Research Article

Narendrababu Reddy Gogireddy* and Phani Kumar Singamsetty

RWWO: an effective strategy for workflow scheduling in cloud computing with predicted energy using Deep Maxout Network

https://doi.org/10.1515/ehs-2021-0014
Received July 5, 2021; accepted September 21, 2021;
published online October 25, 2021

Abstract: Workflow scheduling is the recent researching area in the cloud environment, in which user satisfaction based on the cost and bandwidth is the most challenging task. Several research methods are devised to minimize the execution time and cost, which compromises the attributes. Hence, this research introduces an effective task scheduling mechanism in a cloud environment utilizing the Regressive Whale Water Optimization (RWWO) algorithm, which is derived by the integration of Regressive Whale Optimization (RWO) and Water Cycle Algorithm (WCA). The fitness parameters utilized are Quality of Service (QoS), resource utilization, and predicted energy. However, predicted energy is determined using Deep Maxout Network. Moreover, the proposed RWWO + Deep Maxout Network achieved a minimum task scheduling time of 0.0208, minimum task scheduling cost of 0.0017, minimum predicted energy of 0.1971, and maximum resource utilization of 0.9999.

Keywords: cloud computing; Deep Maxout Network; multitask scheduling; regressive whale optimization (RWO); water cycle algorithm (WCA).

Introduction

Cloud computing is the latest technology to improve virtualized resources, developed for consumers or end-users in a dynamic core to offer better and trustworthy services (Dillon, Wu, and Chang 2010). Cloud computing is a metered technology at different standards over the virtualized system to several consumers (Stephanakis et al. 2013). Also, it is a rapidly developing advancement that improves the computerization of both Information Technology (IT) infrastructure and IT resources. Cloud computing has had tremendous growth in research fields for the last few years and it is considered a system for computation that provides the resource to cloud users. Cloud computing satisfies the heterogeneous requirements utilizing Google mail, Amazon cloud, Google File System (GFS), and Hadoop framework are the more eminent reservoir of cloud computing resources (Jana, Chakraborty, and Mandal 2019; Netaji and Bhole 2020). The cloud computing services charge the users for entering the valuable resources over the network. The enterprises receive the resources as services rather than investing the expenses towards the resources, and thus, consumers can minimize their funding cost over the resources. Service Level Agreement (SLAs) is developed among service providers and consumers to point out guarantees of service providers to users (Kumar and Venkatesan 2019; Michael Mahesh 2020). The information available in the social media (Reddy Bojja et al. 2020) motivates the recommendation system (Hung and Chang 2019) in business (Hien Bui et al. 2021) and education (Hung 2020). The primary intention of the cloud is to offer an effective strategy for accurate manipulation of computational characteristics impelled in an organization and to assist the enterprise in exploiting consumer requirements (Al-Maytami et al. 2019). Due to the massive expansion and increasing demand for cloud computing resources (Singh and Chana 2016; Tsai, Lin, and Ke 2016), energy consumption is considered a major constraint in complicated cloud data centers. High energy utilization leads to heavy expenses and generates unrestricted amounts of heat emissions that frequently cause performance degradation and cannot operate appropriately (Ding et al. 2020).

The limitation of Task scheduling in the cloud environment is an eminent optimization issue. Cloud users always need an efficient load balancing model depending on distributed computing work for their effective deliveries. A

***Corresponding author: Narendrababu Reddy Gogireddy**, GITAM (Deemed to be) University, Hyderabad, India,
E-mail: narendrababureddy14@gmail.com
Phani Kumar Singamsetty, GITAM (Deemed to be) University, Hyderabad, India, E-mail: phanikumar.s@gmail.com

highly-established cloud can optimize parameters like mobility, scalability, elasticity, availability, and minimum expense and improve throughput and storage capability (Beegom and Rajasree 2014; Jana, Chakraborty, and Mandal 2019). Task scheduling dispenses user works over resources to increase the consumption rate and reduce the work implementation time. Optimal task scheduling is a significant concept in the virtualization of the cloud (Tsai et al. 2014). The scheduling strategy is an NP-complete limitation, where the duration required for locating the solution changes concerning the size of the issue. The process of workflow scheduling (Juarez, Ejarque, and Badia 2018) is the most significant problem in the cloud environment, where efficient searching and resolutions are included in determining the best virtual machine (VM). Different computation-based performance metrics are utilized in the scheduling process, like network communication cost, traffic volume, and round trip (Abd Elaziz et al. 2019). Workflow scheduling directly influences the consumption of both cloud resources and Quality of Service (QoS) in response to consumer requests (Ding et al. 2020). The task scheduling mechanism in the cloud environment is outlined using the following phases: resource selection, resource searching, and task submission. The fundamental aim of cloud-based scheduling techniques is to reduce the task implementation time, reduce the transmission time of the task, minimize the execution expenses, and maximize balancing of load and resource availability (Karunakaran 2019; Masdari et al. 2017).

Task scheduling is broadly categorized into two categories, such as static and dynamic scheduling. In the former one, the scheduler learns about the task and resource details, such that the tasks are scheduled to the cloud service providers. However, in the latter one, task and resource particulars are unclear and possess more overhead than the former one. Therefore, the scheduler distributes the dynamic scheduling plans to consumer tasks with effective resources (Kumar and Venkatesan 2019). Optimal task scheduling can be categorized into three kinds: heuristic, meta-heuristic, and hybrid techniques. The heuristic task scheduling techniques deliver the best solutions to schedule the task, whereas in meta-heuristic task scheduling techniques can effectively handle search space to find the best result for workflow scheduling issues within the polynomial duration. The hybrid task technique is the combination of both heuristic and meta-heuristic task scheduling approaches (Abd Elaziz et al. 2019). So far, several algorithms have been developed to meet the requirements of scheduling the tasks, but they all face an NP-complete limitation. Different optimization methods have been utilized in the workflow scheduling process, like electro search, ant colony optimization, genetic algorithm, simulated annealing, tabu search, and particle swarm optimization. Due to the requirement of massive computing energy needed to implement the real-time task, it is essential to utilize an effective algorithm by considering all the functions required to improve workflow scheduling (Velliangiri et al. 2021).

The ultimate goal of this research is to design an effective strategy for workflow scheduling in a cloud environment employing a developed RWWO algorithm. The number of user requests in cloud computing is satisfied according to their needs. However, the problem arises when simultaneously many user requests arrive at the cloud requesting similar resources causes a lack of resources. In order to overcome such issues, a task scheduling technique using the proposed RWWO algorithm is implemented. This scheduling mechanism allocates the resources to the users depending on their implementation time, capacity, and energy. After that, the fitness function is evaluated according to three parameters, like predicted energy, QoS, and resource utilization. However, the Deep Maxout Network is utilized to compute the predicted energy. Moreover, the proposed RWWO algorithm is derived by combining Regressive Whale Optimization (RWO) and Water Cycle Algorithm (WCA).

The primary contribution of this research is given as below:

- **Proposed RWWO + Deep Maxout Network**: An effective task scheduling mechanism is developed by the proposed RWWO algorithm for cloud computing. Due to a shortage of resources, it seems difficult to allocate the user's tasks with the same resources. However, this limitation can be completely solved through the proposed RWWO algorithm by allocating the resources to the user's tasks depending on their capacity, time consumed by the execution process, etc. Moreover, the predicted energy is determined using the Deep Maxout network.

The paper's organization is organized as follows: Section 2 elaborates motivation for developing the proposed approach, and a literature survey of the conventional techniques is also explained. Section 3 describes the system model of cloud computing. Section 4 elaborates the developed RWWO + Deep Maxout Network, and the results and discussion are explained in Section 5, and the research comes to an end in Section 6.

Motivation

This section elaborates the literature review of the conventional approaches related to task scheduling mechanisms

and their merits and limitations that motivate the researchers to design and establish an effective strategy for task scheduling.

Literature survey

Various existing techniques of task scheduling mechanisms are described as follows: Jana, Chakraborty, and Mandal (2019) developed a modified particle swarm optimization (MPSO) method for effectively improving task scheduling. Here, two algorithms were considered for cloud scheduling, namely Max-Min, and Minimum algorithm of implementation time. The experimental results showed that the ratio of execution time and average scheduling time was good than that of the particle swarm optimization algorithm. The major benefit of utilizing this method was it considerably improves performance, especially in the cloud-based business sector. However, the developed method was unsuitable for both high degree and low degree load parameters in real-time applications. Kumar and Venkatesan (2019) designed an efficient algorithm named hybrid genetic-particle swarm optimization (HGPSO) to accomplish workflow scheduling. In this developed scheme, the tasks of the user were preserved in a queue manager. After that, priority was determined, and the most adaptable resources were assigned to the work if it was recurrent. Furthermore, the current tasks were evaluated and preserved on an on-demand service basis. Then the result of the on-demand queue was then applied to the developed HGPSO algorithm, which was derived by incorporating the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) algorithm. At last, the HGPSO algorithm evaluated the adaptable resources for consumer tasks. The developed algorithm minimized the implementation duration and enhanced the factor of scalability and availability. Moreover, only limited QoS parameters were considered as a major drawback as it slows down the task allocation process. Abd Elaziz et al. (2019) presented an algorithm called Moth Search Differential Evolution (MSDE) for minimizing the makespan that is needed to schedule massive works on several virtual machines (VMs). In order to enhance the exploitation phase, DE was used as a local search method. Here, a set of three analyzing series was performed to analyze the performance of the developed MSDE algorithm. The first experiment solved 20 global optimization issues, whereas the second and third set was to resolve the limitation of cloud task scheduling. The ultimate goal of developing the technique was to alleviate makespan to maximize the throughput of the cloud model. The developed approach effectively scheduled tasks to VM. However, it failed to enhance the time complexity of the developed scheme. Ding et al. (2020) designed a Q-learning depending task scheduling paradigm for energy-efficient cloud computing (QEEC). Here, QEEC consisted of two phases, namely centralized task dispatcher and Q-learning-based scheduler. The major task of the centralized task dispatcher was to implement M/M/s queuing system, in which received consumer requests were allocated to every server in the cloud. While in the second stage, more importance was given to the requests, which had task lifespan and task laxity. The QEEC framework was designed and developed to overcome the limitation of energy utilization in both workflow scheduling and task allocation. As a result, it achieved a very short response duration that leads to enhanced energy efficiency. Moreover, the developed scheme reduced the task response time and maximized the CPU utilization of each server.

Rjoub, Bentahar, and Wahab (2020) presented a trust-aware scheduling solution called Big Trust Scheduling was introduced here to encounter the issues, such as access control, intrusion detection, and authentication. The developed system consisted of three phases, namely VM's trust level computation, task priority level computation, and trust-aware scheduling. The ultimate aim of this developed scheme was to compute a trust rate for every VM depending upon its achievement and, after that, selected the works depending on its resource demands. However, they failed to evaluate the performance based on deep learning technique. Mansouri, Zade, and Javidi (2019) introduced a hybrid task scheduling called a Fuzzy system and modified Particle Swarm Optimization (FMPSO) method. It was derived by integrating the fuzzy concept with Modified particle swarm optimization. However, this strategy considered four-velocity updating techniques and utilized the roulette wheel selection method to improve global search capacity. After that, it exploited a mutation and crossover operator to encounter the limitations of PSO, like global optima. Finally, the fitness function was determined using a fuzzy inference system. Moreover, the input parameters considered in this system were the length of the task, CPU speed, RAM capacity, and overall execution duration. The method achieved better performances concerning makespan, efficiency, and execution time. However, it failed to overcome the issues, such as fault tolerance parameters of cloud and load balancing. Al-Maytami et al. (2019) devised an effective scheduling procedure utilizing Directed Acyclic Graph (DAG) depending on prediction of tasks computation time (PTCT) to determine the pre-eminent algorithm for cloud servers. This method effectively minimized the size of the Expected Time to Compute (ETC) matrix. Moreover, the developed

technique considerably reduced the overall time of the task and makespan. However, it failed to enhance the overall utilization of energy is remains a major drawback. Boveiri et al. (2019) introduced a Max-Min Ant System (MMAS), a high-performance approach to encounter static task-graph scheduling in a homogeneous multiprocessor field. The main intention of the developed approach was that it correctly implemented the priority values of the tasks to obtain the best task order. The major disadvantage of this developed scheme was limited infrastructure.

Major challenges

Some of the challenges confronted by the traditional approaches of task scheduling mechanism are explained as follows:

- The dynamic dedicated server scheduling (DDSS) approach developed in (Al-Turjman, Hasan, and Al-Rizzo 2019) effectively enhanced the QoS functions with respect to throughput and delay but failed to minimize the optimal energy consumption.
- In (Karunakaran 2019), GSA and NSGA algorithms provided strong global and local search capabilities with fast convergence speed. However, it failed to implement green cloud computing. In addition, the large power consumption of cloud data centers has remained a challenging task.
- The developed ETSA algorithm achieved better energy consumption and makespan, but it did not reveal the energy and execution cost, which is a major concern for future investigation (Panda and Jana 2019).
- The major barrier in (Chen et al. 2020) is that it had less capability to mitigate the scheduling overhead of the IWC approach in the presence of high workloads. Moreover, the approach was not suitable for various task workloads.
- An electro-search algorithm designed in (Velliangiri et al. 2021), provided an efficient performance in terms of cost and response duration. The only drawback of this method was the limited number of parameters.

System model

This section describes the system framework of the cloud environment. Typically, the environment of the cloud comprises a huge number of infrastructures and service providers to offer effective services to the consumers or end users according to their demands. The cloud environment consists of different physical machines, and it includes several virtual machines. The main purpose of utilizing the physical machines is to offer servers to the users based on their requests. However, an infinite number of requests is claimed for the same resource; there raises a huge problem in assigning the resources to the users due to a lack of sufficient resources. Each virtual machine consists of different configurations, such as memory, size of CPU, and cost for implementing the tasks. Hence, assigning an effective virtual machine that consumes a small amount of time and cost for implementing the task remains challenging in the cloud environment. In order to overcome such issues, it is essential to establish a workflow schedule. The scheduling mechanism sorts the request for offering service to end-users. Moreover, workflow is completely relying on the energy and capacity requires while implementing a task. The cloud environment comprises with n number of physical machines, and it is expressed as,

$$M = \{M_1, M_2, \ldots M_i, \ldots M_n\} \tag{1}$$

where, $|M|$ specifies the total count of physical machines available in cloud, the individual physical machines are represented as $M_1, M_2, \ldots M_i, \ldots M_n$ and M_i indicates the ith physical machine in cloud. The total virtual machines with respect to ith physical machine is represented as,

$$\left|I^i\right| = \left\{I_1^i, I_2^i, \ldots . I_j^i, \ldots . I_h^i\right\} \tag{2}$$

Here, I_i specifies the total number of virtual machines with respect to ith physical machine, h denotes total count of virtual machines and I_h^i denotes h count of virtual machines similar to ith physical machine. However, the jth virtual machine of ith physical machine is expressed as I_j^i. S_j^i implies the resource cost of jth virtual machine exist at ith physical machine, whereas the capacity of jth virtual machine available at ith physical machine is indicated as C_j^i. Let us consider that there are h count of virtual machines and it is denoted as I. The capacity and task length are the significant factors to be considered while allocating resources according to user requests. Let us assume the total tasks as G and it is given by,

$$G = \{g_1, g_2, \ldots g_c, \ldots g_l\} \tag{3}$$

Here, G denotes the total task for execution purpose and l specifies the total count of tasks present in cloud environment. Moreover, g_c indicates the cth task. However, each task includes a number of sub-tasks and sub-tasks in cth task is expressed using the following equation,

$$g_c = \{g_1^c, g_2^c, \ldots ., g_b^c, \ldots g_m^c\} \tag{4}$$

where, $g_1^c, g_2^c, \ldots ., g_b^c, \ldots g_m^c$ specifies the sub-tasks of cth task and m represents the total count of sub-tasks in cth task.

The decision of a virtual machine to implement the task depends upon the task length. Moreover, task length signifies the waiting time of a task for execution.

Proposed RWWO for task scheduling and energy prediction based on Deep Maxout Network

This section elaborates on the proposed RWWO algorithm for the effective scheduling of work in cloud computing. In the cloud environment, an infinite count of user requests is satisfied according to the user demands. If several user requests reach the cloud environment, seeking similar resources is challenging because there are limited resources, and proper scheduling is difficult. There exist different techniques related to scheduling mechanisms. Still, they all faced many hurdles while transmitting the data between the data centers due to the high expense of resources, communication overhead, etc. However, some conventional workflow scheduling mechanisms reduced the cost of resources and the execution time, but they failed to effectively handle the heavy workflow's complexity. In order to encounter the issues mentioned above, the developed RWWO algorithm is established to effectively handle the task scheduling mechanism based on priority. The task schedule mainly depends on three parameters, like QoS, resource utilization, and predicted energy. However, predicted energy is determined using Deep Maxout Network. The proposed RWWO algorithm is derived by incorporating RWO (Narendrababu Reddy and Phani Kumar 2019) and WCA (Eskandar et al. 2012). In addition, the proposed mechanism provides higher performance with low execution costs. Figure 1 represents the multi-objective scheduling utilizing the developed RWWO algorithm.

Multi-objective task scheduling

The requirement of task scheduling appears when the user requests the same resource simultaneously. The task scheduling mechanism is according to the fitness factor of the solution matrix. However, an effective solution is obtained

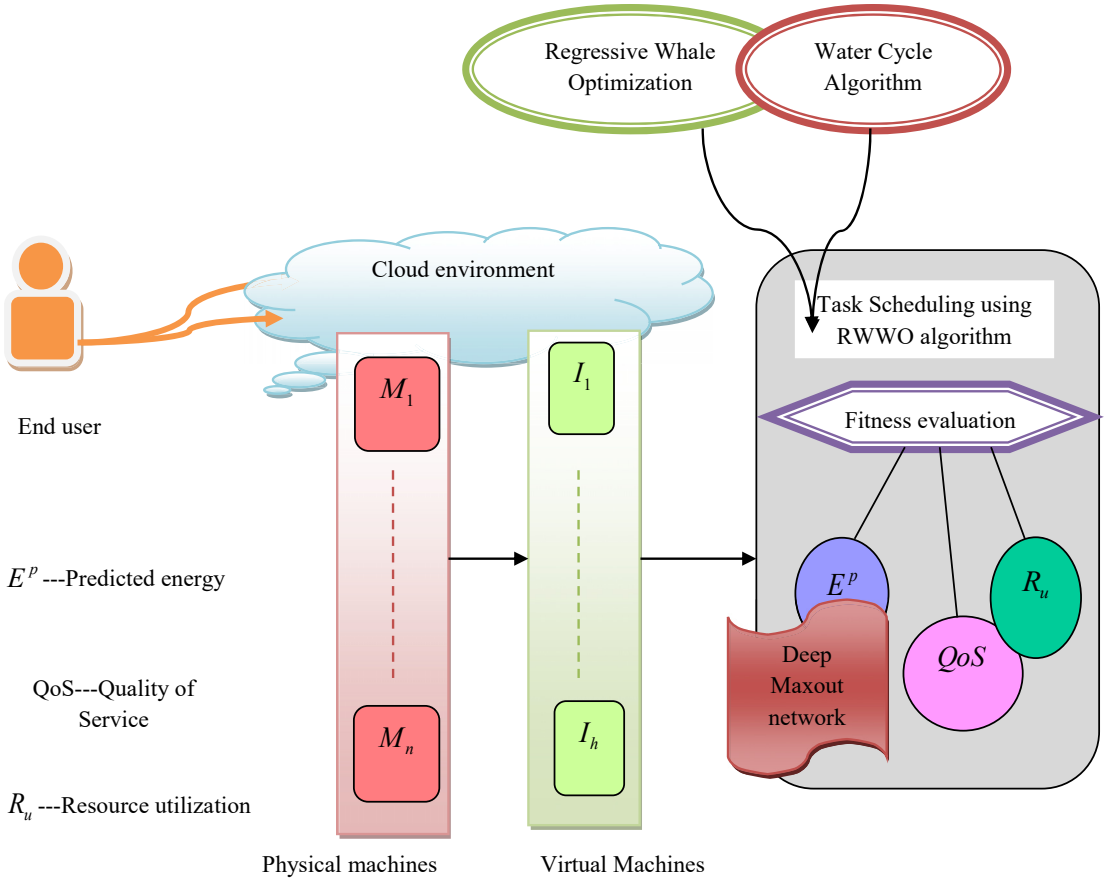

Figure 1: Schematic view of the proposed method of multi-objective task scheduling using RWWO algorithm.

based on fitness evaluation using three parameters: QoS, resource utilization, and predicted energy. The solution matrix with a reduced fitness function is chosen for implementing the work. In order to compute the fitness function, the solution matrix is achieved utilizing the irregularly chosen solution, which corresponds to the virtual machine implementing the assigned task. The solution matrix is expressed as A and A_{jc} is the solution matrix of cth work implemented in jth virtual machine. However, the group matrix is computed from the solution matrix, which is achieved by dividing the task's length and the virtual machine's ability to shine. Let us consider the matrix set as J and the evaluation of fitness is considered a significant parameter for determining the effective virtual machine to execute the work without degrading the method's performance. Specifically, the fitness function is the key factor that depends upon three factors, like QoS, resource utilization, and predicted energy. The predicted energy is determined using the Deep Maxout Network, which is briefly explained in Section 4.2. The effective result is achieved by selecting the fitness parameter with a minimum value. The fitness parameter is represented using the following equation,

$$\text{Fitness } U = \frac{1}{3} \times \left[(1 - R_u) + E^p + (1 - Q) \right] \tag{5}$$

where, R_u highlights resource utilization, E^p specifies predicted energy, obtained using Deep Maxout Network, and the QoS is expressed as Q. The resource utilization and the QoS are maximum in the fitness function.

QoS

QoS is a key parameter that evaluates the entire system performance. It is considered a significant parameter in cloud computing as it schedules the task effectively to the VMs with better performance and high quality. Generally, QoS is based on two functions, like cost and time consumed to execute the work in the virtual machine. However, cost and time parameters should be of the minimum range to offer a standard service to consumers' requests. The scheduling time and cost are denoted as $(1 - g)\,(1 - e)$, and the QoS is computed using the following equation,

$$Q = \frac{1}{2} \left[(1 - g)(1 - e) \right] \tag{6}$$

where, g and e specifies the time and cost while implementing the work in a virtual machine, respectively. The solution matrix evaluates the execution time of the work, and it is represented as,

$$g = \max_{\substack{\forall \\ c \in m}} \left[A_{jc} \times J_{jc} \right] \tag{7}$$

where, A_{jc} is the solution of cth task performed in jth virtual machine and J_{jc} is set the value of cth work implemented in jth virtual machine. The execution cost is represented as,

$$e = \frac{1}{B} \sum_{j=1}^{B} H_{jc} \times \left[S_{jc} \right] \tag{8}$$

$$\text{where, } H_{jc} = A_{jc} \times J_{jc} \tag{9}$$

Here, e denotes the expenses of implementing the work in jth virtual machine, the product factor is specified as H_{jc} and it is determined by employing J_{jc} and A_{jc}. However, S_{jc} implies the resource cost of jth virtual machine. Thus, the parameters with increased QoS are execution time, execution cost, and resource expenses. In order to maximize the achievement of the overall system, the functions should possess a minimum value.

Resource utilization

Resource utilization is another key constraint that should possess maximum value to enhance the entire performance of the model. It mainly depends on the set matrix and solution matrix. Besides, it is also based on set and solution values. The resource utilization is given by,

$$R_u = \frac{1}{m \times D_1} \times \sum_{i=1}^{m} A_{jc} \times J_{jc} \; ; 1 \le j \le B \; ; 1 \le c \le m \tag{10}$$

Here,

$$D_1 = \max (J) \tag{11}$$

where, J highlights the set matrix in m count of tasks, A_{jc} is the solution of cth task implemented in cth virtual machine and J_{jc} denotes the set value of cth task executed in jth virtual machine. However, the parameter D_1 is estimated according to the maximum set matrix range and B represents the number of virtual machines present in the cloud.

Energy

The predicted energy is a key constraint that should hold a minimum value to provide an effective model for effective performance. The predicted energy is obtained by applying the Deep Maxout Network. The energy is considered as an input, and also energy is required to run the tasks, and the output achieved through Deep Maxout Network is the predicted energy. The energy depends upon the solution value of cth task carried out at the jth virtual machine, and thus, the energy needed to execute the cth task implemented in jth virtual machine is expressed as,

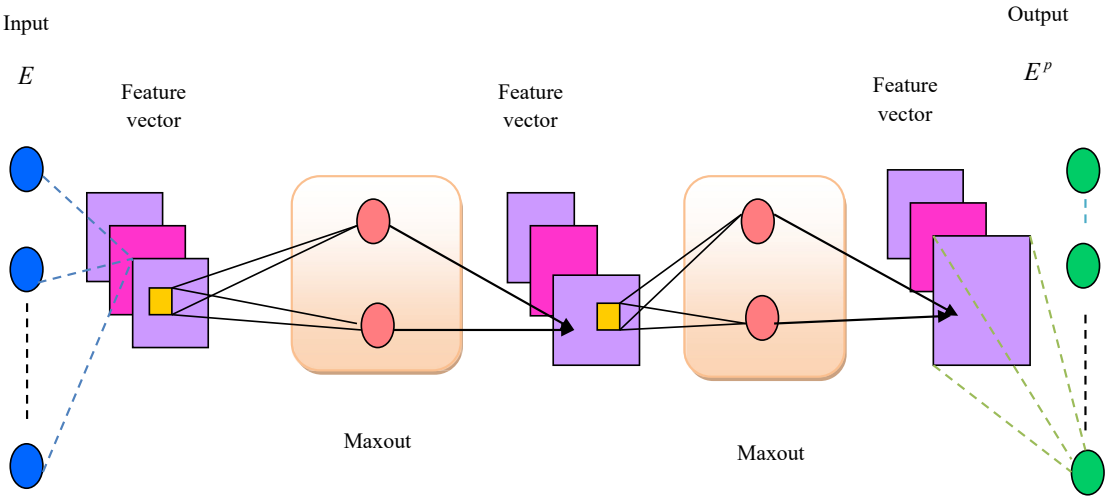

Figure 2: Architecture of Deep Maxout Network.

$$E = \frac{1}{m \times D_2} \sum_{c=1}^{m} A_{jc} \times \tau_{jc} \tag{12}$$

$$D_2 = \max(\tau) \tag{13}$$

where, the total count of tasks available in the virtual machine is termed as m, D_2 is a parameter that should be estimated, which is the high value required to perform the work. Finally, the predicted energy determined using the Deep Maxout Network is specified as E^p.

Energy prediction using Deep Maxout Network

Predicted energy is considered one of the fitness parameters computed using the Deep Maxout Network. Thus, the predicted energy enhances the overall performance of the system.

Architecture of Deep Maxout Network

Deep Maxout Network (Sun, Su, and Wang 2018) is an integral part of a Rectified Linear unit (ReLU), which employs the max operation on trainable linear functions. The main advantage of deep networks is that it effectively performs the complex structures of a huge set of activation functions. In the Deep Maxout Network, the activation function is substituted with a Maxout unit. The feature maps are achieved, and it is then applied to the higher layers. Moreover, each hidden unit in a Deep Maxout Network is considered a maxout unit, which enhances the potential in designing different distributions of various concepts. Deep Maxout Network is a type of trainable activation function comprised of a multi-layer structure. Figure 2 depicts the architecture of the Deep Maxout Network. The input applied to the Maxout Network is termed as E, the activation of a hidden unit is computed as follows:

$$d^1_{q,r} = \max_{r \in [1,\alpha_1]} E^v K_{....qr} + a_{qr} \tag{14}$$

$$d^2_{q,r} = \max_{r \in [1,\alpha_2]} d^{1^v}_{q,r} \; k_{...qr} + a_{qr} \tag{15}$$

$$\vdots$$

$$d^x_{q,r} = \max_{j \in [1,\alpha_x]} d^{x-1^v}_{q,r} K_{....qr} + a_{qr} \tag{16}$$

$$\vdots$$

$$d^y_{q,r} = \max_{r \in [1,\alpha_y]} d^{y-1v}_{q,r} K_{....qr} + a_{qr} \tag{17}$$

$$E^p = \max_{r \in [1,\alpha_y]} d^y_{q,r} \tag{18}$$

Multi-objective task scheduling utilizing proposed RWWO

This section describes the mechanism of multi-objective task scheduling employing the developed RWWO algorithm. The works are assigned to a specific virtual machine according to its execution cost, energy, and implementation time. In order to execute a work effectively, the RWWO algorithm is utilized as it achieves fast convergence for the optimal location with reduced time. Moreover, it prevents the cost of execution tasks, thereby enhancing the effectiveness of the task scheduling mechanism. During the advent of various user requests, the proposed RWWO algorithm achieves optimized results.

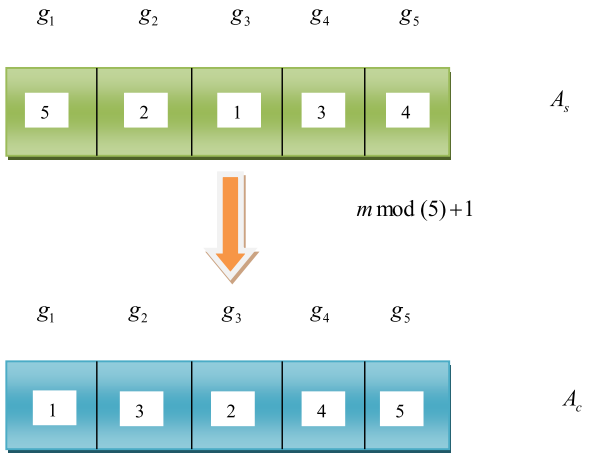

Figure 3: Solution encoding.

Solution encoding

Figure 3 portrays solution encoding. Initially, the solution is selected and then it is subjected for determining the fitness evaluation. The randomly chosen solution is sort out and correct solution is computed exploiting function $m \bmod (5)+1$. Here, the virtual machine I_5 executed the task g_1, task g_2 is executed by the virtual machine I_2, the virtual machine I_1 executed the task g_3 and it goes on. However, the stored solution undergoes an operation using $m \bmod (5)+1$ and the achieved solution identifies the virtual machine, which is suitable for a specific task. Let us assume there are two physical machines, which is illustrated as $M = \{M_1, M_2\}$. Here, two virtual machines is allocated to M_1, whereas three virtual machines is assigned to M_2. The total count of virtual machines is specified as $I = \{I_1, I_2, I_3, I_4, I_5\}$. A_0 specifies the randomly selected solution, which is used to evaluate the fitness function. The randomly selected solution is sorted out depending on virtual machines' potential and A_s denotes the organized solution. After that, the approximation is performed using the function $m \bmod (B)+1$ is represented as A_c. Then, the solution A_c is transformed into a matrix form, which is termed as A. A_{jc} is solution value of cth work executed in jth virtual machine. However, set matrix is determined based on two functions, such as capacity of the task, and length of the task. J implies the set matrix. Moreover, m implies the total count of tasks and five tasks are considered here.

Algorithmic procedure of RWWO

This section elaborates the proposed RWWO algorithm, which is derived by combining the RWO (Narendrababu Reddy and Phani Kumar 2019) algorithm and WCA (Eskandar et al. 2012). The WCA effectively solves the optimization issues based on the cost function. The RWO is an algorithm that is a modified version of WOA, which is easy to implement and has the fast convergence rate. The major benefit of utilizing this RWWO algorithm is that it finds an optimal position depending on the global search mechanism and provides minimum search cost. RWO algorithm is obtained by the incorporation of the regressive parameter of CAViaR's model (Engle and Manganelli 2004) with Whale Optimization Algorithm (WOA) (Chen et al. 2020). Thus, the incorporation of the three algorithms efficient performance enhancement is achieved in term of cost and time.

Step 1: Initialize the population

This is the first step of this algorithm, in which the size of the population is initialized. Let us consider the population of the whale as V, and it is expressed as,

$$V = \{V_1, V_2, \ldots V_o, \ldots V_{tp}\} \tag{19}$$

Here, tp denotes the total population with V_o, which represents the oth population. The whales search for their food depending upon the optimal position based on the global search mechanism.

Step 2: Evaluation of fitness function

The fitness function is utilized to determine the best optimal solution. The fitness value is determined utilizing three functions, namely QoS, resource utilization, and predicted energy.

Step 3: Localization of prey

The next step is to determine the location of prey depending on the current best solution. The localization process is generally based on the encircling phase. The whales encircle the prey at this phase that paves a way to the location of the prey. Thus, the prey has a fixed position, and the searching process is repeated concerning the optimal solution. The current optimal solution is updated utilizing the following equation,

$$\overrightarrow{f} = \left| \overrightarrow{N}.\overrightarrow{z}^{*}(k) - \overrightarrow{z}(k) \right| \tag{20}$$

where, $\overrightarrow{z}^{*}(k)$ represents the current best solution, position vector is specified as $z(k)$ and $\overrightarrow{f}$ depicts the distance between the whale and prey. The coefficient vector is represented as $\overrightarrow{N}$. The above equation is rewritten as,

$$z^{*}(k) = \frac{\overrightarrow{f} + \overrightarrow{z}(k)}{\overrightarrow{N}} \tag{21}$$

By considering the autoregressive model, position vector is expressed as,

$$\overrightarrow{z}(k) = \beta_0 + \beta_1 z_{t-1} + \beta_2 z_{t-2} + \beta_1 P(z_{t-1}) + \beta_2 P(z_{t-2}) \quad (22)$$

Substituting $\overrightarrow{z}(k)$ in Eq. (21), the equation becomes,

$$z^*(k) = \frac{\overrightarrow{f} + \beta_0 + \beta_1 z_{t-1} + \beta_2 z_{t-2} + \beta_1 P(z_{t-1}) + \beta_2 P(z_{t-2})}{\overrightarrow{N}} \quad (23)$$

where, $\overrightarrow{f}$ implies the distance vector, the regressive constant are denoted as $\beta_0, \beta_1, \beta_2$. $z^*(k)$ is the best position of prey. Hence, the best position is improved based on the below-expressed equation,

$$\overrightarrow{z}(k+1) = z^*(k) - \overrightarrow{W}.\overrightarrow{f} \quad (24)$$

Substituting the value of $z^*(k)$ in Eq. (24), then the equation becomes,

$$\overrightarrow{z}(k+1) = \frac{\overrightarrow{f} + \beta_0 + \beta_1 z_{t-1} + \beta_2 z_{t-2} + \beta_1 P(z_{t-1}) + \beta_2 P(z_{t-2})}{\overrightarrow{N}} - \overrightarrow{W}.\overrightarrow{f} \quad (25)$$

$$\overrightarrow{z}(k+1) = \frac{\overrightarrow{f} + \beta_0 + \beta_1 z_{t-1} + \beta_2 z_{t-2} + \beta_1 P(z_{t-1}) + \beta_2 P(z_{t-2}) - \left(\overrightarrow{W}.\overrightarrow{f}\right)\overrightarrow{N}}{\overrightarrow{N}} \quad (26)$$

$$\overrightarrow{z}(k+1) = \overrightarrow{f}\left(\frac{1 - \overrightarrow{W}.\overrightarrow{N}}{N}\right) + \frac{\beta_0 + \beta_1 z_{t-1} + \beta_2 z_{t-2} + \beta_1 P(z_{t-1}) + \beta_2 P(z_{t-2})}{\overrightarrow{N}} \quad (27)$$

Therefore, Eq. (27) is the newly updated position of prey by employing RWO algorithm. Now, substituting the Eq. (20) in Eq. (27), then the final update equation of RWO algorithm is,

$$\overrightarrow{z}(k+1) = \left(\overrightarrow{N}.\overrightarrow{z}^*(k) - \overrightarrow{z}(k)\right)\left(\frac{1 - \overrightarrow{W}\,\overrightarrow{N}}{N}\right) \quad (28)$$

where, $\overrightarrow{z}(k+1)$ represents the final update equation, $z^*(k)$ is the best position of prey, $\overrightarrow{z}^*(k)$ represents the current best solution, coefficient vector is represented as $\overrightarrow{N}$ and $\overrightarrow{W}$.

At this step, the WCA algorithm is incorporated with the RWO algorithm to provide an optimal global solution. WCA is a nature-inspired algorithm that depends on the mechanism of water cycle mechanism and how streams and rivers flow into the sea. The complexities in computational performance and then the convergence rate is enhanced using the below equation,

$$\overrightarrow{z}^*(k) = \overrightarrow{z}(k+1) - \sqrt{\omega}\, randw(1, X_{\text{var}}) \quad (29)$$

As $\overrightarrow{z}(k+1)$ is considered as the optimal solution in WCA, then the Eq. (28) is substituted with Eq. (29),

$$\overrightarrow{z}(k+1) = \left(\overrightarrow{N}\left(\overrightarrow{z}(k+1) - \sqrt{\omega}\, randw(1, X_{\text{var}})\right) - \overrightarrow{z}(k)\right)\left(\frac{1 - \overrightarrow{W}\,\overrightarrow{N}}{\overrightarrow{N}}\right) + \frac{\beta_0 + \beta_1 z_{t-1} + \beta_2 z_{t-2} + \beta_1 P(z_{t-1}) + \beta_2 P(z_{t-2})}{\overrightarrow{N}} \quad (30)$$

$$\overrightarrow{z}(k+1) = \left(\overrightarrow{N}\left(\overrightarrow{z}(k+1) - \left(\overrightarrow{N}\sqrt{\omega}\, randw(1, X_{\text{var}})\right) - \overrightarrow{z}(k)\right)\left(\frac{1 - \overrightarrow{W}\,\overrightarrow{N}}{\overrightarrow{N}}\right) + \frac{\beta_0 + \beta_1 z_{t-1} + \beta_2 z_{t-2} + \beta_1 P(z_{t-1}) + \beta_2 P(z_{t-2})}{\overrightarrow{N}}\right. \quad (31)$$

$$\overrightarrow{z}(k+1) = \left(\overrightarrow{N}\left(\overrightarrow{z}(k+1)\left(\frac{1 - \overrightarrow{W}\,\overrightarrow{N}}{\overrightarrow{N}}\right) - \overrightarrow{N}\sqrt{\omega}\, randw(1, X_{\text{var}})\right) + \overrightarrow{z}(k)\right)\left(\frac{1 - \overrightarrow{W}\,\overrightarrow{N}}{\overrightarrow{N}}\right) + \frac{\beta_0 + \beta_1 z_{t-1} + \beta_2 z_{t-2} + \beta_1 P(z_{t-1}) + \beta_2 P(z_{t-2})}{\overrightarrow{N}} \quad (32)$$

$$\overrightarrow{z}(k+1) - \left(\overrightarrow{N}\left(\overrightarrow{z}(k+1)\left(\frac{1 - \overrightarrow{W}\,\overrightarrow{N}}{\overrightarrow{N}}\right)\right.\right. = \frac{\beta_0 + \beta_1 z_{t-1} + \beta_2 z_{t-2} + \beta_1 P(z_{t-1}) + \beta_2 P(z_{t-2})}{\overrightarrow{N}} - \left(\overrightarrow{N}\sqrt{\omega}\, randw(1, X_{\text{var}})\right) + \overrightarrow{z}(k)\right)\left(\frac{1 - \overrightarrow{W}\,\overrightarrow{N}}{\overrightarrow{N}}\right) \quad (33)$$

$$\overrightarrow{z}(k+1) = \frac{1}{\overrightarrow{W}\,\overrightarrow{N}}\,\frac{\beta_0 + \beta_1 z_{t-1} + \beta_2 z_{t-2} + \beta_1 P(z_{t-1}) + \beta_2 P(z_{t-2})}{\overrightarrow{N}} - \left(\overrightarrow{N}\sqrt{\omega}\, randw(1, X_{\text{var}})\right) + \overrightarrow{z}(k)\left(\frac{1 - \overrightarrow{W}\,\overrightarrow{N}}{\overrightarrow{N}}\right) \quad (34)$$

Step 4: Update the location of prey

After finding the location of the prey, the location of the prey is identified. A parameter known as the probability factor is utilized to determine the position of the prey. If the probability factor is less than that of 0.5, the process of localization is carried out. If the probability factor is greater than that of 0.5, then the process of exploitation is performed. The exploitation phase consists of two steps: encircling the prey with the help of bubbles and position update of spiral movement. In position update of the spiral movement, the distance between the location of the prey and the whale is determined using the following equation,

$$\overrightarrow{f} = \overrightarrow{z}^*(k) - \overrightarrow{z}(k) \quad (35)$$

Rearranging the Eq. (35),

$$\overrightarrow{z}^*(k) = \overrightarrow{f}^*\ \overrightarrow{z}(k) \quad (36)$$

Substituting the value of $\vec{z}(k)$ expressed in Eq. (22) in Eq. (36), the obtained equation is represented as,

$$\vec{z}^{*}(k) = \vec{f}^{*} + \beta_0 + \beta_1 (z_t - 1) + \beta_2 (z_t - 2) + \beta_1 P (z_t - 1) + \beta_2 P (z_t - 2) \quad (37)$$

The position update in the exploration step is based on the following equation,

$$\vec{z}(k+1) = \vec{f}^{*} \cdot T^{kZ} \cdot \cos(2\pi Z) + z^{*}(k) \quad (38)$$

Substitute the value of $\vec{z}^{*}(k)$ in $\vec{z}(k+1)$,

$$\vec{z}(k+1) = \vec{f}^{*}\left(1 + T^{kZ}.\cos(2\pi Z)\right) + \beta_0 + \beta_1 (z_t - 1) + \beta_2 (z_t - 2) + \beta_1 P (z_t - 1) + \beta_2 P (z_t - 2) \quad (39)$$

Substitute Eq. (35) in Eq. (39),

$$\vec{z}(k+1) = \left(\vec{z}^{*}(k) - \vec{z}(k)\right)\left(1 + T^{kZ}.\cos(2\pi Z)\right) + \beta_0 + \beta_1 (z_t - 1) + \beta_2 (z_t - 2) + \beta_1 P (z_t - 1) + \beta_2 P (z_t - 2) \quad (40)$$

Now, incorporating the Eq. (29) of WCA algorithm,

$$\vec{z}^{*}(k+1) = \vec{Z}(k+1) - \sqrt{\omega}\, rand\, w\,(1, X_{\text{var}})$$

$$\vec{z}(k+1) = \left(\vec{z}(k+1) - \sqrt{\omega}\ rand\, w\,(1, X_{\text{var}}) - \vec{z}(k)\right)\left(1 + T^{kZ}.\cos(2\pi Z)\right) + \beta_0 + \beta_1 (z_t - 1) + \beta_2 (z_t - 2) + \beta_1 P (z_t - 1) + \beta_2 P (z_t - 2) \quad (41)$$

$$\vec{z}(k+1) = \vec{z}(k+1)\left(1 + T^{kZ}.\cos(2\pi Z)\right) - \left(\sqrt{\omega}\ rand\, w\,(1, X_{\text{var}}) + \vec{z}(k)\right)\left(1 + T^{kZ}.\cos(2\pi Z)\right) + \beta_0 + \beta_1 (z_t - 1) + \beta_2 (z_t - 2) + \beta_1 P (z_t - 1) + \beta_2 P (z_t - 2) \quad (42)$$

$$\vec{z}(k+1) - \vec{z}(k+1)\left(1 + T^{kZ}.\cos(2\pi Z)\right) = \beta_0 + \beta_1 (z_t - 1) + \beta_2 (z_t - 2) + \beta_1 P (z_t - 1) + \beta_2 P (z_t - 2) - \left(\sqrt{\omega}\ rand\, w\,(1, X_{\text{var}}) + \vec{z}(k)\right)\left(1 + T^{kZ}.\cos(2\pi Z)\right) \quad (43)$$

$$\vec{z}(k+1)\left(1 - 1 - T^{kZ}\cos(2\pi Z)\right) = \beta_0 + \beta_1 (z_t - 1) + \beta_2 (z_t - 2) + \beta_1 P (z_t - 1) + \beta_2 P (z_t - 2) - \left(\sqrt{\omega}\ rand\, w\,(1, X_{\text{var}}) + \vec{z}(k)\right)\left(1 + T^{kZ}.\cos(2\pi Z)\right) \quad (44)$$

$$\vec{z}(k+1) = \frac{1}{T^{kZ}\cos(2\pi Z)} \left[\begin{array}{l} \left(\sqrt{\omega}\ rand\, w\,(1, X_{\text{var}}) + \vec{z}(k)\right)\left(1 + T^{kZ}.\cos(2\pi Z)\right) - \beta_0 - \beta_1 (z_t - 1) - \beta_2 \\ (z_t - 2) - \beta_1 P (z_t - 1) - \beta_2 P (z_t - 2) \end{array}\right] \quad (45)$$

The above equation is the final updated equation of the RWWO algorithm.

However, in the exploration step, the position update is depending on the global search mechanism, and it is represented as,

$$\vec{z}(k+1) = \vec{z}_{\delta}\, \vec{W}.\vec{f} \quad (46)$$

$$\vec{z}(k+1) = \frac{\vec{f}\left(1 - \vec{W}.\vec{f}\right)}{\vec{N}} + \frac{\beta_0 + \beta_1 (z_t - 1) + \beta_2 (z_t - 2) + \beta_1 P (z_t - 1) + \beta_2 P (z_t - 2)}{\vec{N}} \quad (47)$$

Step 5: Termination

The process is continued until the best optimal solution is reached and the condition is satisfied. Then, the updated equation is considered as the best solution in the global search space. Finally, Algorithm 1 portrays the pseudo-code of the developed RWWO algorithm.

Algorithm 1. Pseudo code of RWWO algorithm

Sl. No	Pseudo code of developed RWWO algorithm
1	Input: $V = \{V_1, V_2, ..., V_o, ..., V_{tp}\}$
2	**Output:** $\vec{z}(k+1)$ **– Best position**
3	Parameters: $\vec{N}, \vec{W}, \gamma$
4	**Read**
5	Update the parameters
6	If ($\gamma < 0.5$)
7	And if ($\vec{W} < 1$)
8	Update the position using Eq. (27).
9	Else
10	Update the position using Eq. (45).
11	Else if
12	Update the position based on Eq. (46).
13	Compute the fitness parameter
14	Return the best position, $\vec{z}(k+1)$
15	End

Results and discussion

This section describes the results and discussion of the developed RWWO + Deep Maxout Network concerning the evaluation metrics.

Experimental setup

The RWWO + Deep Maxout Network experimentation is done in JAVA with 4 GB RAM and Intel Core i-3 processor.

Evaluation metrics

The performance of the proposed RWWO + Deep Maxout Network is analyzed using the evaluation metrics, namely QoS, resource utilization, and predicted energy.

QoS

QoS is the quality of service provided to users that depends on task scheduling cost and task scheduling time. However, the scheduling time and cost must be less for an effective task scheduling mechanism. Therefore, the QoS is calculated using Eqs. (7) and (8).

Resource utilization

Resource utilization is the number of resources utilized during the execution of a task, and it should be maximum for an effective mechanism. It is computed using Eq. (10).

Predicted energy

Predicted energy is derived from energy, and it remains maximum for effective performance and is calculated using Eq. (12).

Comparative methods

The performance of the proposed RWWO + Deep Maxout Network is analyzed with the conventional methods, such as ACO (Kumar and Venkatesan 2019), PSO (Boveiri et al. 2019), WOA (Stephanakis et al. 2013), and RWO (Narendrababu Reddy and Phani Kumar 2019).

Comparative analysis

This section elaborates the developed RWWO + Deep Maxout Network analysis using three experimental setups concerning the evaluation metrics, like QoS, resource utilization, and predicted energy. The setup-1 includes 10 physical machines (PM), 41 Virtual machines (VM), five tasks, and 17 sub-tasks. In experimental setup-2, the system is comprised of 10 PM, 43 VM, 10 tasks, and 23 sub-tasks. However, setup-3 includes 5 PM and 20 VM for scheduling five tasks with eight sub-tasks.

Analysis based on setup-1

Figure 4 depicts the analysis of the proposed approach using setup-1 concerning the evaluation metrics by changing the number of iterations. Figure 4a represents the analysis of task scheduling time. If the number of iteration = 10, the time attained by the developed RWWO + Deep Maxout Network is 0.185. However, the conventional schemes achieved the scheduling of 0.5132 for ACO, 0.5132 for PSO, 0.2149 for WOA, and 0.4654 for RWO. By varying the number of iteration to 40, the scheduling time achieved by the traditional techniques, like ACO is 0.5132, PSO is 0.3237, WOA is 0.2149, and RWO is 0.1217, respectively, and the proposed RWWO + Deep Maxout Network attained the scheduling time of 0.0208.

The analysis of task scheduling cost is depicted in Figure 4b. If the iteration = 10, the scheduling cost obtained by developed RWWO + Deep Maxout Network is 0.00815, while the conventional techniques, such as ACO, PSO, WOA, and RWO attained scheduling cost 0.0094, 0.0098, 0.0098, and 0.0118, respectively. Similarly, if the number of iteration = 40, cost achieved by the developed RWWO + Deep Maxout Network is 0.0017. However, the scheduling cost obtained by the conventional approaches is 0.0082 for ACO, 0.0091 for PSO, 0.0098 for WOA, and 0.0046 for RWO.

The analysis of predicted energy is represented in Figure 4c. When the number of iteration = 10, the predicted energy obtained by the proposed scheme is 0.290, whereas the existing methods achieved the predicted energy of 0.5003 for ACO, 0.4792 for PSO, 0.3638 for WOA, and 0.4315 for RWO. By increasing the iteration number to 40, predicted energy obtained by the traditional schemes, such as ACO is 0.4177, PSO is 0.4725, WOA is 0.3638, and RWO is 0.2859. However, the proposed RWWO + Deep Maxout Network attained the predicted energy is 0.197.

Figure 4d represents the analysis of resource utilization. When the number of iteration = 10, the resource utilization

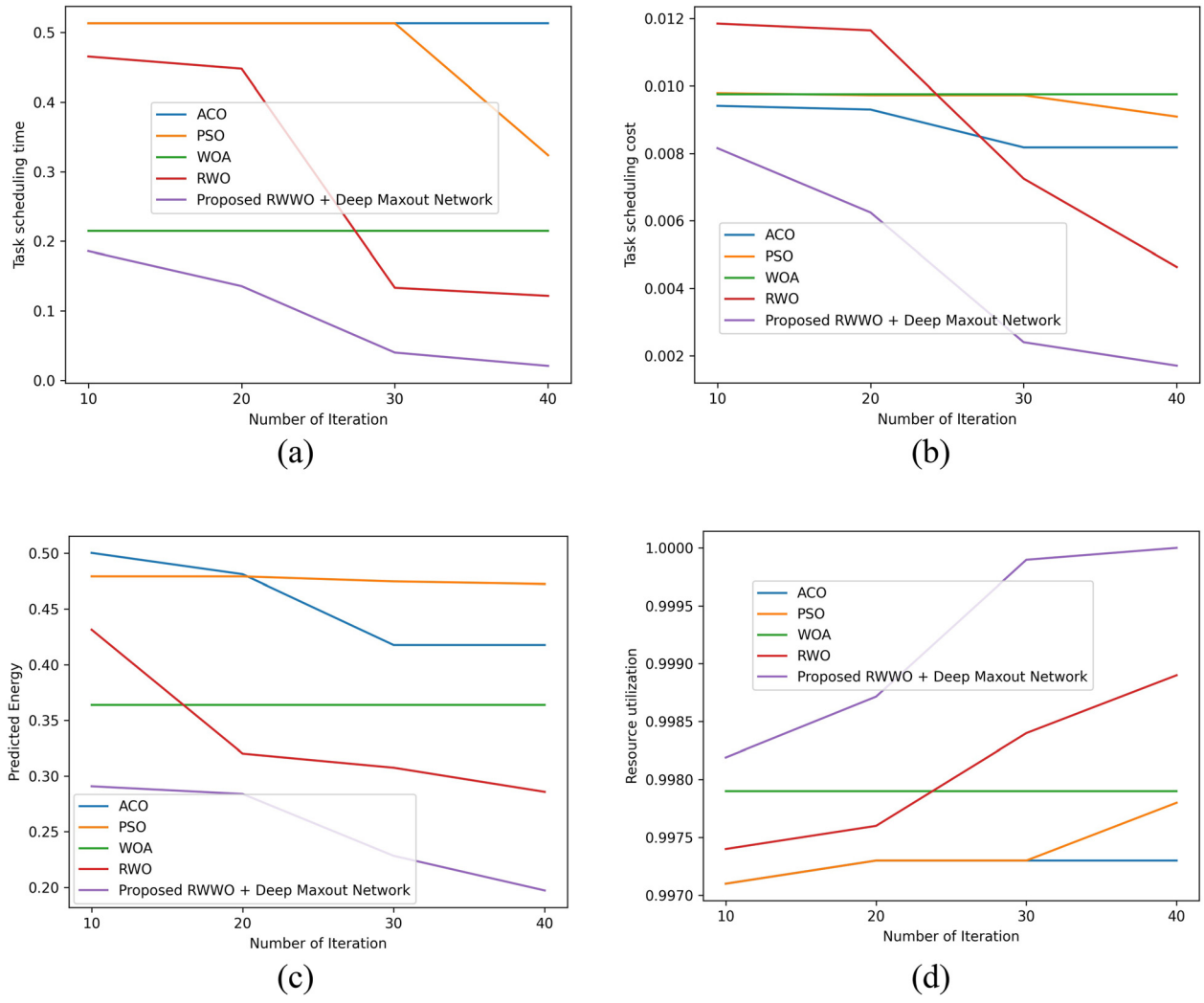

Figure 4: Analysis using experimental setup-1, a) task scheduling time, b) task scheduling cost, c) resource utilization, d) predicted energy.

achieved by the proposed RWWO + Deep Maxout Network is 0.998, whereas the existing methods, like ACO, is 0.9971, PSO is 0.9971, WOA is 0.9979, and RWO is 0.9974. Correspondingly, if the number of iteration = 40, the resource utilization achieved by the proposed RWWO + Deep Maxout Network, conventional schemes ACO, PSO, WOA, and RWO is 0.999, 0.9973, 0.9978, 0.9979, and 0.9989, respectively.

Analysis based on setup-2

Figure 5 illustrates the analysis using setup-2 concerning the performance metrics by changing the count of iterations. Figure 5a portrays the analysis of task scheduling time by changing the number of iterations. If the number of iteration = 10, the task scheduling time obtained by the proposed RWWO + Deep Maxout Network is 0.247. However, the time achieved by the conventional approaches, like ACO is 0.2701, PSO is 0.3408, WOA is 0.2771, and RWO is 0.5837. Similarly, if the number of iteration = 40, the task scheduling time attained by the developed scheme and the traditional schemes, such as ACO is 0.0218, PSO is 0.2701, WOA is 0.2583, and RWO is 0.1619.

Figure 5b depicts the proposed RWWO + Deep Maxout Network analysis concerning the task scheduling cost. If the number of iteration = 10, the cost attained by the traditional techniques, like ACO is 0.0111, PSO is 0.0141, WOA is 0.0129, and RWO is 0.0251. However, the proposed RWWO + Deep Maxout Network obtained the task scheduling cost is 0.0102. Likewise, the number of iteration = 40, the task scheduling cost achieved by developed RWWO + Deep Maxout Network is 0.0045, while the traditional techniques, such as ACO is 0.0111, PSO is

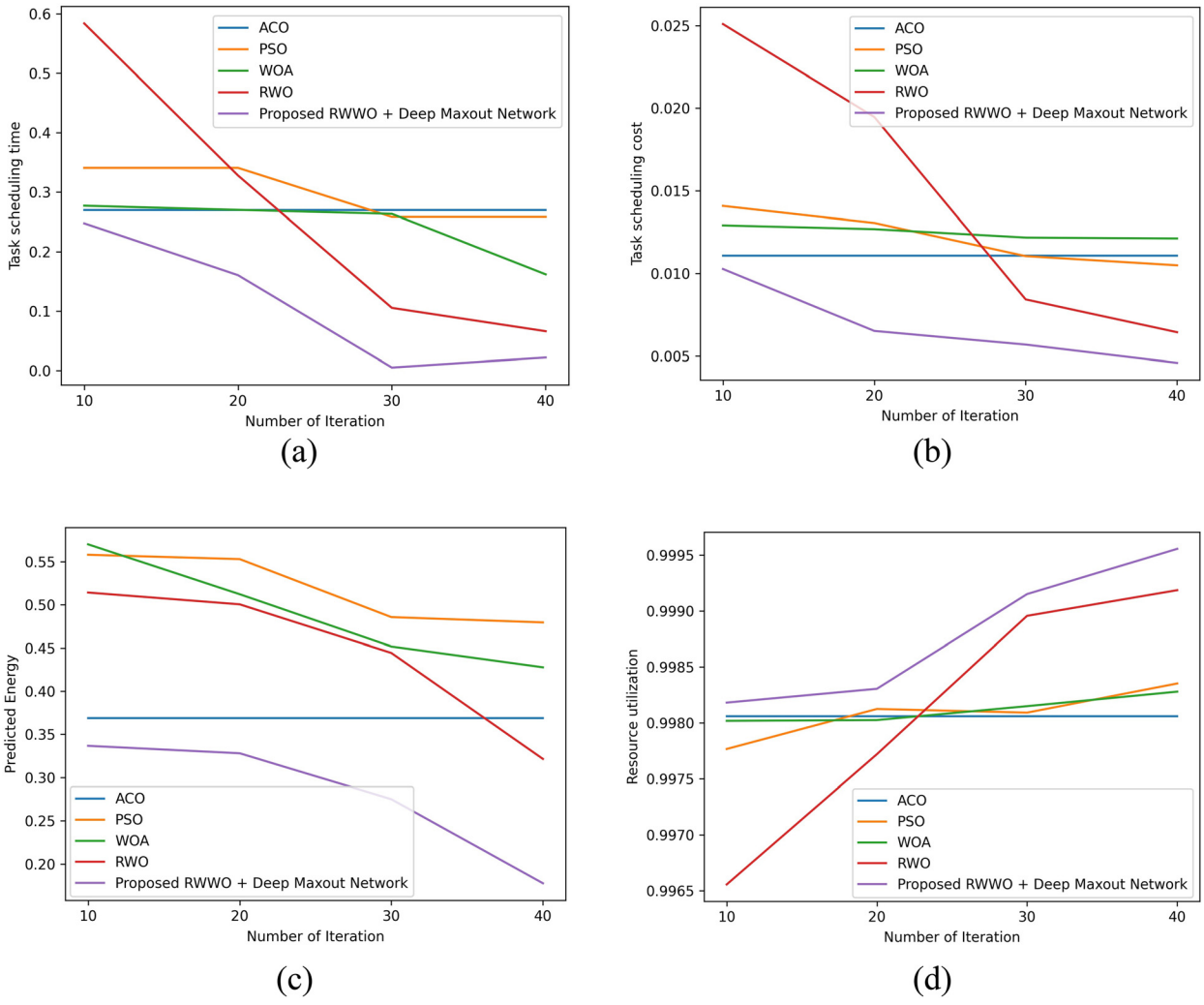

Figure 5: Analysis using experimental setup-2, a) task scheduling time, b) task scheduling cost, c) resource utilization, d) predicted energy.

0.0105, WOA is 0.0121, and RWO is 0.0064, attained the scheduling cost.

Figure 5c represents the proposed RWWO + Deep Maxout Network analysis concerning the predicted energy. If the number of iteration = 10, the proposed RWWO + Deep Maxout Network's predicted energy is 0.336, while the existing methods like ACO are 0.3688, PSO is 0.5580, WOA is 0.5700, and RWO is 0.5141. By varying the number of iteration = 40, the predicted energy attained by the proposed RWWO + Deep Maxout Network is 0.177.

The proposed RWWO + Deep Maxout Network analysis concerning resource utilization is represented in Figure 5d. If the number of iteration = 10, the resource utilization consumed by the proposed RWWO + Deep Maxout Network is 0.998. However, the conventional schemes, such as ACO, PSO, WOA, and RWO, utilized 0.9981, 0.9978, 0.9980, and 0.9966. Similarly, if the number of iteration = 40, the resource utilization attained by the conventional approaches, like ACO is 0.9981, PSO is 0.9984, WOA is 0.9983, and RWO is 0.9992. However, the proposed RWWO + Deep Maxout Network attained the resource utilization as 0.9995.

Analysis based on setup-3

Figure 6 illustrates the proposed RWWO + Deep Maxout Network analysis using setup-3 concerning the performance metrics. The analysis of task scheduling time is represented in Figure 6a. When number of iteration = 10, the scheduling time achieved by developed RWWO + Deep Maxout Network is 0.0173, whereas the conventional schemes, like ACO, PSO, WOA, and RWO attained the task scheduling time of 0.1324, 0.1228, 0.0242, and 0.0182. Similarly, if the number of iteration = 40, the task scheduling time attained by existing schemes, like ACO is 0.0238, PSO is 0.0224, WOA is 0.0224, and RWO is 0.0182. The proposed RWWO + Deep Maxout Network attained the task scheduling time as 0.0035.

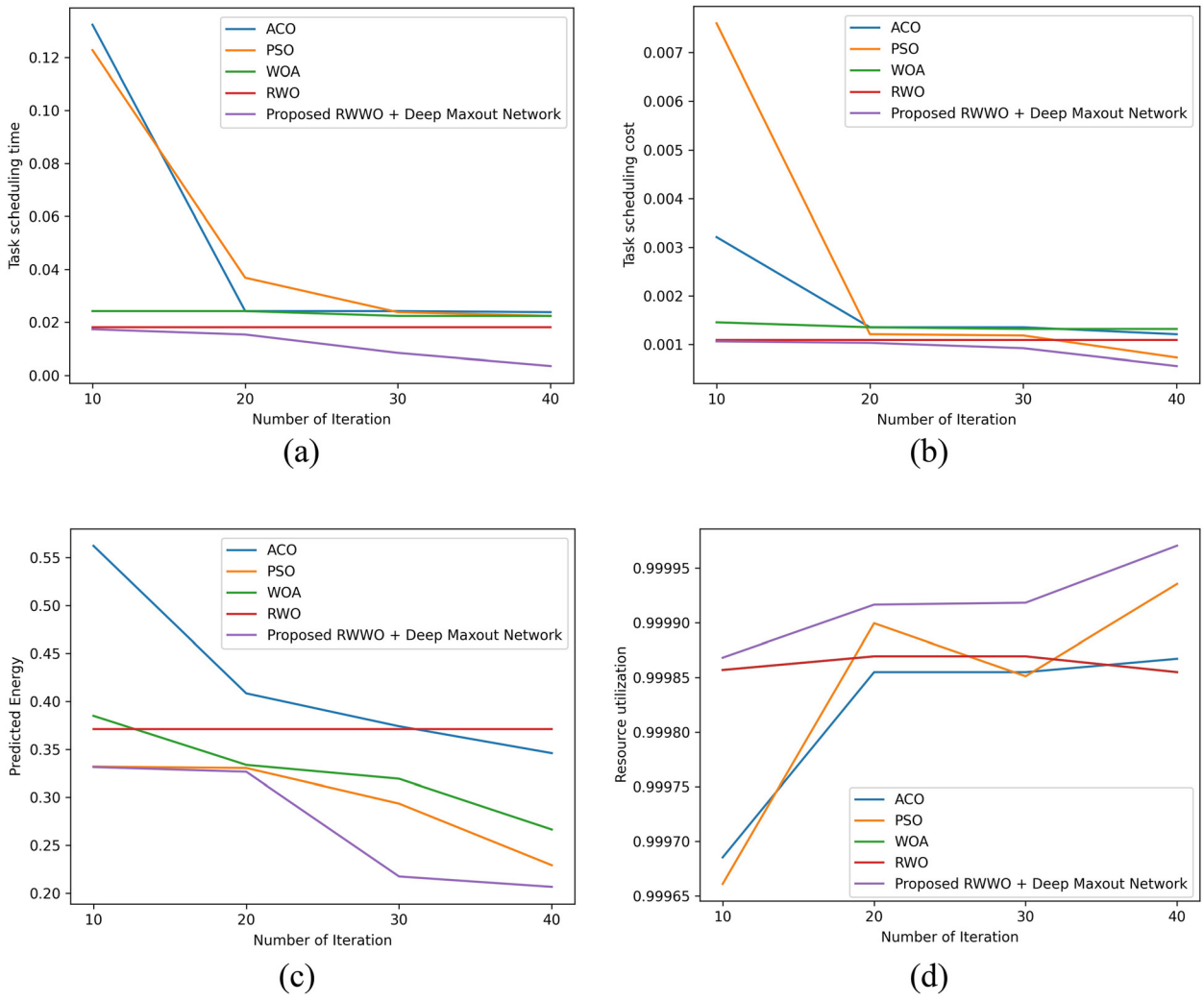

Figure 6: Analysis using experimental setup-3, a) task scheduling time, b) task scheduling cost, c) resource utilization, d) predicted energy.

Figure 6b represents the analysis of task scheduling cost. If the number of iteration = 10, the cost achieved by the proposed RWWO + Deep Maxout Network is 0.001061. However, the conventional schemes achieved the task scheduling cost of 0.0032 for ACO, 0.0076 for PSO, 0.0015 for WOA, and 0.0011 for RWO. By varying the number of

Table 1: Comparative discussion.

Setup	Metrics/Methods	ACO	PSO	WOA	RWO	Proposed RWWO + Deep Maxout Network
Setup-1	*Task scheduling time*	0.5132	0.3237	0.2149	0.1217	**0.0208**
	Task scheduling cost	0.0082	0.0091	0.0098	0.0046	**0.0017**
	Predicted energy	0.4177	0.4725	0.3638	0.2859	**0.1971**
	Resource utilization	0.9973	0.9978	0.9979	0.9989	**0.9999**
Setup-2	*Task scheduling time*	0.2701	0.2583	0.1619	0.0661	**0.0218**
	Task scheduling cost	0.0111	0.0105	0.0121	0.0064	**0.0045**
	Predicted energy	0.3688	0.4795	0.4277	0.3215	**0.1776**
	Resource utilization	0.9981	0.9984	0.9983	0.9992	**0.9995**
Setup-3	*Task scheduling time*	0.0238	0.0224	0.0224	0.0182	**0.0035**
	Task scheduling cost	0.0012	0.0007	0.0013	0.0011	**0.00055**
	Predicted energy	0.3460	0.2292	0.2665	0.3710	**0.2065**
	Resource utilization	0.9999	0.9999	0.9999	0.9999	**0.9999**

iteration = 40, the scheduling cost obtained by the proposed RWWO + Deep Maxout Network, existing ACO, PSO, WOA, and RWO is 0.000557, 0.0012, 0.0007, 0.0013, and 0.0011, respectively.

The analysis of predicted energy concerning the number of iterations is illustrated in Figure 6c. If the number of iteration = 10, the proposed RWWO + Deep Maxout Network's predicted energy is 0.3314. However, the conventional schemes, like ACO, PSO, WOA, and RWO attained the predicted energy of 0.5623, 0.3319, 0.3849, and 0.3710. Similarly, if the number of iteration = 40, the proposed RWWO + Deep Maxout Network's predicted energy is 0.2065, whereas the traditional approaches attained predicted energy of 0.3460 for ACO, 0.2292 for PSO, 0.2665 for WOA, and 0.3710 for RWO.

Figure 6d represents the analysis of resource utilization. If the number of iteration = 50, the proposed RWWO + Deep Maxout Network resource is 0.9998, whereas 0.99997 for iteration = 40. However, the existing schemed consumed the resource utilization of 0.9997 for ACO, 0.9997 for PSO, 0.9999 for WOA, and 0.9999 for RWO for the number of iteration = 10.

Comparative discussion

Table 1 portrays a comparative discussion of the developed scheme. There are three experimental setups, such as setup-1, setup-2, and setup-3. The task scheduling time for the proposed RWWO + Deep Maxout Network is 0.0208; task scheduling cost is 0.0017, predicted energy is 0.1971, and resource utilization is 0.9999. The experimental results reveal that the task scheduling time, cost, and predicted energy is minimum, whereas the resource utilization is maximized.

Conclusions

Cloud computing has tremendously increased its growth, and this technology has a rapid evolution over the past few years because of its simplicity and easy way of accessing resources. This technology plays a vital role in many applications, like software, computing, storage, network, and various heterogeneous needs. Task scheduling is an essential mechanism in cloud computing in allocating the appropriate resources to the users. However, there is a shortage of resources in serving the users when multiple users request similar resources simultaneously. Therefore, it is necessary to provide cost-effective executions with proper scheduling of tasks. Hence, this research presents an effective mechanism for task scheduling using the RWWO algorithm. The proposed approach provides hassle-free executions in the task scheduling mechanism by allocating suitable resources to the machines according to the user requirements. The fitness parameters utilized in this mechanism are QoS, resource utilization, and predicted energy. However, the predicted energy is determined using Deep Maxout Network. Moreover, the proposed RWWO + Deep Maxout Network achieved a minimum task scheduling time of 0.0208, minimum task scheduling cost of 0.0017, minimum predicted energy of 0.1971, and maximum resource utilization of 0.9999. There is still room for enhancement in enhancing the performance of the workflow scheduling mechanism by adding some other optimization algorithms. The implications of the proposed method is efficient for the scheduling the work flow for the selection of appropriate resource. In the future, the new optimization technique based on deep learning will be implemented for the efficient allocation of resource based on the workflow.

Author contributions: All the authors have accepted responsibility for the entire content of this submitted manuscript and approved submission.
Research funding: None declared.
Conflict of interest statement: The authors declare no conflicts of interest regarding this article.

References

Abd Elaziz, M., S. Xiong, K. P. N. Jayasena, and L. Li. 2019. "Task Scheduling in Cloud Computing Based on Hybrid Moth Search Algorithm and Differential Evolution." *Knowledge-Based Systems* 169: 39–52.

Al-Maytami, B. A., P. Fan, A. Hussain, T. Baker, and P. Liatsis. 2019. "A Task Scheduling Algorithm with Improved Makespan Based on Prediction of Tasks Computation Time Algorithm for Cloud Computing." *IEEE Access* 7: 160916–26.

Al-Turjman, F., M. Z. Hasan, and H. Al-Rizzo. 2019. "Task Scheduling in Cloud-Based Survivability Applications Using Swarm Optimization in IoT." *Transactions on Emerging Telecommunications Technologies* 30 (8): e3539.

Beegom, A. A., and M. S. Rajasree. 2014. "A Particle Swarm Optimization Based Pareto Optimal Task Scheduling in Cloud Computing." In *International Conference in Swarm Intelligence*, 79–86. Cham: Springer.

Boveiri, H. R., R. Khayami, M. Elhoseny, and M. Gunasekaran. 2019. "An Efficient Swarm-Intelligence Approach for Task Scheduling in Cloud-Based Internet of Things Applications." *Journal of Ambient Intelligence and Humanized Computing* 10 (9): 3469–79.

Chen, X., L. Cheng, C. Liu, Q. Liu, J. Liu, Y. Mao, and J. Murphy. 2020. "A Woa-Based Optimization Approach for Task Scheduling in Cloud Computing Systems." *IEEE Systems Journal* 14 (3): 3117–28.

Ding, D., X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng. 2020. "Q-learning Based Dynamic Task Scheduling for Energy-Efficient Cloud Computing." *Future Generation Computer Systems* 108: 361–71.

Dillon, T., C. Wu, and E. Chang. 2010. "Cloud Computing: Issues and Challenges." In *24th IEEE International Conference on Advanced Information Networking and Applications*, 27–33.

Engle, R. F., and S. Manganelli. 2004. "CAViaR: Conditional Autoregressive Value at Risk by Regression Quantiles." *Journal of Business & Economic Statistics* 22 (4): 367–81.

Eskandar, H., A. Sadollah, A. Bahreininejad, and M. Hamdi. 2012. "Water Cycle Algorithm–A Novel Metaheuristic Optimization Method for Solving Constrained Engineering Optimization Problems." *Computers & Structures* 110: 151–66.

Hien Bui, T. T., M. Jambulingam, M. Amin, and N. T. Hung. 2021. "Impact of COVID-19 Pandemic on Franchise Performance from Franchisee Perspectives: The Role of Entrepreneurial Orientation, Market Orientation and Franchisor Support." *Journal of Sustainable Finance & Investment* 1–19. https://doi.org/10.1080/20430795.2021.1891787.

Hung, N. T., and J.-C. Chang. 2019. "Preliminary Investigation of the Current Situation and Influencing Factors of International Students in Taiwan under the Background of New Southbound Policy." *Taiwan Educational Review* 2 (2).

Hung, N. T. 2020. "Deciphering the Increased Popularity of Vietnamese Students' Choice of Asian Countries for Overseas Studies: The Influence of Motivation for Studying Abroad on Career Planning and Decision-Making Process of Vietnamese Students in Taiwan." In *Proceeding of the International Conference on Higher Education in Vietnam and Asia: Similarities and Possibilities of Cooperation (IHESP)*.

Jana, B., Chakraborty, M., and Mandal, T. 2019. "A Task Scheduling Technique Based on Particle Swarm Optimization Algorithm in Cloud Environment." In *Soft Computing: Theories and Applications*, 525–36. Singapore: Springer.

Juarez, F., J. Ejarque, and R. M. Badia. 2018. "Dynamic Energy-Aware Scheduling for Parallel Task-Based Application in Cloud Computing." *Future Generation Computer Systems* 78: 257–71.

Karunakaran, V. 2019. "A Stochastic Development of Cloud Computing Based Task Scheduling Algorithm." *Journal of Soft Computing Paradigm (JSCP)* 1 (01): 41–8.

Kumar, A. S., and M. Venkatesan. 2019. "Task Scheduling in a Cloud Computing Environment Using HGPSO Algorithm." *Cluster Computing* 22 (1): 2179–85.

Mansouri, N., B. M. H. Zade, and M. M. Javidi. 2019. "Hybrid Task Scheduling Strategy for Cloud Computing by Modified Particle Swarm Optimization and Fuzzy Theory." *Computers & Industrial Engineering* 130: 597–633.

Masdari, M., F. Salehi, M. Jalali, and M. Bidaki. 2017. "A Survey of PSO-Based Scheduling Algorithms in Cloud Computing." *Journal of Network and Systems Management* 25 (1): 122–58.

Michael Mahesh, K. 2020. "Workflow Scheduling Using Improved Moth Swarm Optimization Algorithm in Cloud Computing." *Multimedia Research* 3 (3), https://doi.org/10.46253/j.mr.v3i3.a5.

Narendrababu Reddy, G., and S. Phani Kumar. 2019. "Regressive Whale Optimization for Workflow Scheduling in Cloud Computing." *International Journal of Computational Intelligence and Applications* 18 (4): 1950024.

Netaji, V. K., and G. P. Bhole. 2020. "Optimal Container Resource Allocation Using Hybrid SA-MFO Algorithm in Cloud Architecture." *Multimedia Research* 3 (1): 11–20.

Panda, S. K., and Jana, P. K. 2019. "An Energy-Efficient Task Scheduling Algorithm for Heterogeneous Cloud Computing Systems." *Cluster Computing* 22 (2): 509–27.

Reddy Bojja, G., M. Ofori, J. Liu, and L. Sai Ambati. 2020. "Early Public Outlook on the Coronavirus Disease (COVID-19): A Social Media Study." In *Proceedings of Americas Conference on Information Systems (AMCIS) 2020*.

Rjoub, G., J. Bentahar, and O. A. Wahab. 2020. "BigTrustScheduling: Trust-Aware Big Data Task Scheduling Approach in Cloud Computing Environments." *Future Generation Computer Systems* 110: 1079–97.

Singh, S., and I. Chana. 2016. "A Survey on Resource Scheduling in Cloud Computing: Issues and Challenges." *Journal of Grid Computing* 14 (2): 217–64.

Stephanakis, I. M., I. P. Chochliouros, G. Caridakis, and S. Kollias. 2013. "A Particle Swarm Optimization (PSO) Model for Scheduling Nonlinear Multimedia Services in Multicommodity Fat-Tree Cloud Networks." In *International Conference on Engineering Applications of Neural Networks*, 257–68. Berlin, Heidelberg: Springer.

Sun, W., F. Su, and L. Wang. 2018. "Improving Deep Neural Networks with Multi-Layer Maxout Networks and a Novel Initialization Method." *Neurocomputing* 278: 34–40.

Tsai, C. F., W. C. Lin, and S. W. Ke. 2016. "Big Data Mining with Parallel Computing: A Comparison of Distributed and MapReduce Methodologies." *Journal of Systems and Software* 122: 83–92.

Tsai, C. W., W. C. Huang, M. H. Chiang, M. C. Chiang, and C. S. Yang. 2014. "A Hyper-Heuristic Scheduling Algorithm for Cloud." *IEEE Transactions on Cloud Computing* 2 (2): 236–50.

Velliangiri, S., P. Karthikeyan, V. A. Xavier, and D. Baswaraj. 2021. "Hybrid Electro Search with Genetic Algorithm for Task Scheduling in Cloud Computing." *Ain Shams Engineering Journal* 12 (1): 631–9.