Yanbu Journal of Engineering and Science

Volume 19 | Issue 1 Article 3

2022

Capuchin search algorithm based task scheduling in cloud computing environment

Sivakumar Ramu
Yanbu University College

Rameshkumar Ranganathan
Yanbu University College

Ramakrishnan Ramamoorthy
Yanbu University College

Follow this and additional works at: https://yjes.researchcommons.org/yjes

The journal in which this article appears is hosted on Digital Commons, an Elsevier platform.

Recommended Citation

Ramu, Sivakumar; Ranganathan, Rameshkumar; and Ramamoorthy, Ramakrishnan (2022), Capuchin search algorithm based task scheduling in cloud computing environment, *Yanbu Journal of Engineering and Science*: Vol. 19: Iss. 1, 18-29.
DOI: https://doi.org/10.53370/001c.33767
Available at: https://yjes.researchcommons.org/yjes/vol19/iss1/3

This Research Article is brought to you for free and open access by Yanbu Journal of Engineering and Science. It has been accepted for inclusion in Yanbu Journal of Engineering and Science by an authorized editor of Yanbu Journal of Engineering and Science.

Ramu S, Ranganathan R, Ramamoorthy R. Capuchin search algorithm based task scheduling in cloud computing environment. *YJES*. 2022;19(1):18-29.
doi:10.53370/001c.33767

Research article

Capuchin search algorithm based task scheduling in cloud computing environment

Sivakumar Ramu[1], Rameshkumar Ranganathan[1], Ramakrishnan Ramamoorthy[1]

[1] Computer Science & Engineering, Yanbu University College

Keywords: task scheduling, task clustering, cloud computing, capuchin search, makespan

https://doi.org/10.53370/001c.33767

Yanbu Journal of Engineering and Science

Vol. 19, Issue 1, 2022

Cloud computing is mathematical process that provides more power and flexibility in computing infrastructure. Cloud computing provides internet services using a network of remote services. The core service for any environment is the best business plan that supports better quality of service (QoS). Task scheduling in the cloud is a key issue that needs to be addressed to improve system performance and high customer satisfaction. The task scheduling affects the exact time of operation and the cost of using the system. In this paper, we propose a capuchin search algorithm based task scheduling (CSTS) in cloud computing environment. In CSTS method, first we introduce an improved cuttlefish optimization (ICFO) algorithm for task clustering which groups user task into two set as normal and emergency task. Then, we develop a modified capuchin search (MCS) algorithm for priority based optimal task scheduling which minimize makespan and improve resource utilization. Finally, the simulation results of proposed CSTS method is compared with the existing state-of-art methods in terms of makespan, execution time, deadline violation rate and resource utilization.

1. INTRODUCTION

Due to the rapid development of cloud and cloud applications, many applications go to the cloud.[1] Due to the growing demand and demand for cloud computing, many existing projects require strong computational development. An important part of the cloud is the integration of multiple computing elements, which means that users can have unlimited access and make the user.[2,3] Cloud computing is a new technology platform that enables technology devices to provide graduates with complex environments with reliable and reliable service. The services offered include the ability to ask questions and other online services, which is one of four key technologies that could be improved. Depending on the needs of the program, special business planning training is offered. DAG[4,5] method is used for DNA analysis, labeling, mapping, weather information and other useful applications. Algorithms are used to solve server problems and configure multiple tasks. Four business plans have been developed to meet different needs. Participants' interests in various mental health services are an important factor to consider when organizing a competition because participants are not able to make sacrifices when assigned to perform their duties. Participation in future mental health programs.[6] Because so many software programs are shared with users, a good business plan is essential to use the best tools and practices. Many system parameters, such as output capacity, memory, and bandwidth, affect performance.[7] In other words, different types of software and software in different regions exacerbate the problem of inefficiency. In addition, data is used between the ports, servers, and data centers until it can be terminated. Modern research on different types of tools and applications[8] focuses on performance level optimization across multiple sectors, small tasks for optimal solutions, and machine learning. However, due to the high strength of the back and legs, a similar procedure is not possible.[9] In addition, access to the cloud may be restricted due to access restrictions on the virtual machine (VM). Several software schemes have been developed recently to improve cloud performance.

Energy-aware dynamic task scheduling (EDTS) algorithms[10] are used to reduce the power consumption of mobile phones as well as allow them to maximize their potential. The CWSA[11] is used for structured conference activities, as well as four shift periods, delays, costs and delays. One working group the temporal task scheduling algorithm (TTSA) algorithm[12] has been developed by the CDC and globalization. Algorithms are used in many places, and basic methods are used to reduce inefficiency.[13] The best price offers the best price by using the best methods and conversions, real estate. Multi-queue interlacing peak scheduling method (MIPSM)[14] based on data from different service types and devices. This is called the opportunistic task scheduling over co-located clouds (OSCC),[15] which allows you to use this standard to change latency. Two-way methods are used to improve the configuration and performance of the cloud.[16] The classification system[17] is based on a modified genetic algorithm (MGA) system that uses more than two binary, family-recognized and developed systems. Avoid impossible solutions. Johnson's law combined the Johnson's-rule-based genetic algorithm (JRGA)[18] to create an art system that recognizes the function of mul-

Table 1. Summary of Research Gap.

Refs.	Algorithm	Task nature	Findings	Challenges
20	TBTS, SLA-LB	Independent	Algorithm provide better cloud utilization	Energy efficiency of the algorithm is very low
21	QEEC	Independent	Scheduler on each server prioritizes and requests by task laxity and life time.	Higher possibility of overloaded
22	Fuzzy TOPSIS-PSO	Independent	Thrives to accomplish the total task with eminent resource utilization	Task rescheduling causes slower the process.
23	ANN	Independent	To reduce make span, energy consumption, execution overhead and number of active racks.	Execution cost is not considered.
24	IACO	Independent	Shortest waiting time, degree of resource load balance and cost of task	Complexity and high monitoring overhead
25	QoS-DPSO	Independent	Time, reliability and cost	Does not reduce the make-span time
26	MVO-GA	Independent	Enhance tasks transfer via the cloud	Energy efficiency of the algorithm is very low
27	AC-PCO	Independent	Enhance make span, cost, and resource utilization rate	Higher possibility of overloaded.
28	PBACO	Independent	Enhance cost, the violation rate of deadline and resource utilization	Task rescheduling causes slower the process.
29	DQTS	Independent	Enhance make span and load balance	Energy efficiency of the algorithm is very low

tiple genes. GACCRATS is a cloud based customer service as well as transmission algorithm.[19]

Our contributions. To expand, Capuchin Search Algorithm (CSTS) has been developed for cloud environments. The main purpose of this research project is to solve the problem of cloud performance planning, which improves performance based on important factors such as price, power and MacPhone. The other chapters of the story are as follows: Section. 2 describe the latest functionality and configuration of cloud computing services. Section 3 discusses the problem-solving and configuration of the required CSTS system. The role of the CSTS system is described in the section 4 has a good example of mathematics. In Sect. 5, we describe the experimental results and the comparative analysis of the established and existing methods. Eventually, the work left concludes in Sect. 6.

2. RELATED WORKS

Over the past few years, many cloud studies have been conducted on task scheduling from around the world. These texts are grouped into categories and listed in Table 1.

Two cloud scheduling have been developed: TBTS and SLA-LB.[20] TBTS is a two-step algorithm that prevents operation within a package. The supports optimal solution systems and technologies based on TBTS algorithm based on ETC matrix. VM performance is lower than expected rewards. The SLA-LB algorithm is an online tool for tailoring performance to customer needs in two ways, such as time and budget. Q-learning based task scheduling framework[21] is guidelines for energy-efficient cloud computing (QEEC). The hacker also uses the M/M/S type, so users can request applications on a daily basis. Q-Learning applications include professional work, the installation of all applications based on professional life and work, then the implementation of VM applications, reduced response time, and the use of energy for personal development. Fuzzy-TOPSIS and particle swarm optimization (PSO)[22] is proposed for task scheduling with recommended VM numbers. Problems in SLA planning are solved by Fuzzy TOPSIS, which requires better power, cost and downtime. The Fuzzy TOPSIS approach benefits many applications and has a significant impact on global decision-making. For personal use and energy consumption using optical links,[23] Makespan is used to reduce energy consumption. The functionality of the neural computer network takes into account the current working conditions in the current cloud and uses the best mathematical model for all tasks, a combination of our goals. They used this data to train network networks using algorithms and backhaul algorithms with 99.9% accuracy to generate big data. Satisfaction of the task scheduling is designed to test the best work solution[24] with three objectives: minimum time. , Infrastructure and billing. They use them to improve two conditions, pheromone synthesis and pheromone vulcanization, to increase the number of colony algorithms. VM causes overweight during local pheromone connections to achieve weight balance.

QoS performance assessment is based on the configuration of the infrastructure to achieve the required QoS performance (QoS-PSO).[25] The analysis also shows that for low-performance applications, the shutdown time of the QoS-DPSO algorithm and DBC algorithm is usually shorter than that of other algorithms with less time remaining. An effective method for i.e. hybrid multi-verse optimizer with a genetic algorithm (MVO-GA) used to improving operational monitoring.[26] MVO-GA is used to improve the performance of network applications due to the high performance of cloud applications. The conversion result must be obtained when returning the survey work based on the work done that day. MVO-GA operates on several cloud assets:

speed, capacity, number of jobs, number of jobs, number of VMs, and performance.

The hybrid AC-PSO algorithm[27] is a service planning system that provides services in cloud computing systems. There are many applications related to these services, and these services are categorized according to the importance and size of the service used. Research studies show that AC-PSO algorithms reduce production time and increase material efficiency and efficiency compared to conventional methods. Depending on the type of equipment purchased, this may be the way to meet many expectations using multi-objective optimization scheduling method.[28] This system uses Makespan and user budgeting as the most efficient solution for the most efficient and cost-effective performance. To resolve this issue, the optimal solution has been updated using improved ant colony algorithm. There are two management functions that are used to analyze and provide information on budgeting and billing performance. A deep Q-learning task scheduling (DQTS)[29] combine the benefits of this type of Q learning with deep neural connections. The purpose of DQTS is to solve the problem of managing acyclic simulations that can be performed across the cloud. The main idea of our program is to use the popular Q-in-dept curriculum to teach the basics when planning a project. The experimental method has little to no problem solving the online learning problem developed by DQTS algorithm.

3. PROBLEM METHODOLOGY AND SYSTEM MODEL

3.1 PROBLEM STATEMENT

The cloud computing multi-objective task scheduling optimization (MOTSO)[30] was developed using its own storage system. Select additional criteria such as time constraints, equipment load balance, and the purpose of the daily inspection program. The purpose of the cloud is a multi-functional work plan. As the results show, this system works best in terms of time, space, and hardware usage. The algorithm sets three goals: short-term customer expectation, high product balance, high service cost, one service that can be tested, and one customized. As a result, it provides more effective planning and shorter turnaround time. Cloud computing is designed to meet the mathematical needs of current and future scientific applications. This consumes a lot of resources, so computing is an important part of the process. Using this method to reduce work time and earn more money for the project is necessary to organize the work efficiently. But leadership is not a good decision. Therefore, a well-designed system is needed to achieve the desired goals in a cloud environment. The best solution for working in a cloud VM is to find the best solution for the NP problem that can be solved in polynomial time. To solve above problems, a capuchin search algorithm based task scheduling (CSTS) method is proposed for cloud computing environment. The main contributions of proposed CSTS method for cloud computing environment are summarized as follows:

1. An improved cuttlefish optimization (ICFO) algorithm is first proposed for task clustering which groups' user task into two set as normal and emergency task.
2. A modified capuchin search (MCS) algorithm is used for priority based optimal task scheduling which minimize makespan and improve resource utilization.

3.2 SYSTEM MODEL OF PROPOSED TASK SCHEDULING

As shown in Figure 1, we show the expected CSTS format in the cloud. The pilot project has two parts - closure and action planning. All cloud members assign their tasks in the form of T-Set functions using rows or diagrams. When receiving a service from a cloud service provider, the employee checks the employee's integrity. If the staff receives the request, the request will be accepted, otherwise, the request will be rejected. The servant gathered all the work he had received from his master and handed it over to the cloud server as a distribution of resource provisioning and task scheduling.

4. PROPOSED METHODOLOGY

In this section, the working process of task clustering and task scheduling has been explained with the ICFO and MCS algorithm respectively.

4.1 TASK CLUSTERING

The cuttlefish optimization (CFO) algorithm describes the process of changing the color of a needle to solve a global problem.[31] The color and type of cuttlefish are reflected in three different parts of the cell. Evolution is the science of evolution. CFO is a fun algorithm that imitates the behavior of carved fish by changing its color depending on the environment. In this project, we develop an improved cuttlefish optimization (ICFO) algorithm based on the CFO algorithm. In the past, these concepts were created using two code of conduct.

Initially, two (coded R_1 and R_2) parameters are used to create a reflection strategy. Meanwhile, the other two (coded U_1 and U_2) are used for the visibility strategy. At the initial stage, it is randomly generated with an initial population (called Q).

$$Q[y_j].points[i] = random * (Upperlimit - Lowerlimit) + Lowerlimit \quad (1)$$

where are the lower and upper limits of the problem area. Specifies a random number between [0, 1]. Each decision has two values: the first value is a magnitude vector of successive values composed of decision points (called $Points$), the second value represents the relevance of the decision, where y_j it represents an individual decision, and point i is the individual decision in the first sense.

$$H_{size} = \frac{Q}{4} \quad (2)$$

H_{size} given the size of each group, Q is the first digit. During the solution development process, y_{new} new solutions are developed using (3) until the completion process is met:

$$y_{new} = reflection + visibility \quad (3)$$

where y_{new} is a new solution, it is also a collection of compiled values are reflector and visibility. In many cases, r and

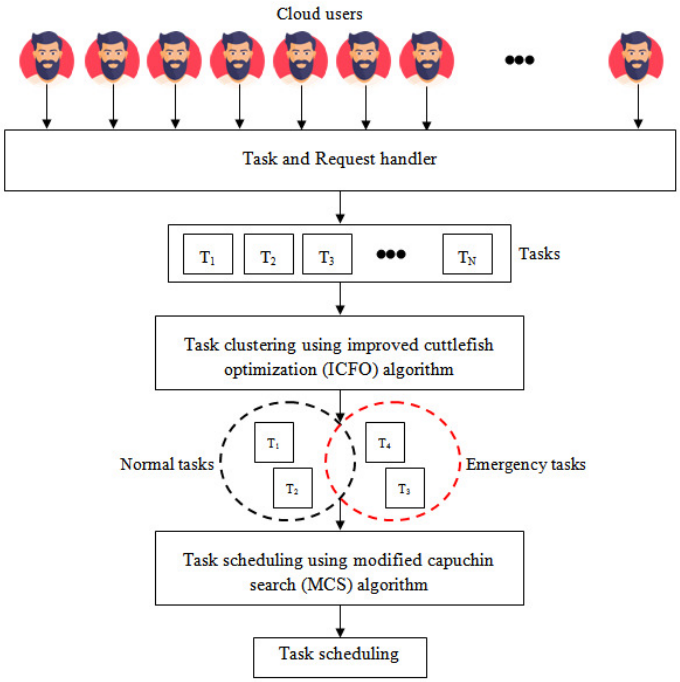

Figure 1. Proposed system model for CSTS method.

u values are calculated using (4), (5), r and u to calculate perception and visibility. However, in cases 3, 4 and 5, r is set to 1 and in cases 1 and 2, u is adjusted to 1.

$$r = random() * (R_1 - R_2) + R_2 \quad (4)$$

$$u = random() * (U_1 - U_2) + U_2 \quad (5)$$

where $random()$ is the random number from 0 to 1. These processes R_1, R_2, U_1, and U_2 can be manually configured at the initial stage. The average value of the best set of values (Bu_{best}) is calculated using (6). It was used in 5 cases (one-third), and the sum of the best solution points divided by N is found and calculated, where N is the absolute number of points.

$$Bu_{best} = \frac{\sum[bestsolution.points[i]]}{N} \quad (6)$$

At the end of the initial phase, four search engines are grouped into: (7) and (8) new solutions to calculate reflection and visibility, respectively. H_1 works as a public survey (observation) and is calculated using the value of r (4):

$$reflection = r * H_1[y_i].points[i] \quad (7)$$

$$visibility = u * (bestsolution.points[i] - H_1[y_j].points[i]) \quad (8)$$

whwere H_1 is the first solution. y_j is a solution for one and draw the i-th point of the solution. $[y_j].points[i]$ is the best solution. Develop a new solution $bestsolution.points[i]$ around the best solution y_{new} by using (8) and (9) to calculate meditation and visibility, respectively. H_2 acting as a local survey, your profits are calculated using (5):

$$reflection = r * bestsolution.points[i] \quad (9)$$

Based on this, y_{new} modified solution was developed. Eqns. (9) and (10) used for calculate reflection and visibility respectively. It works as a local search H_3 (exploitation) and is calculated using the value of u (5):

$$visibility = u * (bestsolution.points[i] - Bu_{best}) \quad (10)$$

For example, if the best solution has two heads (6, -2), U_1, and U_2 is fixed (1, -1). Test results Bu_{best} are calculated as follows: Bu_{best}=(6-2)/2=2

$$H_{5-\text{size}} = \left[((H5T_{percent} * H_{size}) * 4) + \frac{H_{size}}{2}\right] \quad (11)$$

Solutions from the four groups are initially stored to produce only the best solutions as H_{5-size}

$$\begin{aligned} H_5 = {} & (\uparrow H_1 * H5T_{percent}) \\ & + (\uparrow H_2 * H5T_{percent}) \\ & + (\uparrow H_3 * H5T_{percent}) \\ & + (\uparrow H_4 * H5T_{percent}) \end{aligned} \quad (12)$$

where ↑ represents the optimal best solution which is designed from good to worst solution. Algorithm 1 describes the working function of user task clustering using improved cuttlefish optimization (ICFO) algorithm.

4.2 TASK SCHEDULING

Capuchin Search (CS) algorithm[32] is a population-based forum promoted by social media. Depending on the shape of the capuchins, there is a structural design. In this project, we transformed the CS program from curriculum using modified capuchin search (MCS). MCS algorithms are used for priority based optimal task scheduling. First, we initialize cloud node (user) task (CN) with each of sizes i of CN_j was activated as follows:

$$\begin{aligned} CN_{ji} = {} & CN_{Minj} \\ & + Rand[0,1](CN_{Maxj} - CN_{Minj}) \end{aligned} \quad (13)$$

where CN_{Maxj} and CN_{Minj} are limits of CN_j in ith position [0, 1] and direction and random is a haphazard digit that is evenly distributed over the range [0, 1].

$$\begin{aligned} CN_{Newji} = {} & CN_{ji} + Rand[0,1](WW_{zi} - CN_{ji}) \\ & + Rand[-1,1](CN_{si} - CN_{ji}) \end{aligned} \quad (14)$$

Algorithm 1. Task clustering using ICFO algorithm.

Input	: q points and y_j
Output	: H_{5-size}, Bu_{best}
1	Initialize the population Q
2	Initialize parameters R_1, R_2 *and* U_1, U_2
3	Calculate fitness $H_{size} = \frac{Q}{4}$
4	Calculate best solution in $Bu_{best} = \frac{\sum[bestsolution.points[i]]}{N}$
5	If i=0, j=1
6	Determine (y) values $y_{new} = reflection + visibility$
7	Calculate r and u $r = random() * (R_1 - R_2) + R_2$ $u = random() * (U_1 - U_2) + U_2$
8	Update position using activation functions $visibility = u * (bestsolution.points[i] - Bu_{best})$
9	End while
10	End

where the jth measurement of the ith CN is denote by CN_{ji}, the ith scale of the local group manager location indicates zth WW_{zi}. The level update equation for the global leader level is given

$$CN_{Newji} = CN_{ji} + Rand[0,1](FW_{zi} - CN_{ji}) + Rand[-1,1](CN_{si} - CN_{ji}) \quad (15)$$

where FW_{zi} is the ith the measurement of the global leader situation and i [{1, 2, ..., D} is an randomly particular list. The probability $PROB_j$ is calculated as follows

$$PROB_j = 0.9 \times \frac{Fitness_j}{Max\ fitness} + 0.1 \quad (16)$$

This is done using the level upgrade equation at the local leader stage

$$CN_{Newji} = CN_{ji} + Rand[0,1](WW_{zi} - CN_{ji}) + Rand[-1,1](CN_{si} - CN_{ji}) + g(CN_{si} - CN_{ji}) \quad (17)$$

where g is determined by GSS process. This is done using the level update equation, in the global leader phase

$$CN_{Newji} = CN_{ji} + Rand[0,1](FW_{zi} - CN_{ji}) + Rand[-1,1](CN_{si} - CN_{ji}) + g(CN_{si} - CN_{ji}) \quad (18)$$

G is defined by the GSS process. The modified equations are:

$$CN_{Newji} = CN_{ji} + PROB_j(FW_i - CN_{ji}) + (1 - PROB_j)(CN_{ji} - WW_{zi}) \quad (19)$$

where $PROB_j$ the likelihood which depends on fitness. Capuchin fitness is used to calculate size according to the status update process.

$$q_j = \frac{1}{2} \times \left\{ \left[\left(a_i^2 - h_i^2\right) g(B) + \left(h_i^2 - b_i^2\right) g(A) + \left(b_i^2 - a_i^2\right) g(H) \right] \div \left[(a_i - h_i) g(B) + (h_i - b_i) g(A) + (b_i - a_i) g(H) \right] \right\} \quad (20)$$

The formula for the current situation can be written as follows

$$H_{(T)} = g(H_{(T-1)}, Y_T) \quad (21)$$

Here, H_T is the new state, $H_{(T-1)}$ is the previous condition while Y_T is the current input. The activation function is $\tan H$, load on a continuous neuron is L_{HH} and the load at the input neuron is L_{YH}, Let's write the equation for the state when necessary T as

$$H_T = \tan H(L_{HH} H_{T-1} + L_{YH} Y_T) \quad (22)$$

The output level can be calculated by calculating the current level

$$X_T = L_{HX} H_T \quad (23)$$

Let's calculate the current situation H_T.

$$H_T = \tan H(L_{HH} H_{T-1} + L_{YH} Y_T) \quad (24)$$

In each state a continuous neural network is produced. Let's calculate X_T.

$$X_T = L_{HX} H_T \quad (25)$$

The working function of modified capuchin search (MCS) based task scheduling is described in Algorithm 2.

5. RESULTS AND DISCUSSION

The performance of capuchin search algorithm-based task scheduling (CSTS) is evaluated and validated using different scales, workloads, and standard values. CSTS system powered by Intel (R) Xeon (R) CPU developed data center to test the performance of open cloud environments with 16 RAM and 2.5 GHz on the HP Z620 Workstation Environment platform. Use Cloudsim-3.0. Each unit has one hundred and 10 machine guns. The simulation results of proposed CSTS method is compared with the existing state-of-art PBACO,[28] DQTS[29] and MOTSO[30] task scheduling models.

5.1 PERFORMANCE METRICS

Examples of metric performance include makespan, execution time, deadline violation rate and resource utilization that compares and compares all features; Due to different plans and resources, process time will vary; Delay in QoS is because it is a well-designed system and requires information to test its effectiveness; The fourth indication is the use of resources. The model shows the speed it takes and the quality of success planned. If the working time is longer than the scheduled time, the working time is considered to be over.

5.2 COMPARATIVE ANALYSIS OF PROPOSED AND EXISTING TASK SCHEDULING METHODS

5.2.1 VARYING NUMBER OF TASKS

In this test, we consider to vary the number of cloud computing multi-objective tasks with the fixed standard format. Figure 2 shows the Makespan comparative analysis of proposed and existing task scheduling methods with varying number of tasks. The plot clearly reflects that the Makespan of proposed CSTS method is 49.636%, 35.909% and 19.355% much thrived than the existing PBACO,[28] DQTS[29] and MOTSO[30] task scheduling models. Figure 3 shows the execution time comparative analysis of proposed and existing

Algorithm 2. Task scheduling using MCS algorithm.

Input	: Position of Capuchin
Output	: Entropy error
1	Initialise the parameters
2	Update the global leader position using $$CN_{Newji} = CN_{ji} + Rand[0,1](FW_{zi} - CN_{ji}) + Rand[-1,1](CN_{si} - CN_{ji})$$
3	Determine the probability using $$PROB_j = 0.9 \times \frac{Fitness_j}{Max\ fitness} + 0.1$$
4	Compute the local leader position using $$CN_{Newji} = CN_{ji} + Rand[0,1](FW_{zi} - CN_{ji}) + Rand[-1,1](CN_{si} - CN_{ji}) + g(CN_{si} - CN_{ji})$$
5	Determine the objective function using $$q_j = \frac{1}{2}\frac{(a_i^2 - h_i^2)g(B) + (h_i^2 - b_i^2)g(A) + (b_i^2 - a_i^2)g(H)}{(a_i - h_i)g(B) + (h_i - b_i)g(A) + (b_i - a_i)g(H)}$$
6	Define the current level with the output level
7	Calculate X_T using $$X_T = L_{HX}H_T$$
8	Compute the entropy value
9	End procedure

task scheduling methods with varying number of tasks. The plot clearly reflects that the execution time of proposed CSTS method is 49.757%, 38.516% and 27.230% lower than the existing PBACO,[28] DQTS[29] and MOTSO[30] task scheduling models. Figure 4 shows the deadline violation rate comparative analysis of proposed and existing task scheduling methods with varying number of tasks. The plot clearly reflects that the deadline violation rate of proposed CSTS method is 81.304%, 70.345% and 50.857% lower than the existing PBACO,[28] DQTS[29] and MOTSO[30] task scheduling models. Figure 5 shows the resource utilization comparative analysis of proposed and existing task scheduling methods with varying number of tasks. The plot clearly reflects that the resource utilization of proposed CSTS method is 16.920%, 11.063% and 4.989% efficient than the existing PBACO,[28] DQTS[29] and MOTSO[30] task scheduling models.

5.2.2 *VARYING TASK STANDARD (TASK STD.)*

In this test, we consider to vary the task standard format with the fixed cloud computing multi-objective tasks. Figure 6 shows the Makespan comparative analysis of proposed and existing task scheduling methods with varying task standard format. The plot clearly reflects that the Makespan of proposed CSTS method is 45.998%, 32.050% and 15.996% much thrived than the existing PBACO,[28] DQTS[29] and MOTSO[30] task scheduling models. Figure 7 shows the execution time comparative analysis of proposed and existing task scheduling methods with varying task standard format. The plot clearly reflects that the execution time of proposed CSTS method is 51.402%, 40.144% and 26.285% lower than the existing PBACO,[28] DQTS[29] and MOTSO[30] task scheduling models. Figure 8 shows the deadline violation rate comparative analysis of proposed and existing task scheduling methods with varying task standard format. The plot clearly reflects that the deadline violation rate of proposed CSTS method is 81.250%, 71.622% and 52.273% lower than the existing PBACO,[28] DQTS[29] and MOTSO[30] task scheduling models. Figure 9 shows the resource utilization comparative analysis of proposed and existing task scheduling methods with varying task standard format. The plot clearly reflects that the resource utilization of proposed CSTS method is 13.775%, 7.886% and 3.049% efficient than the existing PBACO,[28] DQTS[29] and MOTSO[30] task scheduling models.

6. CONCLUSION

We have proposed a new version of task scheduling method based on capuchin search algorithm (CSTS) in cloud computing environment. The main contributions of proposed CSTS method are summarized as follows:

- An improved cuttlefish optimization (ICFO) algorithm is first proposed for task clustering which groups user task into two set as normal and emergency task.
- A modified capuchin search (MCS) algorithm is then introduced for priority based optimal task scheduling which minimize makespan and improve resource utilization.

From the simulation results, we showed that the effectiveness of proposed CSTS method over the existing state-of-art PBACO,[28] DQTS[29] and MOTSO[30] task scheduling models in terms of makespan, execution time, deadline violation rate and resource utilization. The average makespan of proposed CSTS method is 34.966% and 31.3478% much thrived than the existing task scheduling models for number of tasks and task standard respectively. The average execution time of proposed CSTS method is 38.501% and 39.27% much thrived than the existing task scheduling models for number of tasks and task standard respectively. The average deadline violation rate of proposed CSTS method is 67.502% and 68.38% much thrived than the existing task scheduling models for number of tasks and task

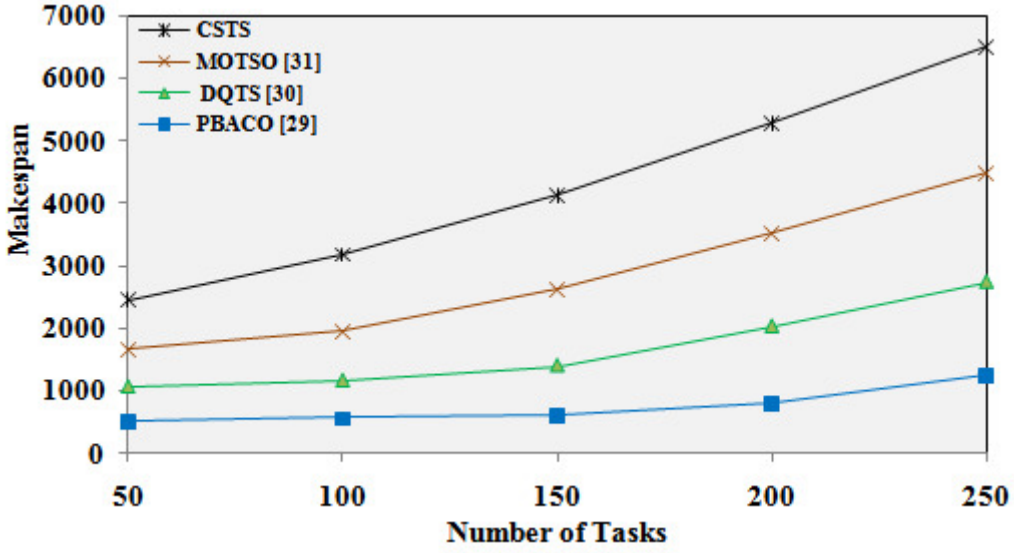

Figure 2. Makespan with number of task.

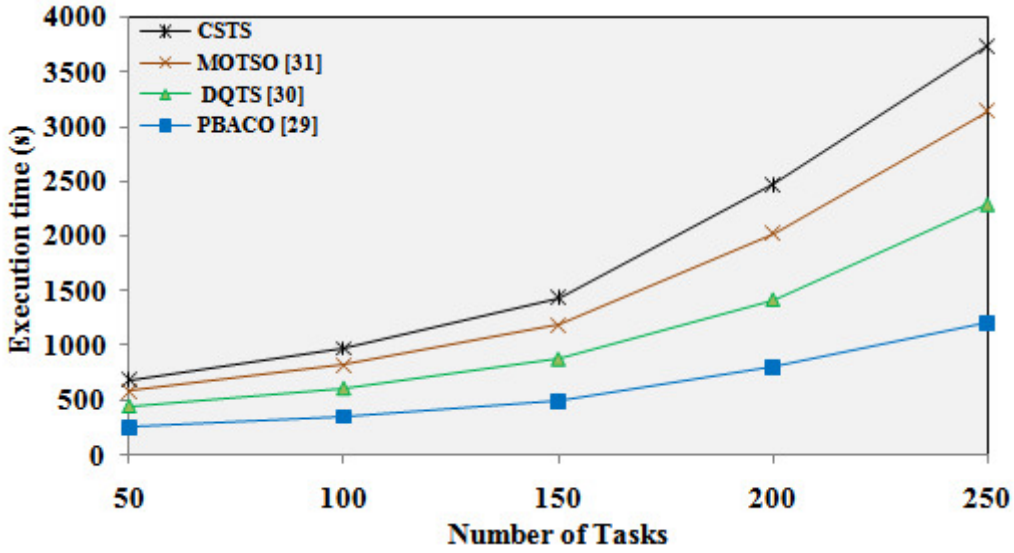

Figure 3. Execution time with number of task.

standard respectively. The average resource utilization of proposed CSTS method is 10.99% and 8.236% much thrived than the existing task scheduling models for number of tasks and task standard respectively.

Submitted: February 16, 2022 +03, Accepted: March 16, 2022 +03

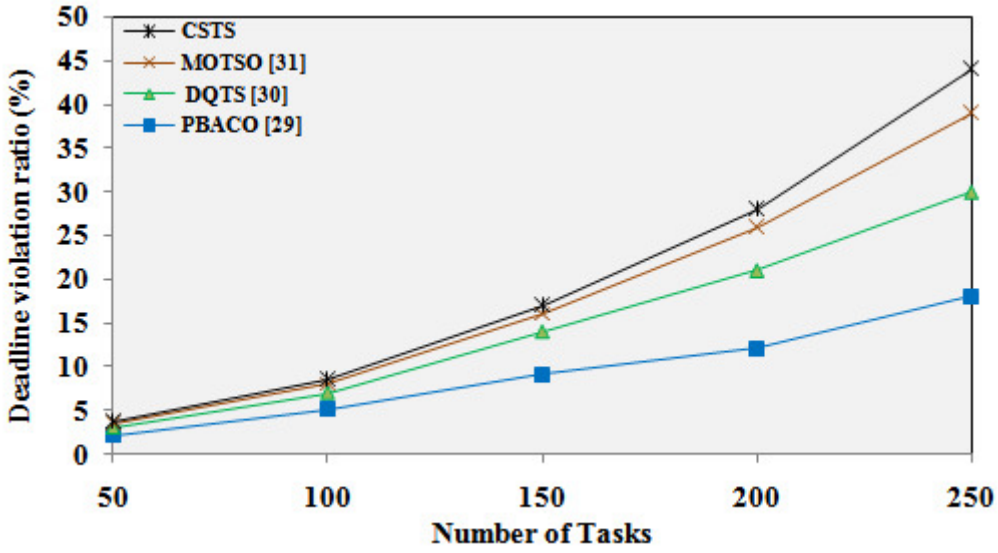

Figure 4. Deadline violation rate with number of task.

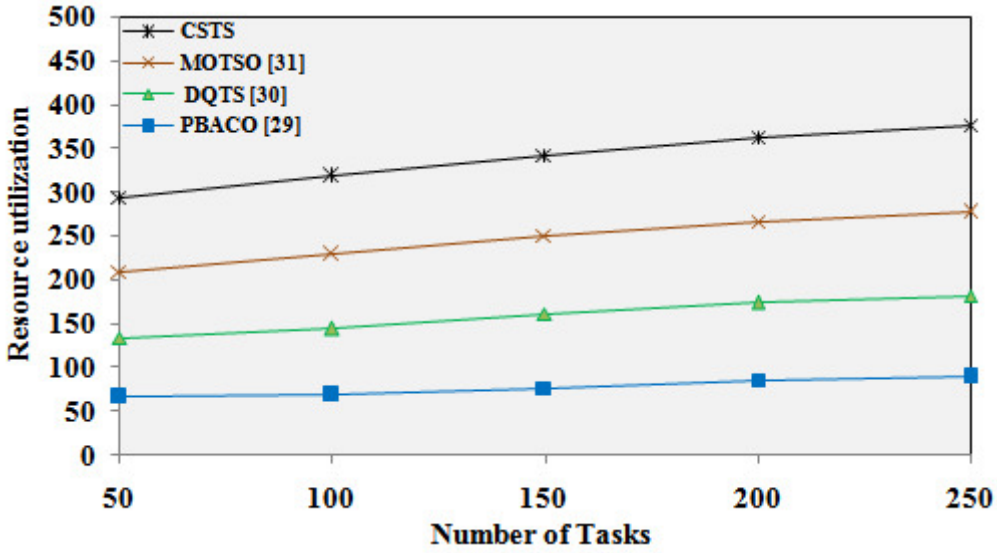

Figure 5. Resource utilization with number of task.

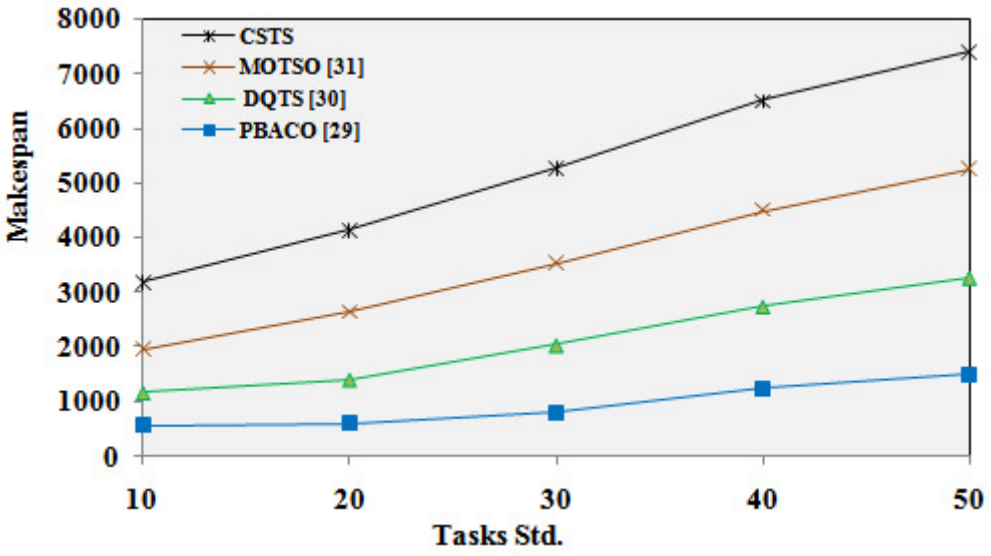

Figure 6. Makespan with task standard format.

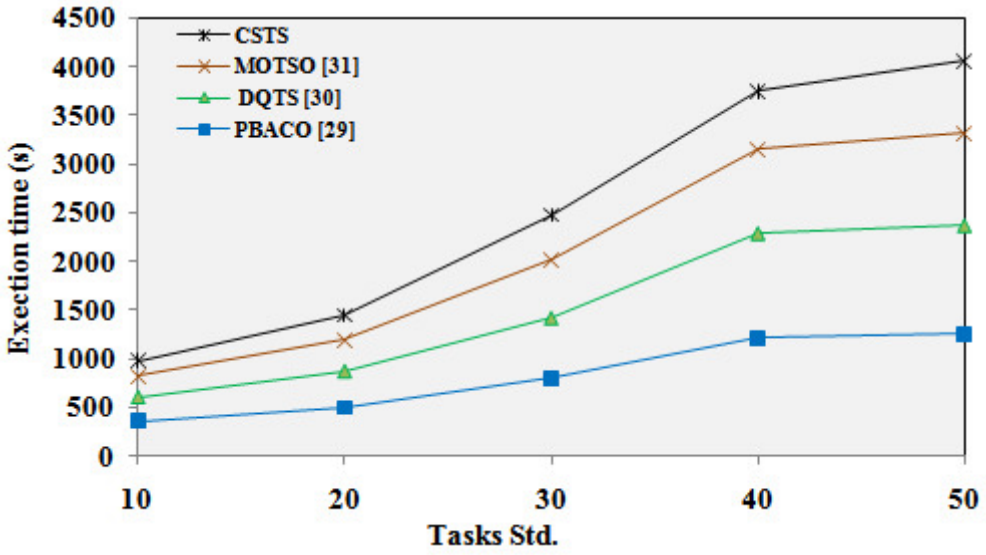

Figure 7. Execution time with task standard format.

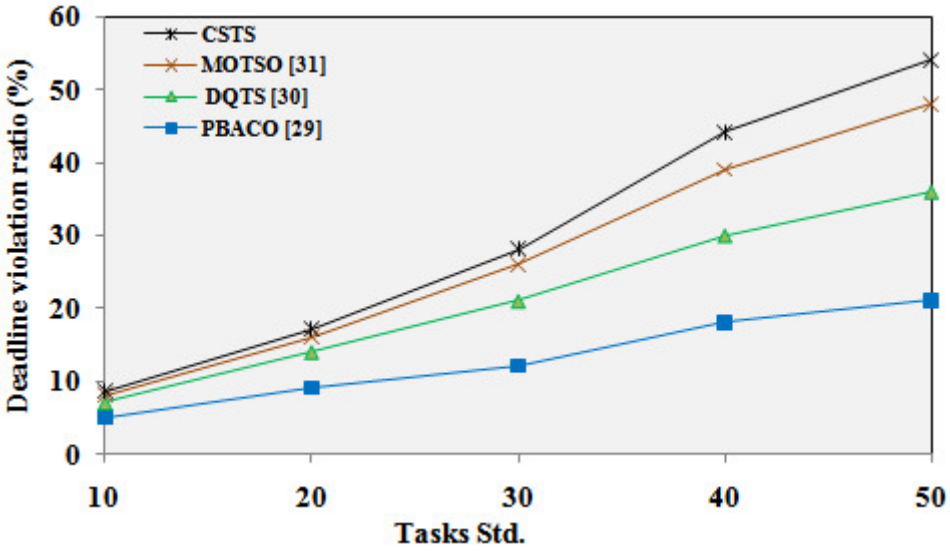

Figure 8. Deadline violation rate with task standard format.

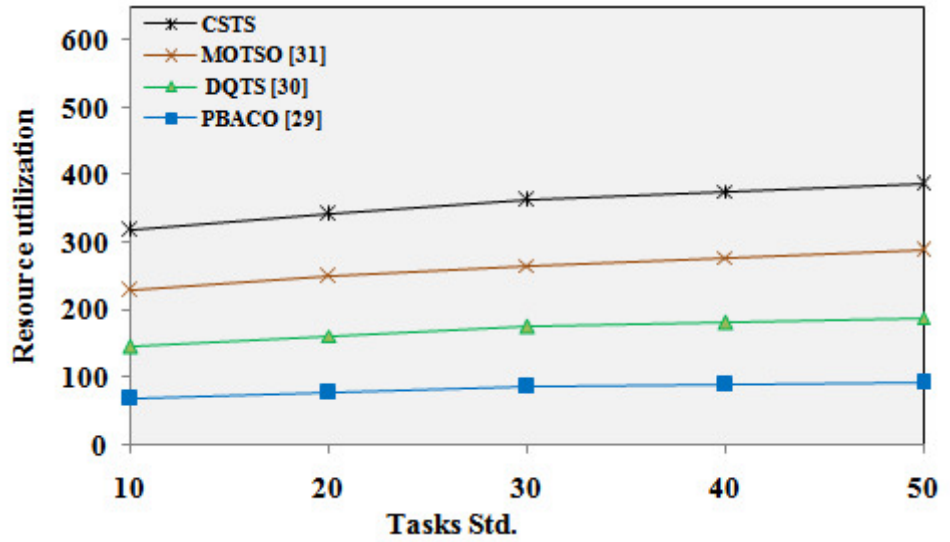

Figure 9. Resource utilization with task standard format.

This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CCBY-4.0). View this license's legal deed at http://creativecommons.org/licenses/by/4.0 and legal code at http://creativecommons.org/licenses/by/4.0/legalcode for more information.

REFERENCES

1. Singh S, Chana I. QRSF: QoS-aware resource scheduling framework in cloud computing. *J Supercomput*. 2015;71(1):241-292. doi:10.1007/s11227-014-1295-6

2. Wu Q, Qin G, Huang B. The research of multimedia cloud computing platform data dynamic task scheduling optimization method in multi core environment. *Multimed Tools Appl*. 2017;76(16):17163-17178. doi:10.1007/s11042-016-3667-9

3. Peng Z, Cui D, Zuo J, Li Q, Xu B, Lin W. Random task scheduling scheme based on reinforcement learning in cloud computing. *Cluster Comput*. 2015;18(4):1595-1607. doi:10.1007/s10586-015-0484-2

4. Pradeep K, Prem Jacob T. A hybrid approach for task scheduling using the cuckoo and harmony search in cloud computing environment. *Wireless Pers Commun*. 2018;101(4):2287-2311. doi:10.1007/s11277-018-5816-0

5. Jena T, Mohanty JR. GA-based customer-conscious resource allocation and task scheduling in multi-cloud computing. *Arab J Sci Eng*. 2018;43(8):4115-4130. doi:10.1007/s13369-017-2766-x

6. Gawali MB, Shinde SK. Task scheduling and resource allocation in cloud computing using a heuristic approach. *J Cloud Comp*. 2018;7(1):1-16. doi:10.1186/s13677-018-0105-8

7. Tang C, Hao M, Wei X, Chen W. Energy-aware task scheduling in mobile cloud computing. *Distrib Parallel Databases*. 2018;36(3):529-553. doi:10.1007/s10619-018-7231-7

8. Kamalinia A, Ghaffari A. Hybrid task scheduling method for cloud computing by genetic and DE algorithms. *Wireless Pers Commun*. 2017;97(4):6301-6323. doi:10.1007/s11277-017-4839-2

9. Abdulhamid SM, Abd Latiff MS, Madni SHH, Abdullahi M. Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm. *Neural Comput & Applic*. 2018;29(1):279-293. doi:10.1007/s00521-016-2448-8

10. Li Y, Chen M, Dai W, Qiu M. Energy optimization with dynamic task scheduling mobile cloud computing. *IEEE Systems Journal*. 2017;11(1):96-105. doi:10.1109/jsyst.2015.2442994

11. Rimal BP, Maier M. Workflow scheduling in multi-tenant cloud computing environments. *IEEE Trans Parallel Distrib Syst*. 2017;28(1):290-304. doi:10.1109/tpds.2016.2556668

12. Yuan H, Bi J, Tan W, Zhou M, Li BH, Li J. TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds. *IEEE Trans Cybern*. 2017;47(11):3658-3668. doi:10.1109/tcyb.2016.2574766

13. Shah-Mansouri H, Wong VWS, Schober R. Joint optimal pricing and task scheduling in mobile cloud computing systems. *IEEE Trans Wireless Commun*. 2017;16(8):5218-5232. doi:10.1109/twc.2017.2707084

14. Zuo L, Dong S, Shu L, Zhu C, Han G. A multiqueue interlacing peak scheduling method based on tasks' classification in cloud computing. *IEEE Systems Journal*. 2018;12(2):1518-1530. doi:10.1109/jsyst.2016.2542251

15. Chen M, Hao Y, Lai CF, Wu D, Li Y, Hwang K. Opportunistic task scheduling over co-located clouds in mobile environment. *IEEE Trans Serv Comput*. 2018;11(3):549-561. doi:10.1109/tsc.2016.2589247

16. Zhang P, Zhou M. Dynamic cloud task scheduling based on a two-stage strategy. *IEEE Trans Automat Sci Eng*. 2018;15(2):772-783. doi:10.1109/tase.2017.2693688

17. Sun F, Hou F, Cheng N, et al. Cooperative task scheduling for computation offloading in vehicular cloud. *IEEE Trans Veh Technol*. 2018;67(11):11049-11061. doi:10.1109/tvt.2018.2868013

18. Xiong Y, Huang S, Wu M, She J, Jiang K. A Johnson's-rule-based genetic algorithm for two-stage-task scheduling problem in data-centers of cloud computing. *IEEE Trans Cloud Comput*. 2019;7(3):597-610. doi:10.1109/tcc.2017.2693187

19. Jena T, Mohanty JR. GA-based customer-conscious resource allocation and task scheduling in multi-cloud computing. *Arab J Sci Eng*. 2018;43(8):4115-4130. doi:10.1007/s13369-017-2766-x

20. Lavanya M, Shanthi B, Saravanan S. Multi objective task scheduling algorithm based on SLA and processing time suitable for cloud environment. *Computer Communications*. 2020;151:183-195. doi:10.1016/j.comcom.2019.12.050

21. Ding D, Fan X, Zhao Y, Kang K, Yin Q, Zeng J. Q-learning based dynamic task scheduling for energy-efficient cloud computing. *Future Generation Computer Systems*. 2020;108:361-371. doi:10.1016/j.future.2020.02.018

22. Samriya JK, Kumar N. An optimal SLA based task scheduling aid of hybrid fuzzy TOPSIS-PSO algorithm in cloud environment. *Mater Today: Proc*. Published online December 7, 2020.

23. Sharma M, Garg R. An artificial neural network based approach for energy efficient task scheduling in cloud data centers. *Sustain Comput: Inform Syst*. 2020;26:100373. doi:10.1016/j.suscom.2020.100373

24. Wei X. Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing. *J Ambient Intell Human Comput*. Published online October 21, 2020:1-12. doi:10.1007/s12652-020-02614-7

25. Jing W, Zhao C, Miao Q, Song H, Chen G. QoS-DPSO: QoS-aware task scheduling for cloud computing system. *J Netw Syst Manage*. 2021;29(1):1-29. doi:10.1007/s10922-020-09573-6

26. Abualigah L, Alkhrabsheh M. Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing. *J Supercomput*. 2022;78(1):740-765. doi:10.1007/s11227-021-03915-0

27. Dubey K, Sharma SC. A hybrid multi-faceted task scheduling algorithm for cloud computing environment. *Int J Syst Assur Eng Manag*. Published online March 15, 2021:1-15. doi:10.1007/s13198-021-01084-0

28. Zuo L, Shu L, Dong S, Zhu C, Hara T. A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. *IEEE Access*. 2015;3:2687-2699. doi:10.1109/access.2015.2508940

29. Tong Z, Chen H, Deng X, Li K, Li K. A scheduling scheme in the cloud computing environment using deep Q-learning. *Information Sciences*. 2020;512:1170-1191. doi:10.1016/j.ins.2019.10.035

30. Guo X. Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm. *Alex Eng J*. 2021;60(6):5603-5609. doi:10.1016/j.aej.2021.04.051

31. Al Daweri MS, Abdullah S, Ariffin KAZ. A migration-based cuttlefish algorithm with short-term memory for optimization problems. *IEEE Access*. 2020;8:70270-70292. doi:10.1109/access.2020.2986509

32. Braik M, Sheta A, Al-Hiary H. A novel meta-heuristic search algorithm for solving optimization problems: capuchin search algorithm. *Neural Comput & Applic*. 2021;33(7):2515-2547. doi:10.1007/s00521-020-05145-6