Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.029854
Article

Tech Science Press

Oppositional Red Fox Optimization Based Task Scheduling Scheme for Cloud Environment

B. Chellapraba[1,*], D. Manohari[2], K. Periyakaruppan[3] and M. S. Kavitha[4]

[1]Department of Information Technology, Karpagam Institute of Technology, Coimbatore, 641032, Tamilnadu, India
[2]Department of Computer Science and Engineering, St. Joseph's Institute of Technology, Chennai, 600119, India
[3]Department of Computer Science & Engineering, SNS College of Engineering, Coimbatore, 641107, India
[4]Department of Computer Science & Engineering, SNS College of Technology, Coimbatore, 641035, India
*Corresponding Author: B. Chellapraba. Email: chellapraba@gmail.com
Received: 13 March 2022; Accepted: 26 April 2022

Abstract: Owing to massive technological developments in Internet of Things (IoT) and cloud environment, cloud computing (CC) offers a highly flexible heterogeneous resource pool over the network, and clients could exploit various resources on demand. Since IoT-enabled models are restricted to resources and require crisp response, minimum latency, and maximum bandwidth, which are outside the capabilities. CC was handled as a resource-rich solution to aforementioned challenge. As high delay reduces the performance of the IoT enabled cloud platform, efficient utilization of task scheduling (TS) reduces the energy usage of the cloud infrastructure and increases the income of service provider via minimizing processing time of user job. Therefore, this article concentration on the design of an oppositional red fox optimization based task scheduling scheme (ORFO-TSS) for IoT enabled cloud environment. The presented ORFO-TSS model resolves the problem of allocating resources from the IoT based cloud platform. It achieves the makespan by performing optimum TS procedures with various aspects of incoming task. The designing of ORFO-TSS method includes the idea of oppositional based learning (OBL) as to traditional RFO approach in enhancing their efficiency. A wide-ranging experimental analysis was applied on the CloudSim platform. The experimental outcome highlighted the efficacy of the ORFO-TSS technique over existing approaches.

Keywords: Metaheuristics; task scheduling; cloud computing; internet of things; makespan; red fox optimizer

1 Introduction

Internet of Things (IoT) is the vital technique to form smart city because it enables objects or entities to deliver data and service to users by communicating and collaborating with others [1]. There has been a rapid progression that the multiple devices get interconnected to the system with the tremendous growth of the IoT. Once the device requests resource service from the cloud datacentre simultaneously, it would take a massive network bandwidth, as well as information access and data transmission would be slow [2]. Furthermore,

when some -sensitive requests namely emergency and medical are uploaded to the remote cloud to process, the delay created by bandwidth constraint and resource bottleneck of the cloud datacentre affects the quality of service (QoS). In the meantime, cloud computing (CC), a novel computing structure, was extensively employed in the last few decades. CC is a technique which focuses on providing a flexible heterogeneous source pool via the system, and users rent distinct resources on demand [3,4]. User procures and releases computing resource that is generally virtual machine (VM) with distinct provisions, based on the particular requirements within a limited period. Since these techniques are highly dependent on the Internet, the CC and IoT are strongly associated with the role. The IoT digitalizes various information and wisely manages equipment, and CC is utilized by a carrier for higher-speed information utilization, processing, and storage. CC provides the advantage of security, speed, and convenience that the lacks IoT, and the technique that makes intelligent analysis and the realtime dynamic management of the IoT consistent [5,6].

Even though implementation of IoT applications in CC has different benefits, many problems continue a challenge. Firstly, scheduling of tasks is considered an NP-hard problem [7]. Especially, scheduling of tasks represents the task assignment to virtual resources based on sequential implementation. Next, once a computing resource (VM) is released or leased, an appropriate handoff takes time [8]. Indeed, there are distinct kinds for the skilled data traffic in IoT paradigm and it requires distinct QoS limitations which should be resolved when increasing the offered Cloud resource usage [9]. Consequently, it can be important to present a proper scheduling approach for offering optimal queue management for multiple class data traffic to guarantee the likely usage of the IoT resource allocation amongst the heterogeneous schemes of personal devices and servers. There could be case where scheduling algorithm in CC model could not fulfil the QoS constraint. Also, this ineffective scheduling might result in low network throughput and undesirable lengthy delay [10].

This article focuses on the design of an oppositional red fox optimization based task scheduling scheme (ORFO-TSS) for IoT enabled cloud environment. The presented ORFO-TSS model resolves the challenge of allocating resources from the IoT based cloud platform. It realizes the makespan by performing task scheduling (TS) procedures with various aspects of incoming tasks. The designing of ORFO-TSS approach includes the idea of oppositional based learning (OBL) as to traditional RFO technique in enhancing its efficiency. A wide-ranging experimental analysis was applied on the CloudSim platform.

The rest of the paper is organized as follows. Section 2 offers a brief related works and Section 3 provides the proposed model. Next, Section 4 gives performance validation and Section 5 draws conclusions.

2 Related Works

In [11], the authors proposed a Hybrid ant genetic approach for scheduling tasks. The presented technique adopts features of genetic algorithm (GA) and ant colony optimization (ACO) and splits virtual machines and tasks into small groups. Afterward task allocation, pheromone is included in VMs. Bezdan et al. [12] present a method that is capable of finding an estimated (near-optimum) solution for multiobjective TS issues in cloud platforms and simultaneously reduces the searching time. Then proposed a hybridized bat algorithm, a swarm intelligence (SI) based approach, for multiobjective TS. In [13], integrates security effective using TS in CC with a hybrid ML (RATS-HM) method is projected for overcoming the problem. The presented method is described in the following: Firstly, an enhanced cuckoo search optimization (CSO) approach-based shorter scheduler for TS (ICS-TS) maximizes throughput and diminishes the makespan time. Next, a group optimization-based DNN (GO-DNN) for effectual RA with distinct limitations including resource load and bandwidth.

Fu et al. [14] investigated the procedure of cloud TS and projected a particle swarm optimization (PSO) genetic hybrid approach-based phagocytosis PSO_PGA. Initially, generation of PSO is separated, in addition, the location of particles in the sub-population has been altered by crossover mutation of genetic algorithm and phagocytosis for expanding the searching space of the solution. In [15], an effectual hybridized scheduling method that imitates food gathering habits of the crow bird and the parasitic action of the cuckoo, called Cuckoo Crow Search approach (CCSA) was introduced to enhance the scheduling task. They often stares at its neighbour to search for an improved food source. In certain circumstance, the crow goes a further step and steal the neighbor's food. Amer et al. [16] introduce an adapted Harris hawk's optimizer (HHO), named elite learning HHO (ELHHO), for multiobjective scheduling issues. The modification is performed by utilizing a smart technology named elite OBL for enhancing the superiority of exploration stage of the typical HHO approach.

3 The Proposed Model

In this article, a novel ORFO-TSS algorithm has been developed to resolve the problem of allocating resources from the IoT based cloud platform. It accomplishes the makespan by performing optimal TS procedures with various aspects of incoming task. The designing of ORFO-TSS technique includes the idea of OBL as to typical RFO algorithm in enhancing its efficiency. Fig. 1 depicts the overall procedure of ORFO-TSS approach.

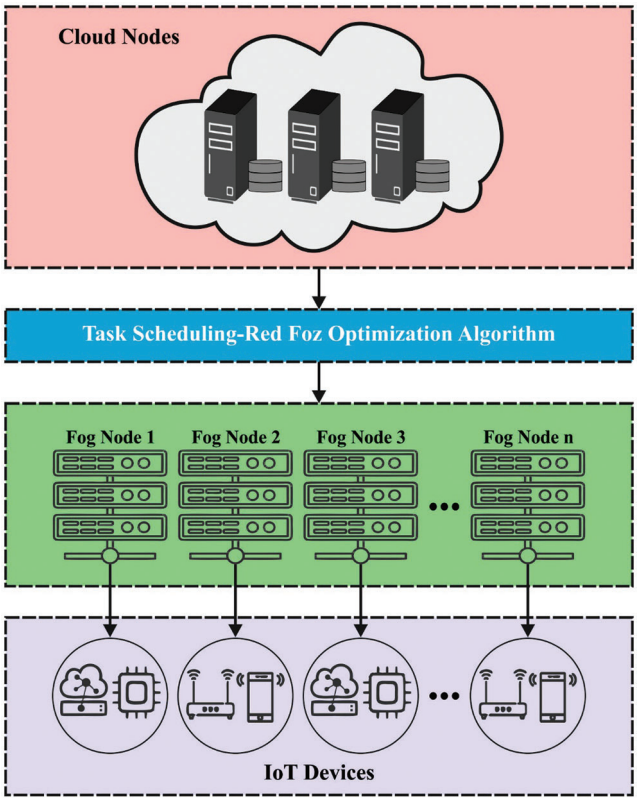

Figure 1: Overall process of proposed technology

3.1 Process Involved in ORFO Algorithm

According to hunting procedure of red foxes, a novel meta-heuristic technique is determined that is named as RFO technique. During the RFO approach, the red foxes search for food from territory. During the next stage, then moving on the territories for obtaining nearer sufficient to prey before attack. This technique begins with constant amount of arbitrary candidates as foxes whereas, all of them determined a point such that $\bar{x} = (x_0, x_1, \ldots, x_{n-1})$ and n defines co-ordinates. For discriminating all foxes $\bar{x}^i$ in iteration t, whereas i refers the fox number from the population, it can be present notation $\left(\bar{x}_j^i\right)^t$, whereas i demonstrates the co-ordinate as specified by the solution space dimensional. With assuming $f \in \Re^n$ as the condition function of n variables dependent upon the dimensional of searching space, and with assuming the notation $(\bar{x})^{(i)} = [(\bar{x}_0)^{(i)}, (\bar{x}_1)^{(i)}, (\bar{x}_{n-1})^{(i)}]$ refers the point from the space $[a, b]^n$ whereas $a, b \in \Re$. At that moment, $(\bar{x})^{(i)}$ is an optimum solution when the value of function $f((\bar{x})^{(i)})$ is global optimal on $[a, b]$. The outcomes of the estimated function by candidates were initially sorted dependent upon the fitness criteria, and for $(\bar{x}^{best})^t$, the square of Euclidean distance was measured for the candidates as [17]:

$$D\left(\left((\bar{x})^{(i)}\right)^t, \left((\bar{x})^{\text{best}}\right)^t\right) = \sqrt{\left\|\left((\bar{x})^{(i)}\right)^t - \left((\bar{x})^{\text{best}}\right)^t\right\|} \tag{1}$$

and the candidate moves nearby an optimum population as follows:

$$((\bar{x})^{(i)})^t = ((\bar{x})^{(i)})^t + \alpha \times sgn((((\bar{x})^{best})^t - \left(\bar{x}\right)^{(i)})^t\right) \tag{2}$$

whereas, α defines the arbitrary integer whereas $\alpha \in (0, D((\bar{x})^{best})^{\iota}, ((\bar{x})^{best})^t)$.

During the RFO technique, the observation and movement for delude prey but hunting as to local searching phase. For simulating a fox possibility modeling nearby prey, an arbitrary value $\gamma \in [0, 1]$ set from the iteration to every candidate.

$$\begin{cases} move\ doser\ if & \gamma > 3/4 \\ stay\ and\ hile\ i & \gamma \leq 3/4 \end{cases} \tag{3}$$

whereas, $\gamma \in [0, 1]$.

One other terms under the simulation is radius. The radius contains: a as an arbitrary value amongst 0 and 0.2 and φ_0 refers the arbitrary value amongst 0 and 2π. It is demonstrated in the subsequent formula [18]:

$$r = \begin{cases} a \times \sin(\varphi_0)/\varphi_0 & if\ \varphi_0 \neq 0 \\ \beta & if\ \varphi_0 = 0 \end{cases} \tag{4}$$

In which, β refers the arbitrary value from the range zero and one. It is formulated in the following:

$$\begin{cases} \chi_0^{New} = a \times r \times \cos(\varphi_1) + X_0^{actual} \\ x_1^{New} = a \times r \times \sin(\varphi_1) + a \times r \times \cos(\varphi_2) + x_1^{actual} \\ x_1^{New} = a \times r \times \sin(\varphi_1) + a \times r \times \sin(\varphi_2) + a \times r \times \cos(\varphi_3) + x_2^{actual} \\ \vdots \\ x_{n-1}^{New} = a \times r \times \sum_{k=1}^{n-2} \sin(\varphi_1) + a \times Y \times \cos(\varphi_{n-1}) + X_{n-2}^{actual} \\ x_{n-1}^{Neo} = a \times r \times \sin(\varphi_1) + a \times r \times \sin(\varphi_2) + \ldots + a \times r \times \sin(\varphi_{n-1}) + a \times r \times \sin(\varphi_{n-1}) + X_{n-a}^{actual} \end{cases} \tag{5}$$

In 5 percent of worse-case candidates are extracted and some upgraded candidates are then exchanged. Similarly, two of optimum individuals are attained as $(X(1))^t$ and $(X(2))^t$ as an alpha couple from iteration t. It can be mathematical as under:

$$H_c^t = \frac{1}{2}(X(1))^t - (X(2))^t \tag{6}$$

And the diameter of habitat utilizing Euclidean distance is attained as:

$$H_d^t = \left(\left\|(X(1))^t - (X(2))^t\right\|\right)^{\frac{1}{2}} \tag{7}$$

An arbitrary value, θ, is also regarded as:

$$\begin{cases} New\ nomadic\ candidate & if\ \theta > 0.45 \\ Reproduction\ of\ the\ alpha\ couple & if\ \theta \leq 045 \end{cases} \tag{8}$$

where, $\theta \in [0, 1]$.

Afterward, a novel candidate is reached by the alpha couple as follows:

$$(X^{rep})^t = \frac{\theta}{2}(X(1))^t - (X(2))^t \tag{9}$$

Even though the RFO approach provides better achievement to resolve the optimization problem, in some cases, it could not get make it to attain the optimal achievement due to the local optimum points trapping, lower consistency, and premature convergence. One amendment is to apply the oppositional based learning (OBL) method. This method determines that possible location from the solution space comprises an opposite position. It could be an appropriate improvement to enhance the exploration process. Therefore, it produces the opposite position of all the solutions, a comparison with its opposite is performed for selecting an optimal one as the novel solutions. By considering X_i as a solution among $X_i \in \underline{X}_i, \overline{X_i}$ and its opposite value $\check{X}_{ij}$, the equation can be modelled by the following. Fig. 2 illustrates the steps involved in RFO.

$$\left((\bar{x})^{(i)}\right)_{op}^t = \left((\bar{x})^{(i)}\right)_l^t + \left((\bar{x})^{(i)}\right)_u^t - \left((\bar{x})^{(i)}\right)^t \tag{10}$$

whereas, $\left((\bar{x})^{(i)}\right)^t$, $\left((\bar{x})^{(i)}\right)_{op}^t \in R^i$.

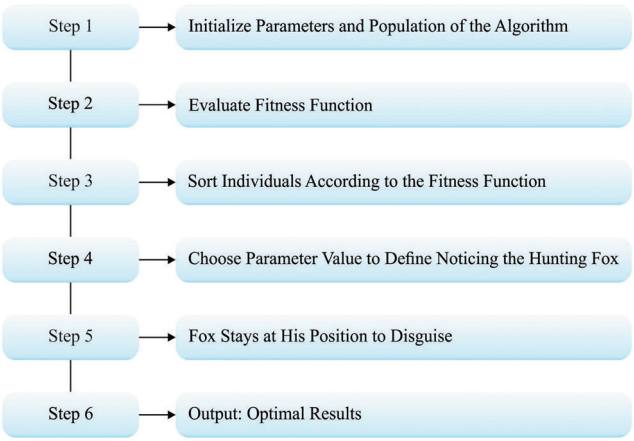

Figure 2: Steps involved in RFO

3.2 Steps Involved in ORFO-TSS Algorithm

The presented model fulfills the makespan by performing optimal TS procedures with various aspects of incoming tasks. In this approach, the cloud application was considered as collected of user jobs (UJs) that implement complex computing tasks employing cloud resources [19]. Let $UserJob = (U_1, U_2, U_3 \ldots U_N)$ represents the batch of user applications obtained at particular time. Every UJ (U_i) is represented as duplet $< a_i, d_i >$. In which a_j stands for the arrival time of UJ (U_i) and d_i stands for the purpose of UJs (U_i). During the scheduling approach, the UJ was allocated for the available DCs $(D_1, D_2, D_3 \ldots D_M)$, whereas $N \leq$ M. All DCs (D_i) has linked to duplet $< C_i,\ m_j > .c_i$, the price per unit time charged as DC to implement UJ, m_i demonstrates the count of available Processing Elements (PEs) to apply UJs. Each DC has a set of PEs $\{PE_1, PE_2 \ldots PE_m\}$ to compute allocated UJs. Every PE is linked to duplet $< s, p >$. $'s'$ and $'p'$ represent the executed speed and power consumption of all PEs correspondingly. The set of nodes $V = \{T_1 \ldots T_n\}$ represents the tasks, and the set of arcs demonstrates the control or data dependency among tasks. Optimal scheduling of UJs to accessible PEs from cloud in different DC is a vital objective of this work.

Considering the UJs U_i has assigned to DC (D_j). T_k implies the set of tasks of the UJs (U_i) was given to PE (P_j). Once the time need that applying T_k employ P_j is signified as G_j. The termination time of T_j is expressed as:

$$Finish(T_k) = start(T_k) + \Gamma_j \tag{11}$$

Therefore, the whole time vital to complete the UJ as D_j is demoted as $Makespan_j$ and defined as:

$$Makespan_j = \max\{Finish(T_k)\} \tag{12}$$

whereas $T_{(k=1\ldots n)}$ the tasks are assigned to D_j. The energy consumption for calculating the UJs (U_i) by DC D_j has measured as:

$$E_i = \sum_{k=1}^{n} (\Gamma_k \times p_k) \tag{13}$$

In which p_k denotes the power consumption per unit time through PEs $\left(P_j\right)$ for procedure offered task (T_k). The cost of processing the UJ as DC D_j is calculated as:

$$C_j = c_j \times Makespan_j \tag{14}$$

whereas c_j denotes the price per unit time charged by DC D_j to applied the UJs.

The consumption $\left(U_j\right)$ of DCs $\left(D_j\right)$ is estimated as:

$$U_j = \frac{Makespan_j}{\max\{Makespan_k\}, k = 1 \ldots M} \tag{15}$$

An objective function of this projected approach are expressed as:

$$Minimize\ Makespan_j\ j = 1..M \tag{16}$$

$$Minimize\ E_j \mathrm{j} = 1..\mathrm{M} \tag{17}$$

$$Minimize\left\{\sum_{j=1}^{M} C_j\right\} \tag{18}$$

$$Maximize\ U_j \mathrm{j} = 1..\mathrm{M} \tag{19}$$

Subjected to:

- The UJ requirement end before deadline (d_i)

- Every UJ is allocated to only one DC.

The count of UJs is reduced to the count of present DC at a particular time.

4 Performance Validation

The performance validation of the ORFO-TSS approach is carried out using distinct types of tasks based on the number of sizes namely extra-large (EL), large (LAR), medium (MED), and small (SMA). Tab. 1 provides a brief average response time (ART) and average turnaround time (ATAT) of the ORFO-TSS model with recent methods.

Table 1: ATAT and ART analysis of ORFO-TSS technique with other scheduling methods

Methods	Average turnaround time (s)	Average response time (s)
RR algorithm	0.196	0.144
FCFS algorithm	0.172	0.120
SJF algorithm	0.169	0.119
Firefly algorithm	0.158	0.115
FIMPSO-TS	0.142	0.108
CAIOA-TS	0.118	0.095
ORFO-TSS	0.085	0.074

Fig. 3 inspects a detailed ATAT examination of the ORFO-TSS model with existing models. The figure indicated that the round robin (RR) model has gained poor outcomes with higher ATAT of 0.196 s. Followed by, the first come first serve (FCFS) and shortest job first (SJF) models have resulted in slightly enhanced performance with ATAT of 0.172 s and 0.169 s respectively. Along with that, the Firefly and FIMPSO-TS models have reached reasonable ATAT of 0.158 s and 0.142 s respectively. However, the ORFO-TSS model has outperformed the other methods with maximum ATAT of 0.085.

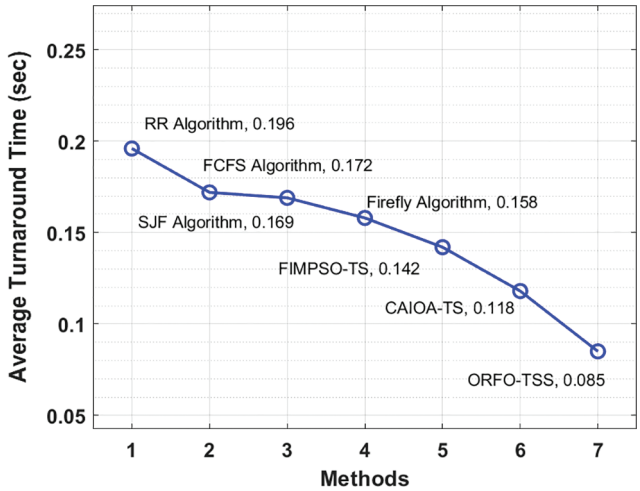

Figure 3: ATAT analysis of ORFO-TSS technique with recent algorithms

Fig. 4 examines a detailed ART examination of the ORFO-TSS model with existing models. The figure indicated that the RR approach has gained poor outcomes with maximum ART of 0.144 s. Afterward, the FCFS and SJF models have resulted in somewhat enhanced performance with ART of 0.120 s and 0.119 s respectively. Likewise, the Firefly and FIMPSO-TS models have reached reasonable ART of 0.115 s and 0.108 s correspondingly. Eventually, the ORFO-TSS technique has outperformed the other methods with maximal ART of 0.074.

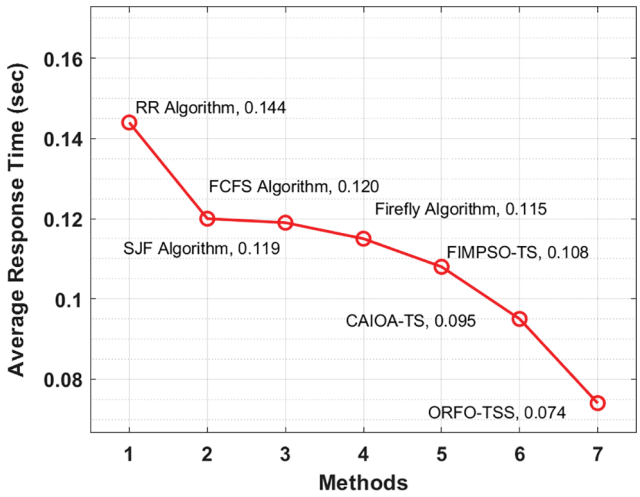

Figure 4: ART analysis of ORFO-TSS technique with recent algorithms

Tab. 2 and Fig. 5 illustrate a brief CPU utilization (CPUU) investigation of the ORFO-TSS model with other models on distinct types of tasks [20,21]. The results indicated that the ORFO-TSS model has reached effectual outcomes with higher CPUU. For instance, with small tasks, the ORFO-TSS model has obtained higher CPUU of 77.78% whereas the ACO-TS, GWO-TS, IPSO-TS, FIMPSO-TS, and CAIOA-TS models have reached lower CPUU of 58.01%, 65.91%, 67.64%, 71.17%, and 75.68% respectively. Moreover, with large tasks, the ORFO-TSS methodology has obtained superior CPUU of 98.33% whereas the ACO-TS, GWO-TS, IPSO-TS, FIMPSO-TS, and CAIOA-TS algorithms have reached to lower CPUU of 78.50%, 86.67%, 86.66%, 91.59%, and 96.18% correspondingly.

Table 2: CPU utilization analysis ORFO-TSS technique in terms of various measures

CPU utilization (%)				
Methods	Small	Medium	Large	Extra-Large
ACO-TS	58.01	69.73	78.50	88.69
GWO-TS	65.91	75.13	86.67	94.19
IPSO-TS	67.64	75.97	86.66	96.20
FIMPSO-TS	71.17	77.20	91.59	97.52
CAIOA-TS	75.68	83.92	96.18	98.11
ORFO-TSS	77.78	85.96	98.33	99.13

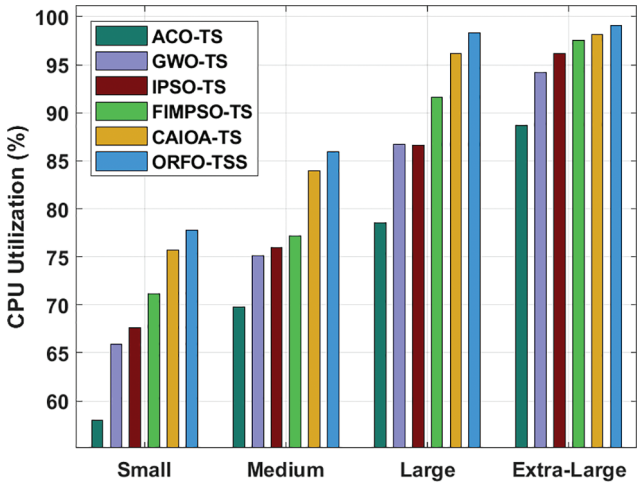

Figure 5: CPUU analysis ORFO-TSS technique interms of various measures

Tab. 3 and Fig. 6 depict a brief memory utilization (MU) investigation of the ORFO-TSS model with other models on distinct types of tasks. The results designated that the ORFO-TSS model has achieved effectual outcomes with higher MU. For sample, with small tasks, the ORFO-TSS model has obtained higher MU of 75.95% whereas the ACO-TS, GWO-TS, IPSO-TS, FIMPSO-TS, and CAIOA-TS methods have reached to lower MU of 53.83%, 57.54%, 57.65%, 60.98%, and 68.73% respectively. Additionally, with large tasks, the ORFO-TSS model has obtained higher MU of 95.64% whereas the ACO-TS, GWO-TS, IPSO-TS, FIMPSO-TS, and CAIOA-TS approaches have reached minimal MU of 70.89%, 79.49%, 84.23%, 83.64%, and 90.77% correspondingly.

Table 3: Memory utilization analysis ORFO-TSS technique with respect to distinct measures

	Memory utilization (%)			
Methods	Small	Medium	Large	Extra-Large
ACO-TS	53.83	63.16	70.89	83.57
GWO-TS	57.54	67.84	79.49	89.39
IPSO-TS	57.65	69.44	84.23	90.75
FIMPSO-TS	60.98	71.39	83.64	92.01
CAIOA-TS	68.73	75.71	90.77	94.45
ORFO-TSS	75.95	77.66	95.64	96.61

A detailed makespan examination of the ORFO-TSS approach with other existing models is compared in Tab. 4 and Fig. 7. The experimental results indicated that the ORFO-TSS model has accomplished effectual outcomes with minimal makespan. For instance, with small tasks, the ORFO-TSS model has provided minimal makespan of 38.69 s whereas the ACO-TS, GWO-TS, IPSO-TS, FIMPSO-TS, and CAIOA-TS models have obtained maximum makespan of 51.12 s, 48.47 s, 47.15 s, 45.52 s, and 43.93 s respectively. In line with, with large tasks, the ORFO-TSS approach has provided minimal makespan of 135.26 s whereas the ACO-TS, GWO-TS, IPSO-TS, FIMPSO-TS, and CAIOA-TS models have obtained maximal makespan of 181.34 s, 154.31 s, 151.13 s, 150.22 s, and 144.44 s correspondingly.

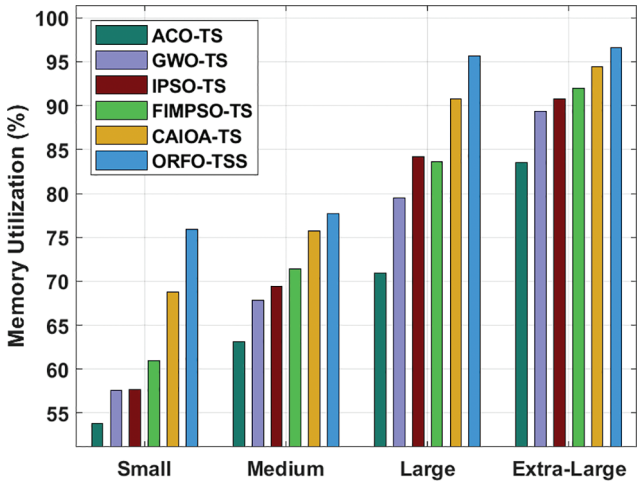

Figure 6: MU analysis ORFO-TSS technique interms of various measures

Table 4: Makespan analysis of ORFO-TSS technique with existing algorithms

	Makespan (s)			
Methods	Small	Medium	Large	Extra-Large
ACO-TS	51.12	99.3	181.34	261.72
GWO-TS	48.47	91.65	154.31	176.44
IPSO-TS	47.15	88.93	151.13	175.69
FIMPSO-TS	45.52	84.79	150.22	160.82
CAIOA-TS	43.93	80.91	144.44	167.88
ORFO-TSS	38.69	77.86	135.26	148.65

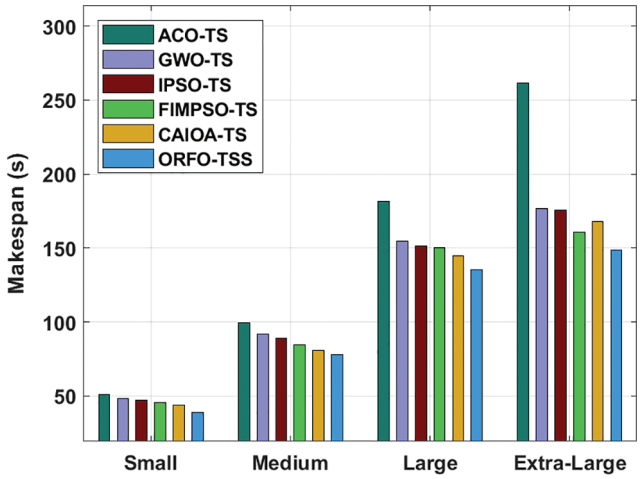

Figure 7: Makespan analysis of ORFO-TSS technique with existing algorithms

Tab. 5 and Fig. 8 demonstrate a brief average throughput (ATHPT) analysis of the ORFO-TSS technique with other techniques. The results indicated that the ORFO-TSS model has reached effectual outcomes with higher ATHPT. For instance, with small tasks, the ORFO-TSS approach has attained higher ATHPT of 99.78% whereas the ACO-TS, GWO-TS, IPSO-TS, FIMPSO-TS, and CAIOA-TS models have reached lower ATHPT of 89.42%, 95.27%, 99.23%, 99.70%, and 98.73% correspondingly. Furthermore, with large tasks, the ORFO-TSS technique has obtained higher ATHPT of 92.10% whereas the ACO-TS, GWO-TS, IPSO-TS, FIMPSO-TS, and CAIOA-TS techniques have reached to minimal ATHPT of 63.01%, 73.13%, 71.91%, 75.16%, and 86.91% correspondingly.

Table 5: Average throughput analysis of ORFO-TSS technique with existing algorithms

Average throughput (%)				
Methods	Small	Medium	Large	Extra-Large
ACO-TS	89.42	74.94	63.01	52.74
GWO-TS	95.27	82.57	73.13	64.26
IPSO-TS	99.23	82.74	71.91	68.88
FIMPSO-TS	99.70	87.62	75.16	73.31
CAIOA-TS	98.73	90.71	86.91	77.65
ORFO-TSS	99.78	95.96	92.10	89.80

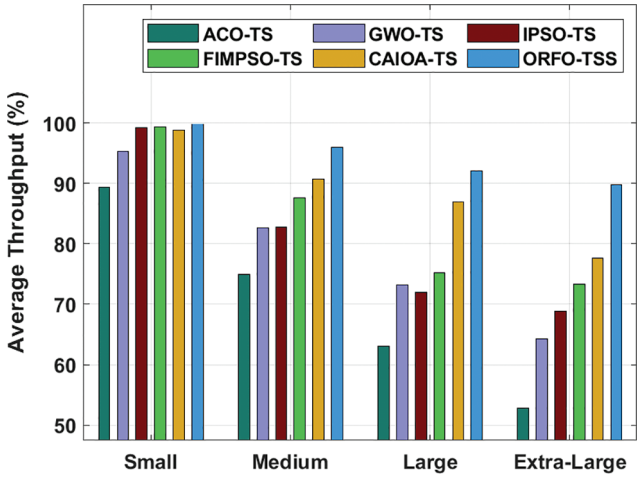

Figure 8: Average throughput analysis of ORFO-TSS technique with existing algorithms

After examining the above mentioned tables and discussion, it is evident that the ORFO-TSS model has accomplished maximum scheduling efficiency over the other models.

5 Conclusion

In this article, a novel ORFO-TSS algorithm has been developed to resolve the problem of allocating resources from the IoT based cloud platform. It fulfils the makespan by performing optimum TS procedures with various aspects of incoming tasks. The designing of ORFO-TSS method includes the idea of OBL as to typical RFO algorithm in enhancing its efficiency. A wide-ranging experimental

analysis was applied on the CloudSim platform. The experimental outcome highlighted the efficacy of the ORFO-TSS technique over existing approaches. Thus, the ORFO-TSS technique can be exploited for optimizing the efficacy of the IoT enabled cloud environment. In future, hybrid deep learning models can be employed to schedule the sources that exist in the IoT enabled cloud environment.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] E. H. Houssein, A. G. Gad, Y. M. Wazery and P. N. Suganthan, "Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends," *Swarm and Evolutionary Computation*, vol. 62, no. 3, pp. 100841, 2021.

[2] S. E. Shukri, R. A. Sayyed, A. Hudaib and S. Mirjalili, "Enhanced multi-verse optimizer for task scheduling in cloud computing environments," *Expert Systems with Applications*, vol. 168, no. 4, pp. 114230, 2021.

[3] S. Velliangiri, P. Karthikeyan, V. M. 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.

[4] M. A. 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.

[5] K. Karthikeyan, R. Sunder, K. Shankar, S. K. Lakshmanaprabu, V. Vijayakumar *et al.,* "Energy consumption analysis of virtual machine migration in cloud using hybrid swarm optimization (ABC-BA)," *Journal of Supercomputing*, vol. 76, no. 5, pp. 3374–3390, 2020.

[6] P. Bal, S. Mohapatra, T. Das, K. Srinivasan and Y. Hu, "A joint resource allocation, security with efficient task scheduling in cloud computing using hybrid machine learning techniques," *Sensors*, vol. 22, no. 3, pp. 1242, 2022.

[7] W. Jing, C. Zhao, Q. Miao, H. Song and G. Chen, "QoS-DPSO: QoS-aware task scheduling for cloud computing system," *Journal of Network and Systems Management*, vol. 29, no. 1, pp. 5, 2021.

[8] M. S. Sanaj and P. M. J. Prathap, "An efficient approach to the map-reduce framework and genetic algorithm based whale optimization algorithm for task scheduling in cloud computing environment," *Materials Today: Proceedings*, vol. 37, pp. 3199–3208, 2021.

[9] H. B. Alla, S. B. Alla, A. Ezzati and A. Touhafi, "A novel multiclass priority algorithm for task scheduling in cloud computing," *Journal of Supercomputing*, vol. 77, no. 10, pp. 11514–11555, 2021.

[10] R. Masadeh, N. Alsharman, A. Sharieh, B. A. Mahafzah and A. Abdulrahman, "Task scheduling on cloud computing based on sea lion optimization algorithm," *International Journal of Web Information Systems*, vol. 17, no. 2, pp. 99–116, 2021.

[11] M. S. Ajmal, Z. Iqbal, F. Z. Khan, M. Ahmad, I. Ahmad *et al.,* "Hybrid ant genetic algorithm for efficient task scheduling in cloud data centers," *Computers & Electrical Engineering*, vol. 95, no. 3, pp. 107419, 2021.

[12] T. Bezdan, M. Zivkovic, N. Bacanin, I. Strumberger, E. Tuba *et al.,* "Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm," *Journal of Intelligent & Fuzzy Systems*, vol. 42, no. 1, pp. 411–423, 2021.

[13] P. Bal, S. Mohapatra, T. Das, K. Srinivasan and Y. Hu, "A joint resource allocation, security with efficient task scheduling in cloud computing using hybrid machine learning techniques," *Sensors*, vol. 22, no. 3, pp. 1242, 2022.

[14] X. Fu, Y. Sun, H. Wang and H. Li, "Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm," *Cluster Computing*, vol. 51, no. 7, pp. 9, 2021.

[15] P. Krishnadoss, "CCSA: Hybrid cuckoo crow search algorithm for task scheduling in cloud computing," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 4, pp. 241–250, 2021.

[16] D. A. Amer, G. Attiya, I. Zeidan and A. A. Nasr, "Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing," *Journal of Supercomputing*, vol. 78, no. 2, pp. 2793–2818, 2022.

[17] E. Khorami, F. M. Babaei and A. Azadeh, "Optimal diagnosis of COVID-19 based on convolutional neural network and red fox optimization algorithm," *Computational Intelligence and Neuroscience*, vol. 2021, no. 3, pp. 1–11, 2021.

[18] M. Zhang, Z. Xu, X. Lu, Y. Liu, Q. Xiao *et al.,* "An optimal model identification for solid oxide fuel cell based on extreme learning machines optimized by improved red fox optimization algorithm," *International Journal of Hydrogen Energy*, vol. 46, no. 55, pp. 28270–28281, 2021.

[19] R. Raj, M. Varalatchoumy, V. L. Josephine, A. Jegatheesan, S. Kadry *et al.,* "Evolutionary algorithm based task scheduling in IOT enabled cloud environment," *Computers, Materials & Continua*, vol. 71, no. 1, pp. 1095–1109, 2022.

[20] A. Devaraj, M. Elhoseny, S. Dhanasekaran, E. Lydia and K. Shankar, "Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in Cloud computing environments," *Journal of Parallel and Distributed Computing*, vol. 142, no. 4, pp. 36–45, 2020.

[21] M. Golchi, S. Saraeian and M. Heydari, "A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation," *Computer Networks*, vol. 162, no. 6, pp. 106860, 2019.