC) refers to a current computing method that provides the virtualization of computing services as a utility to Cloud service users. Problems based on ineffective task mapping to cloud resource frequently happen in a cloud atmosphere. Task scheduling (TS), thus, means effective scheduling of rational allocation and computational actions of computing resource in certain limitations in the IaaS cloud network. Job scheduling was to allocate tasks to the most appropriate sources to reach more than one goal. Thus, choosing a suitable work scheduling technique for rising CC resource efficiency, whereas maintaining high quality of service (QoS) assurances, becomes a significant problem that remains to attract interest of researchers. Metaheuristic techniques shown remarkable efficacy in supplying near-optimal scheduling solutions for a complicated large-sized issues. Recently, a rising number of independent scholar has examined the QoS rendered by TS approaches. Therefore, this study develops an Energy Efficient Task Scheduling Strategy using Modified Coot Optimization Algorithm (EETSS-MCOA) for CC environment. The EETSS-MCOA method carries out the derivation of features and MCOA is applied to schedule tasks. In addition, the MCOA algorithm is derived by the combination of adaptive β hill climbing concept with the COA for enhanced task scheduling. The conventional COA is stimulated by the swarming characteristics of birds known as coots. The COA followed two distinct stages of bird movements on water surface. The experimental results of the EETSS-MCOA model are validated on CloudSim tool. The solutions attained by the EETSS-MCOA model are found to be better than the existing algorithms.

Received: August 17, 2023 Revised: November 22, 2023 Accepted: February 22, 2024

Keywords: Cloud computing; Task scheduling; Metaheuristic algorithms; Quality of service; Coot optimization algorithm

1. Introduction:

Cloud computing (CC) becomes the latest computing method that presents the computing services virtualization to Cloud service users [1,2]. It is an idea to acquire sources from a flexible shared resource, namely a set of applications,

networks, utilities, servers, and storage immediately and demand on request. Cloud service providers utilize virtualized technology for using resources well by permitting many virtual machines (VMs) to work on topmost area of one computer [3]. Cloud services users were mechanically provision related to Service-Level Agreements (SLA) that were generally constituted by negotiations among Cloud service users and Cloud service providers [4]. Problems based on ineffective task mapping to cloud sources frequently happen in a cloud atmosphere. Thus, Task scheduling (TS) effectively scheduling the computational acts and rational allotment of computing sources has certain limitations in the IaaS cloud setting. Fig. 1 depicts the process of TS in CC environment.

Job Scheduling is assigning tasks to the most appropriate sources for achieving several goals [5]. Thus, choosing a suitable work scheduling method for using CC resource effectiveness, whereas maintaining high quality of service (QoS) guarantees, becomes a significant problem which attracts the interest of researchers [6, 7]. In accordance with the wide solution spaces and the complexity presence of heterogeneous sources in CC, the TS issue falls to the NP-hard issues group [8]. Metaheuristic and heuristic scheduling approaches were employed for addressing the TS issue in CC [9]. Heuristic techniques offer best outcomes, however, the size of the issue increases, solution produced by such methods will be less optimal. Conversely, metaheuristic methods depicted impressive efficiency in bringing near optimal scheduling solutions for complicated larger sized issues [10].

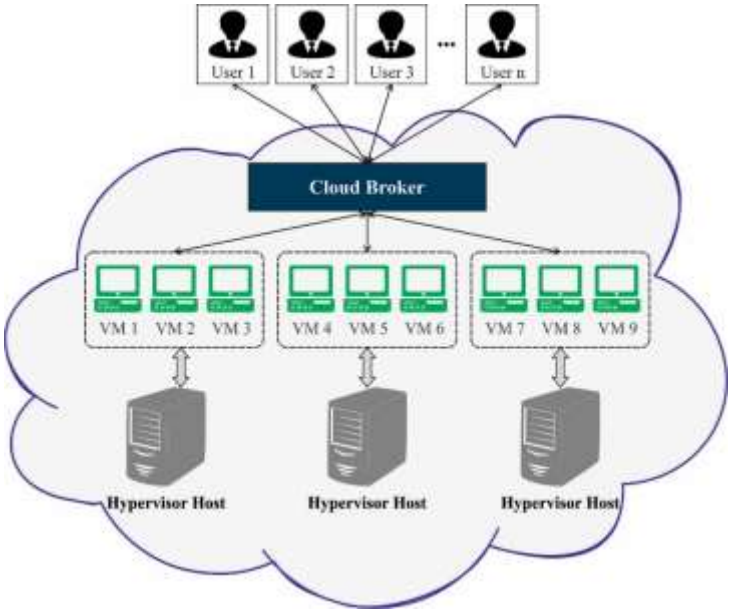

Figure 1: Task scheduling in CC environment

Medara and Singh [11] present an energy-effective and reliability aware workflow TS in a cloud atmosphere (EERS) technique that preserves energy and optimizes the system dependability. The EERS has 5 sub-techniques. Initially, author applied a task rank calculation technique for preserving task dependency. Secondly, a task cluster approach for reducing transmission costs that minimizes energy utilization. Then the sub-target time distributing technique for defining sub_makespan for every task. Huang et al. [12] introducers a TS with number of discrete variants of the PSO technique for TS in CC. For evaluating the performances, such techniques are compared with 3 renowned heuristic techniques on TS perplexities.

In [13], an altered Henry gas solubility optimization (HGSO) was proposed depends on comprehensive opposition-based learning (COBL) and the WOA for optimal TS. The presented technique was known as HGSWC. In this work, WOA can be employed as a local search process for the betterment of the quality of solutions, where COBL has been applied for fostering the poor solutions by calculating the opposite solution and chosing best among them. Velliangiri et al. [14] devised Hybrid Electro Search with genetic algorithm (HESGA) for improving the TS conducts through considerations of variables like cost of the multi-cloud, makespan, utility of resources, and load balancing. The devised technique compiled the benefit of an electro search algorithm and GA. The GA offers the finest local optimal solution, where the Electro search technique presents the finest global optimal solution.

In [15], the first- ever, author applied the modern meta-heuristics WOA for cloud TS with multiobjective optimizing technique, aims to improve the presentation of a cloud mechanism with computing sources. On that basis, author presents a new technique named Improved WOA for Cloud TS (IWC) to enhance the optimum solution search ability of the WOA-related technique. Najafizadeh et al. [16] suggest multiobjective simulated annealing (MOSA) technique for allocating tasks steadily on the fog and cloud nodes related to deadline limits. The Goal Programming Approach (GPA) can be implemented for finding a compromised solution that satisfies many goals. Likewise, concerning the distribution of IoT task cloud nodes and among fog, an innovative goal was made named scheduling and access level based on users' demand.

This study develops an Energy Efficient Task Scheduling Strategy using Modified Coot Optimization Algorithm (EETSS-MCOA) for CC environment. The EETSS-MCOA method carries out the derivation of features and MCOA is applied to schedule tasks. In addition, the MCOA algorithm is derived by the combination of adaptive β hill climbing concept with the COA for enhanced task scheduling. The conventional COA is stimulated from the swarming characteristics of birds known as coots. The COA followed two distinct stages of bird movements on water surface. The experimental results of the EETSS-MCOA model are validated on CloudSim tool.

2. The Proposed Model

In this study, a new EETSS-MCOA method was formulated for scheduling tasks in the cloud environment. The EETSS-MCOA approach carries out the derivation of features and MCOA is applied to schedule tasks.

A. Algorithmic Design of MCOA Technique

COA is a meta-heuristic technique dependent upon the performance of coots if it can be considered for foods naturally. The COA has divided the population into leaders and members whereas leaders are demonstrated by higher quality solutions and members lower quality solutions [17]. The 3 important stages of COA are formulated as:

1. Generation of Initial Solution Set

All the coots k from the population were assumed as a solution A_k, and A_k refers the created as follows:

$$A_k = A^{\min} + r_1(A^{\max} - A^{\min}); i = 1, \ldots, N_{Po}, \quad (1)$$

whereas A^{ax} and $A^{\min}$ were upper as well as lower bounds of all the present solutions correspondingly, and r_1 random number from the range of zero and onr.

In Eq. (1), N_{Po} represents the size of populations, and it can be separated as 2 groups, optimum group with N_L leader and bad group with N_M members. All the individuals from the population were estimated by computing fitness function (FF), afterward, all the individuals are placed in an appropriate group dependent upon their achieved fitness value. All the solutions from the leader group were demonstrated by Ld_m, whereas $m = 1, \ldots, N_L$, and all the solutions from the member groups were demonstrated by Mr_j, whereas $j = 1, \ldots, N_M$. Fig. 2 showcases the flowchart of COA.

1. Update of Member Group

All the solutions from the member group were upgraded by utilizing one out of 3 subsequent techniques:

$$\begin{aligned} Mr_j^{new} &= Mr_j + FG_1.r_2.(A_{rd} - Mr_j), \\ Mr_j^{new} &= 0.5 \times \left(\left(Mr_{j-1} + Mr_j\right)\right), \\ Mr_j^{new} &= Ld_{rd} + 2.r_3.\cos(2.\pi.r').(Ld_{rd} - Mr_j), \end{aligned} \quad (2)$$

whereas r_2 and r_3 signifies the arbitrary numbers in $zero$ and one, but r' represents the arbitrary number in -1 and 1. Ld_{rd} stands for arbitrarily picked solutions from the leader group. A_{rd} and FG_1 are the arbitrary solution and function of iteration achieved as:

$$\begin{aligned} A_{rd} &= A^{\min} + r_4(A^{\max} - A^{\min}), \\ FG_1 &= 1 - G \times \left(\frac{1}{G^{\max}}\right), \end{aligned} \quad (3)$$

In which r_4 refers the arbitrary number in $zero$ and one. G and $G^{\max}$ signifies the present iteration and maximal iteration number correspondingly.

2. Update for Leader Group

COA upgrades solutions from the optimum solution of populations, whereas Ld_{best}. leader group by utilizing local search, viz., search around 2 distinct approaches were utilized to the group as:

$$\begin{cases} Ld_m^{new} = Ld_{best} + [FG_2.r_5.cos(2.\pi.r').(Ld_{best} - Ld_m)] if r_6 < 0.5 \\ Ld_m^{new} = -Ld_{best} + [FG_2.r_7.cos(2.\pi.r').(Ld_{best} - Ld_m)] if r_6 \geq 0.5' \end{cases} \quad (4)$$

whereas r_5, r_6, and r_7 demonstrates the arbitrary numbers amongst one and zero. FG_2 refers the function of iterations formulated as:

$$FG_2 = 2 - \left(\frac{1}{G\max}\right) G. \tag{5}$$

The MCOA algorithm is derived by the combination of adaptive β hill climbing concept. Adaptive β-hill climbing (Aβ-HC) is a state-of-the-art projected local search-based system that is fundamentally an improved form of β-hill climbing (βHC) [18]. An investigation is established that $A\beta HC$ offers optimum efficiency to number of another famous local search approaches. To enhance the approaches exploitation abilities and quality of end solutions, $A\beta HC$ was integrated as an important COA in support of seeking the neighborhoods for optimum solutions under this study. For providing present solution $X_i = (x_{i,1}, x_{i,2}, \dots, x_{i,D})$, $A\beta HC$ was iteratively created an enhanced solutions $X_i'' = (x_{i,1}'', x_{i,2}'', \dots, x_{i,D}'')$ based on the 2 control operators such as $\mathcal{N}$- and β-operators. The N-operator primary transmissions X_i to a new neighborhood solution $X_i' = (x_{i,1}', x_{i,2}', \dots, x_{i,D}')$ that is determined in Eqs. (6) and (7) as:

$$x_{i,j}' = x_{i,j} \pm U(0,1) \times \mathcal{N}, j = 1,2, \dots, D \tag{6}$$

$$\mathcal{N}(t) = 1 - \frac{t^{\frac{1}{K}}}{\text{Max } iter^{\frac{1}{K}}} \tag{7}$$

whereas $U(O,1)$ implies an random value within [0,1], x_{ij} implies the value of decision variables in the j^{th} dimensional, t signifies the present iteration, $Maxiter$ indicates the maximal amount of iterations, N stands for the bandwidth distance among the present solution and its neighbor, D indicates the spatial dimensional, and the parameter K is constant:

$$x_{i,j}' \leftarrow \begin{cases} x_{i,r}, & if r_8 < \beta \\ x_{i,j}', & else \end{cases} \tag{8}$$

$$\beta(t) = \beta_{\min} + (\beta_{\max} - \beta_{\min}) \times \frac{t}{\text{Max } iter} \tag{9}$$

In which r_8 denotes the random value within [0,1], $x_{i,r}$ refers the other arbitrary number selected in the probable range that specific dimensional of problems, the maximal and minimal values of probability value $\beta \in [0,1]$ are $\beta_{\max}$ and $\beta_{\min}$, correspondingly. When the created solution X_i'' is superior to the present solution in consideration X_i, afterward X_i is exchanged by X_i''. The pseudocode of adaptive β-hill climbing is provided in Algorithm 1.

Algorithm 1: Adaptive β-hill climbing algorithm
Initializing the parameters $\beta_{\max}$, $\beta_{\min}$, and K
Input the existing solution $X_i = (x_{i,1}, x_{i,2}, \dots, x_{i,D})$
Compute the fitness value $F(X_i)$
While $t \leq$Maxiter do
Make the neighbouring solution X_i' utilizing in Eq. (6)
For $j = 1$ to D do
If $r_8 <$ then
$x_{i,j} = x_{i,r}$
Else
$x_{i,j}'' = x_{i,j}'$
End If
End For
Measured the fitness value $F(X_i'')$
If $F(X_i'') < F(X_i)$ then
$X_i = X_i''$
End If
$t = t + 1$
End while

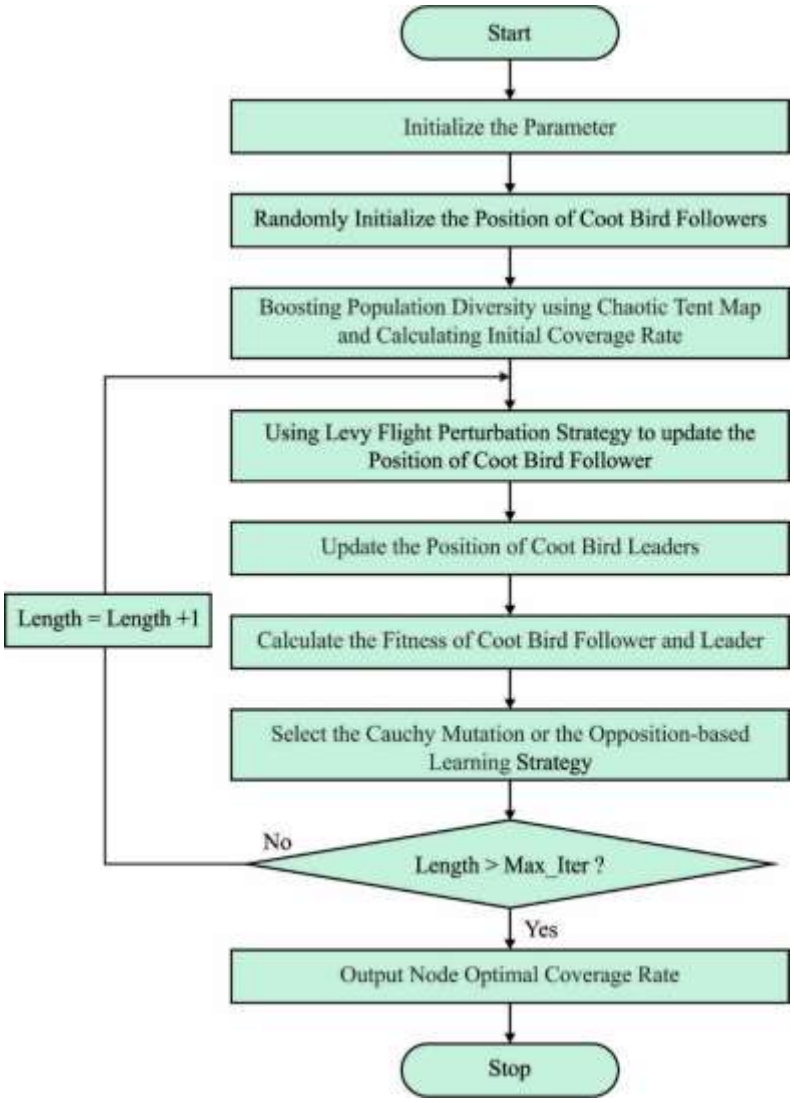

Figure 2: Flowchart of COA

B. Task Scheduling Process

In the EETSS-MCOA technique, user several tasks were dynamically allocated to wanted virtual machine (VM) or cloud resource [19]. Task scheduling (TS) is led to cloud broker (CB) with cloud user has queried the cloud information service (CIS), the register comprising the datacenter details assuming that the accessible services which are utilized for performing the job. It can be considered that tasks $T = \{T^1, T^2, T^3, \dots, T^n\}$ were established by CB from a certain time interval and processing elements (VMs) that are heterogeneous naturally due to its different processing power, so enhancing the makespan of particular VMs but decreasing the makespan of others. The cost of processing task from all the VMs differs during the sense. Supposing there are m VMs$\{VM^1, VM^2, VM^3, .../VM^m\}$ at the datacenter if the CB receives the tasks. An initial drive of this effort for identifying an optimum schedule to implement a set of tasks from VMs under shortest possible delay, but enhancing the percentage of source consumption, for instance, to TS on accessible VMs for obtaining the least makespan with higher resource consumption. Therefore, the presented technique makes use of the Expected Time to Compute (ETC) of jobs that scheduling on all the VMs for determining scheduling decisions. According to the millions of instructions per second (MIPS) of VMs and the task time or length, ETC values were estimated. ETC matrix is $(n \times m)$ that retains track of extensive all the task obtained, for instance, T^i on every VM^j. It can be measured as the ratio of Task lengths to VM ability. Consider $ETC^{i,j} i = 1,2,3, \dots, n, j = 1,2,3, \dots, m$ as a processing time of implementing task T^i on all the VM^j. It can be ratio of

task length T^j measured from the MI (millions of instructions) to speed of VM^j measured from MIPS. It can be assumed as:

$$ETC^{i,j} = Lenth\ of\ Task T^i / capacity\ of\ VM^j = MI^i / MIPS^j \quad (10)$$

Consider the task T^i start time is $ST(T^i, VM^j)$ on VM^j. The existing time of VM^j is represented by push (VM^j). Eqs. (8) and (9) determine the initial and end time of tasks T^i on VM^j.

$$ST(T^i, VM^j) = \max\{push(VM^j)\} \quad (11)$$

Also, consider the time of tasks T^i be $PT(T^i, VM^j)$ on VM^j finish.

$$FT(T^i, VM^j) = ST(T^i, VM^j) + ETC^{i,j} \quad (12)$$

Eq. (13) that measure the strength of organism degree of the adaptation to ecosystem was utilized for calculating the fitness value of all the organisms.

$$Objective\ Function = \max\{\sum_{j=1}^{m} \frac{f(VM^j)}{m}\} \quad (13)$$

$$f(VM^j) = \frac{z}{makespan} \quad (14)$$

$$z = \sum_{j=1}^{m} \frac{r^j}{m} \quad (15)$$

$$r^j = \frac{Task T^i}{makespan} \quad (16)$$

$$Makespan = max\{ETC^{i,j} i \in T, i = 1,2,3,\dots,n\ and\ j \in VM, j = 1,2,3,\dots,m\} \quad (17)$$

But Eq. (14) implies the VM^j, s fitness value, Eq. (15) depicts the average usage of VM utilized to task implementation signified by X.

The consumption of VM^j is determined by r^j, as shown in Eq. (16). Eq. (17) determines the maximal time VMs work from the parallel to achieve task.

The Degree of Imbalance (DOI) projected in Eq. (18) was the load distributed amongst the entire count of VMs based on its computational abilities. DOI is computed utilizing the subsequent formula. At this point, $T^{\max}$, $T^{\min}$ and $T^{average}$ denotes the maximal, minimal, and average implementation times amongst every VM. The implementation time of VMs was the entire time that VM was busy.

$$DOI = \frac{T^{\max} - T^{\min}}{T^{average}} \quad (18)$$

Assume $\{Pc^1, Pc^2, Pc^3, \dots, Pc^m\}$ define the unit cost of VM^j per time quantum. The unit cost of implementing a task T^j on VM^j under this study can be regarded per second basis.

An entire cost of processing tasks T^i with T^n on the accessible VM^j with VM^m is represented in Eq. (19) as:

$$TCost = \sum_{i=1,j=1}^{n,m} \{C^{i,j} * Pc^j\} \quad (19)$$

The cost of data broadcast and retrieval were unnoticed as the tasks were independent during this case.

The cost matrix is a $1 \times n$ matrix which comprises all the VMs costs. The values of ETC matrix and cost matrix were normalization by separating them with its corresponding maximal values.

Thus, as the VMs are heterogeneous naturally, the related cost of utilizing all the VMs differs. The response time of processors or machines was the time interval amongst the task submission duration and beginning of task executions. It can be total variance amongst submission period and waiting or delay duration.

$$Response\ Time = (Start\ time - Submission\ time)$$

So, an entire response time represented by $TResp^{Time}$ is demonstrated in Eq. (20) and is computed as:

$$TResp^{Time} = \sum_{i=1}^{n} (Str^{Time T^i} - Sub^{Time T^j}) \quad (20)$$

The purpose of scheduling problems, based on the important features of TS in CC context, is for mapping all the tasks T^i dynamically $i = 1,2,3,\dots,n$ to appropriate cloud VM^j $j = 1,2,3,\dots,m$ for reducing entire execution time, response time, and cost, maximizing resource utilization and minimizing the DOI amongst the scheduling VMs.

3. Results and Discussion

In this section, the TS performance of EETSS-MCOA method is examined in detail. Table 1 and Fig. 3 report a comparative makespan (MKS) inspection of the EETSS-MCOA model under several tasks [20-22]. The experimental outcomes reported that the EETSS-MCOA model has shown improved outcomes with minimal values of MKS. For instance, with 100 tasks, the EETSS-MCOA model offered minimal MKS value of 27 whereas the PSO, SOS, and G_SOS models obtained increased MKS values of 69, 56, and 48 respectively.

Table 1: MKS analysis of EETSS-MCOA approach with existing algorithms under several tasks

Makespan				
Number of Tasks	PSO	SOS	G_SOS	EETSS-MCOA
100	69	56	48	27
200	114	112	96	64
300	191	191	183	128
400	263	257	236	181
500	371	329	313	242
600	472	454	387	311
700	578	536	464	379
800	671	642	509	435
900	801	753	615	523
1000	920	841	687	554

In the meantime, with 200 tasks, the EETSS-MCOA technique has presented reduced MKS value of 64 whereas the PSO, SOS, and G_SOS algorithms have gained increased MKS values of 114, 112, and 96 correspondingly. Moreover, with 300 tasks, the EETSS-MCOA technique has provided lesser MKS value of 128 whereas the PSO, SOS, and G_SOS approaches have gained increased MKS values of 191, 191, and 183 correspondingly. Eventually, with 400 tasks, the EETSS-MCOA methodology has granted least MKS value of 181 whereas the PSO, SOS, and G_SOS approaches have reached increased MKS values of 263, 257, and 236 correspondingly.

Table 2: Cost analysis of EETSS-MCOA approach with existing algorithms under several tasks

Cost ($)				
Number of Tasks	PSO	SOS	G_SOS	EETSS-MCOA
100	195	162	136	119
200	203	180	144	125
300	203	175	151	131
400	217	191	158	138
500	233	206	174	147
600	231	213	187	156
700	249	225	203	166
800	256	229	200	169
900	259	231	203	164
1000	274	237	195	161

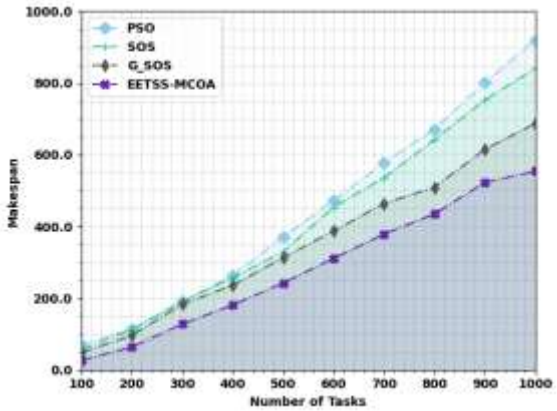

Figure 3: MKS analysis of EETSS-MCOA approach under several tasks

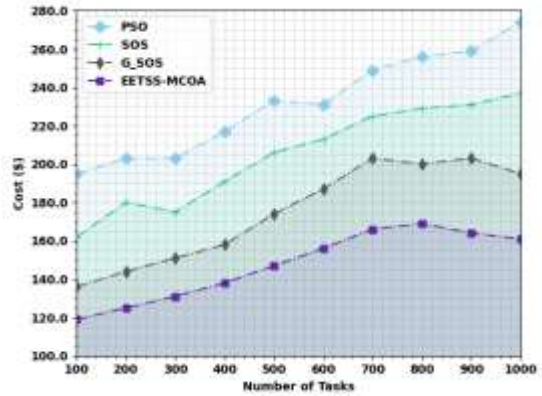

Figure 4: Cost analysis of EETSS-MCOA approach under several tasks

Table 2 and Fig. 4 demonstrate a detailed cost inspection of the EETSS-MCOA algorithm under several tasks. The experimental outcomes reported that the EETSS-MCOA approach has displayed improved outcomes with minimal values of cost. For example, with 100 tasks, the EETSS-MCOA algorithm has rendered minimal cost value of 119$ whereas the PSO, SOS, and G_SOS models have reached increased cost values of 195$, 162$, and 136$ correspondingly. Simultaneously, with 200 tasks, the EETSS-MCOA techniques have presented reduced cost value of 125$ whereas the PSO, SOS, and G_SOS models have attained increased cost values of 203$, 180$, and 144$ correspondingly. Further, with 300 tasks, the EETSS-MCOA approach has presented minimum cost value of 131$ whereas the PSO, SOS, and G_SOS approaches have obtained increased cost values of 203$, 175$, and 151$ correspondingly. Finally, with 400 tasks, the EETSS-MCOA model has granted reduced cost value of 138$ whereas the PSO, SOS, and G_SOS models have gained increased cost values of 217$, 191$, and 158$ correspondingly.

Table 3: Response time analysis of EETSS-MCOA approach with existing algorithms under several tasks

Response Time (sec)				
Number of Tasks	PSO	SOS	G_SOS	EETSS-MCOA
100	9	9	8	4
200	17	15	12	8
300	24	21	20	14
400	29	33	22	17
500	35	41	33	22
600	47	47	33	22
700	54	54	31	22
800	59	56	40	29
900	69	64	49	36
1000	73	69	48	38

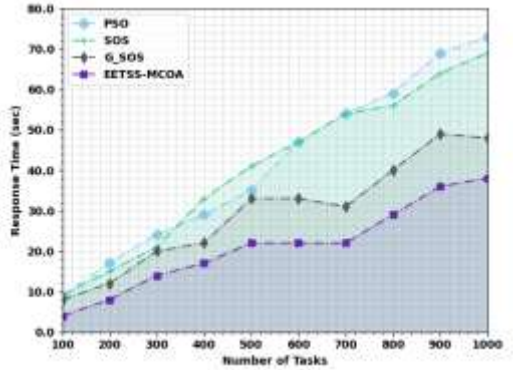

Figure 5: RET analysis of EETSS-MCOA approach under several tasks

Table 3 and Fig. 5 portray a response time (RET) inspection of the EETSS-MCOA model under several tasks. The experimental outcomes reported that the EETSS-MCOA approach has shown improved outcomes with minimal values of RET. For example, with 100 tasks, the EETSS-MCOA technique has granted reduced RET value of 4sec whereas the PSO, SOS, and G_SOS models have gained increased RET values of 9sec, 9sec, and 8sec correspondingly. Temporarily, with 200 tasks, the EETSS-MCOA methodology has presented minimum RET value of 8sec whereas the PSO, SOS, and G_SOS models have gained increased RET values of 17sec, 15sec, and 12sec correspondingly. Further, with 300 tasks, the EETSS-MCOA approach has granted reduced RET value of 14sec whereas the PSO, SOS, and G_SOS algorithms have reached increased RET values of 24sec, 21sec, and 20sec correspondingly. Eventually, with 400 tasks, the EETSS-MCOA method has presented minimal RET value of 17sec whereas the PSO, SOS, and G_SOS models have obtained increased RET values of 29sec, 33sec, and 22sec correspondingly.

Table 4: DOI analysis of EETSS-MCOA approach with existing algorithms under several tasks

Degree of Imbalance				
Number of Tasks	PSO	SOS	G_SOS	EETSS-MCOA
100	1.668	1.616	1.590	1.480
200	1.577	1.512	1.493	1.435
300	1.655	1.655	1.603	1.454
400	1.758	1.707	1.668	1.493
500	1.901	1.843	1.629	1.493
600	2.005	1.907	1.726	1.558
700	2.050	2.011	1.830	1.571
800	2.128	1.959	1.804	1.603
900	2.205	2.141	1.836	1.603
1000	2.309	2.173	1.946	1.655

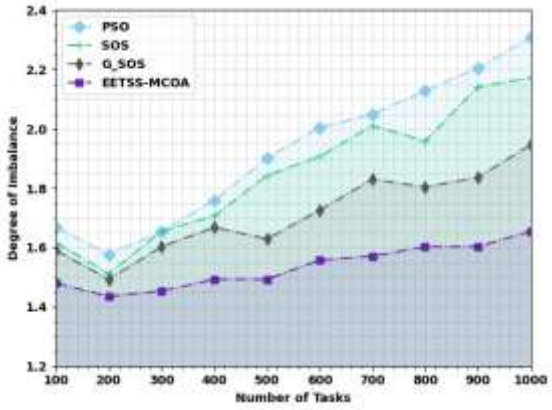

Figure 6: DOI analysis of EETSS-MCOA approach under several tasks

Table 4 and Fig. 6 illustrate a degree of imbalance (DOI) scrutiny of the EETSS-MCOA approach under several tasks. The experimental outcomes reported that the EETSS-MCOA algorithm has displayed improved outcomes with minimum values of DOI. For example, with 100 tasks, the EETSS-MCOA algorithm has presented minimal DOI value of 1.480 whereas the PSO, SOS, and G_SOS techniques have reached increased DOI values of 1.668, 1.616, and 1.590 correspondingly. Meanwhile, with 200 tasks, the EETSS-MCOA approach provided least DOI value of 1.435 whereas the PSO, SOS, and G_SOS approaches have obtained increased DOI values of 1.577, 1.512, and 1.493 correspondingly. In addition, with 300 tasks, the EETSS-MCOA techniques have presented minimum DOI value of 1.454 whereas the PSO, SOS, and G_SOS models have obtained increased DOI values of 1.655, 1.655, and 1.603 correspondingly. Eventually, with 400 tasks, the EETSS-MCOA algorithm has provided reduced DOI value of 1.493 whereas the PSO, SOS, and G_SOS methods have acquired increased DOI values of 1.758, 1.707, and 1.668 correspondingly.

Table 5: Waiting time analysis of EETSS-MCOA approach with existing algorithms under several tasks

Waiting Time (sec)				
Number of Tasks	PSO	SOS	G_SOS	EETSS-MCOA
100	11	9	6	5
200	20	17	10	7
300	28	23	18	11
400	33	29	20	13
500	42	34	26	18
600	52	44	31	23
700	55	50	37	30
800	62	54	46	37
900	71	61	53	45
1000	77	66	58	48

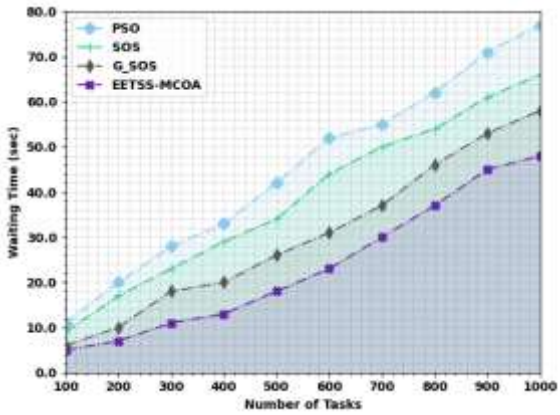

Figure 7: WAITT analysis of EETSS-MCOA approach under several tasks

Table 5 and Fig. 7 depict a waiting time (WAITT) inspection of the EETSS-MCOA model under several tasks. The experimental outcomes reported that the EETSS-MCOA algorithm has exhibited improved outcomes with minimal values of WAITT. For example, with 100 tasks, the EETSS-MCOA method has presented minimal WAITT value of 5s whereas the PSO, SOS, and G_SOS methods have obtained increased WAITT values of 11s, 9s, and 6s correspondingly. In the meantime, with 200 tasks, the EETSS-MCOA algorithm has presented reduced WAITT value of 7s whereas the PSO, SOS, and G_SOS techniques have obtained increased WAITT values of 20s, 17s, and 10s correspondingly. In addition, with 300 tasks, the EETSS-MCOA techniques have presented minimum WAITT value of 11s whereas the PSO, SOS, and G_SOS methodologies have reached increased WAITT values of 28s, 23s, and 18s correspondingly. Eventually, with 400 tasks, the EETSS-MCOA method offered reduced WAITT value of 13s whereas the PSO, SOS, and G_SOS approaches have obtained increased WAITT values of 33s, 29s, and 20s correspondingly.

Finally, Table 6 and Fig. 8 represent a throughput (THRO) examination of the EETSS-MCOA method with recent models. The experimental outcomes demonstrate that the EETSS-MCOA technique has shown enhanced results over other models. For instance, with 100 tasks, the EETSS-MCOA model has reached increased THRO of 95.82% whereas the PSO, SOS, and G_SOS models have resulted in decreased THRO of 79.05%, 81.67%, and 89.27% correspondingly. In the meantime, with 200 tasks, the EETSS-MCOA technique has reached increased THRO of 95.56% whereas the PSO, SOS, and G_SOS methodologies have resulted in decreased THRO of 79.32%, 82.20%, and 88.48% correspondingly. Additionally, with 300 tasks, the EETSS-MCOA method has reached increased THRO of 93.72% whereas the PSO, SOS, and G_SOS algorithms have resulted in decreased THRO of 77.48%, 81.15%, and 88.22% correspondingly. At last with 400 tasks, the EETSS-MCOA method has reached increased THRO of 92.94% whereas the PSO, SOS, and G_SOS approaches have resulted in decreased THRO of 72.50%, 75.77%, and 87.70% correspondingly.

Table 6: Throughput analysis of EETSS-MCOA approach with existing algorithms under several tasks

Throughput (%)				
Number of Tasks	PSO	SOS	G_SOS	EETSS-MCOA
100	79.05	81.67	89.27	95.82
200	79.32	82.20	88.48	95.56
300	77.48	81.15	88.22	93.72
400	72.50	75.77	87.70	92.94
500	64.65	70.31	85.34	91.89
600	56.00	63.69	74.86	83.24
700	41.86	50.59	53.64	71.19
800	29.02	32.78	34.26	58.36
900	16.71	18.30	20.83	37.66
1000	0.00	0.00	2.00	7.28

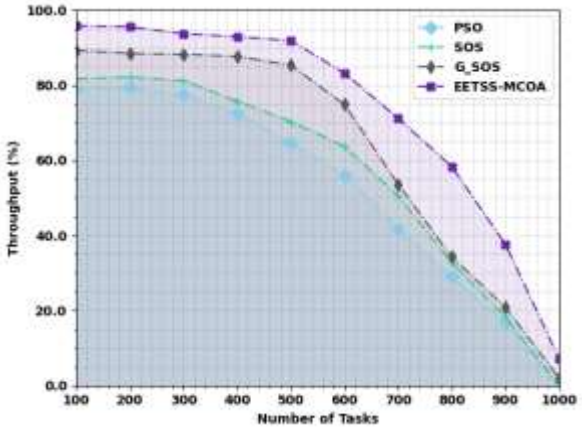

Figure 8: THRO analysis of EETSS-MCOA approach under several tasks

These results confirmed the effectual scheduling performance of the EETSS-MCOA model in the cloud platform.

4. Conclusion

In this study, a new EETSS-MCOA method has been established for scheduling tasks in the cloud environment. The EETSS-MCOA method carries out the derivation of features and MCOA is applied to schedule tasks. In addition, the MCOA algorithm is derived by the combination of adaptive β hill climbing concept with the COA for enhanced task scheduling. The conventional COA is stimulated by the swarming characteristics of birds known as coots. The COA followed two distinct stages of bird movements on water surface. The experimental results of the EETSS-MCOA model are validated on CloudSim tool. The solutions attained by the EETSS-MCOA model are found to be better than the existing algorithms. In future, the efficacy of the EETSS-MCOA technique is improvised using deep learning models.

References

[1] I.M. Ibrahim, "Task scheduling algorithms in cloud computing: A review," Turkish Journal of Computer and Mathematics Education (TURCOMAT), vol. 12, no. 4, pp. 1041-1053, 2021.

[2] J.C. Guevara and N.L. da Fonseca, "Task scheduling in cloud-fog computing systems," Peer-to-Peer Networking and Applications, vol. 14, no. 2, pp. 962-977, 2021.

[3] Y. Natarajan, S. Kannan and G. Dhiman, "Task scheduling in cloud using aco," Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science), vol. 15, no. 3, pp. 348-353, 2022.

[4] K. Pradeep, L.J. Ali, N. Gobalakrishnan, C.J. Raman and N. Manikandan, "CWOA: hybrid approach for task scheduling in cloud environment," The Computer Journal, vol. 65, no. 7, pp. 1860-1873, 2022.
[5] T. Bezdan, M. Zivkovic, N. Bacanin, I. Strumberger and E. Tuba, "Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm," Journal of Intelligent & Fuzzy Systems, vol. 42, no. 1, pp. 411-423, 2022.
[6] S. Potluri and K.S. Rao, "Optimization model for QoS based task scheduling in cloud computing environment," Indonesian Journal of Electrical Engineering and Computer Science, vol. 18, no. 2, pp. 1081-1088, 2020.
[7] M. Ibrahim, S. Nabi, A. Baz, N. Naveed and H. Alhakami, "Towards a task and resource aware task scheduling in cloud computing: An experimental comparative evaluation," International Journal of Networked and Distributed Computing, vol. 8, no. 3, pp. 131-138, 2020.
[8] G. Natesan and A. Chokkalingam, "An improved grey wolf optimization algorithm based task scheduling in cloud computing environment," International Arab Journal of Information Technology, vol. 17, no. 1, pp.73-81, 2020.
[9] P. Albert and M. Nanjappan, "WHOA: hybrid based task scheduling in cloud computing environment," Wireless Personal Communications, vol. 121, no. 3, pp. 2327-2345, 2021.
[10] S.M.G. Kashikolaei, A.A.R. Hosseinabadi, B. Saemi, M.B. Shareh A.K. Sangaiah et al., "An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm," The Journal of Supercomputing, vol. 76, no. 8, pp. 6302-6329, 2020.
[11] R. Medara and R.S. Singh, "Energy efficient and reliability aware workflow task scheduling in cloud environment," Wireless Personal Communications, vol. 119, no. 2, pp. 1301-1320, 2021.
[12] X. Huang, C. Li, H. Chen and D. An, "Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies," Cluster Computing, vol. 23, no. 2, pp. 1137-1147, 2020.
[13] M. Abd Elaziz and I. Attiya, "An improved Henry gas solubility optimization algorithm for task scheduling in cloud computing," Artificial Intelligence Review, vol. 54, no. 5, pp. 3599-3637, 2021.
[14] S. Velliangiri, P. Karthikeyan, V.A. Xavier and D. Baswaraj, "Hybrid electro search with genetic algorithm for task scheduling in cloud computing," Ain Shams Engineering Journal, vol. 12, no. 1, pp. 631-639, 2021.
[15] X. Chen, L. Cheng, C. Liu, Q. Liu, J. Liu et al., "A WOA-based optimization approach for task scheduling in cloud computing systems," IEEE Systems Journal, vol. 14, no. 3, pp. 3117-3128, 2020.
[16] A. Najafizadeh, A. Salajegheh, A.M. Rahmani and A. Sahafi, "Multi-objective Task Scheduling in cloud-fog computing using goal programming approach," Cluster Computing, vol. 25, no. 1, pp. 141-165, 2022.
[17] A.D. Boursianis, M.S. Papadopoulou, M. Salucci, A. Polo, P. Sarigiannidis et al., "Frequency Selective Surface Design Using Coot Optimization Algorithm for 5G Applications," 2022 International Workshop on Antenna Technology (iWAT), 2022, pp. 184-187, doi: 10.1109/iWAT54881.2022.9810997.
[18] K. Sun, H. Jia, Y. Li and Z. Jiang, "Hybrid improved slime mould algorithm with adaptive β hill climbing for numerical optimization," Journal of Intelligent & Fuzzy Systems, vol. 40, no. 1, pp. 1667-1679, 2021.
[19] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A. De Rose and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and experience, vol. 41, no. 1, pp. 23-50, 2011.
[20] A.A. Zubair, S.A. Razak, M.A. Ngadi, A. Al-Dhaqm, W.M. Yafooz et al., "A Cloud Computing-Based Modified Symbiotic Organisms Search Algorithm (AI) for Optimal Task Scheduling," Sensors, vol. 22, no. 4, pp. 1674, 2022.
[21] Ahmed M. AbdelMouty. (2022). An advanced optimization technique for integrating IoT and cloud computing on manufacturing performance for supply chain management. Journal of Journal of Intelligent Systems and Internet of Things, 7 (2), 30-39 (Doi : https://doi.org/10.54216/JISIoT.070203)
[22] S. Phani Praveen, Balamuralikrishna Thati, Ch Anuradha, S. Sindhura, Mohammed Altaee, M. Abdul jalil. (2023). A Novel Approach for Enhance Fusion Based Healthcare System In Cloud Computing. Journal of Journal of Intelligent Systems and Internet of Things, 9 (1), 84-96 (Doi : https://doi.org/10.54216/JISIoT.090106)