Deep Learning Approach for Workload Prediction and Balancing in Cloud Computing

Syed Karimunnisa[1], Yellamma Pachipala[2]
Research Scholar, Department of Computer Science and Engineering,
Koneru Lakshmaiah Education Foundation, Vaddesvaram, Guntur, AP, India-522503[1]
Associate Professor, Department of Computer Science and Engineering,
Koneru Lakshmaiah Education Foundation, Vaddesvaram, Guntur, AP, India-522503[2]

***Abstract*—Cloud Computing voted as one of the most revolutionized technologies serving huge user demand engrosses a prominent place in research. Despite several parameters that influence the cloud performance, factors like Workload prediction and scheduling are triggering challenges for researchers in leveraging the system proficiency. Contributions by practitioners given workload prophesy left scope for further enhancement in terms of makespan, migration efficiency, and cost. Anticipating the future workload in due to avoid unfair allocation of cloud resources is a crucial aspect of efficient resource allocation. Our work aims to address this gap and improve efficiency by proposing a Deep Max-out prediction model, which predicts the future workload and facilitates workload balancing paving the path for enhanced scheduling with a hybrid Tasmanian Devil-assisted Bald Eagle Search (TES) optimization algorithm. The results evaluated proved that the TES scored efficiency in makespan with 16.342%, and migration efficiency of 14.75% over existing approaches like WACO, MPSO, and DBOA (Weighted Ant Colony Optimization Modified Particle Swarm Optimization, Discrete Butterfly Optimization Algorithm). Similarly, the error analysis during the evaluation of prediction performance has been figured out using different approaches like MSE, RMSE, MAE, and MSLE, among which our proposed method overwhelms with less error than the traditional methods.**

***Keywords*—Task scheduling; virtual machines; optimization; workload prediction; migration; QoS**

Nomenclature

Abbreviation	Description
SIN	Service Invocation Number
DBN	Deep Belief Network
LSTM	Long Short-Term Memory
SVM	Support vector machine
MA	Moving Average
ARIMA	Autoregressive Integrated Moving Average
WACO	Weighted Ant Colony Optimization
DBOA	Discrete Butterfly Optimization Algorithm
Grid-LSTM	Grid-Long Short-Term Memory
Bi-LSTM	Bi-directional Long Short-Term Memory
SDWF	Self-Directed Workload Forecasting
JEES	Joint Energy Efficiency Optimization Scheme
SE	Sum Squared Error
CE	Cross Entropy Error
TDO	Tasmanian Devil Optimization
CSO	Cat Swarm Optimization
MFO	Moth Flame Optimization
BES	Bald Eagle Search
HWOA	Hybrid Whale Optimizer
MPSO	Modified Particle Swarm Optimization

I. Introduction

With the pervasive expansion of Internet access and the rise of Big Data, cloud computing has gained increasing prominence in today's business landscape [1]. In comparison to alternative distributed computing methodologies such as cluster and grid computing, cloud computing offers an adaptive and scalable approach to providing customized services to consumers. It provides a means for consumers to access computing resources and platforms without the necessity of owning the underlying technology, enabling them to utilize these resources in a pay-per-use manner. Numerous resources, including processing power, storage capacity, and network bandwidth, are easily accessible in the field of cloud computing. The complexity lies in distributing them fairly across different users and jobs to meet a range of demands and priorities. Therefore, allocating resources in this dynamic and diverse environment presents a challenging task for researchers.

The main challenge in cloud computing is its diversified fluctuation of workloads and user needs [2]. Task requirements differ in kinds and amounts of resources, with dynamically evolving user needs. Secondly, cloud resources are limited, therefore it is crucial to allocate them wisely aiming for maximum performance and service effectiveness. It becomes imperative to load balance to avoid performance drops by resource saturation, due to resource conflicts within the system. At times several users or jobs may inevitably compete for the same resources simultaneously, leading to resource conflicts and delays. Eventually minimizing disputes and guaranteeing even distribution and efficient resource utilization, for strong resource allocation and scheduling is an urged need.

To address issues pertaining to the performance of task scheduling [3-5], researchers have presented a variety of innovative approaches. The following categories generally describe the available task-scheduling techniques in a cloud environment. Static Scheduling Methods: These include algorithms such as Shortest Job First (SJF), Earliest Deadline First (EDF), and Minimum Remaining Time (MRT) to

determine the sequence of work allocation and execution before a job is submitted. They are easy to deploy, but they are not flexible enough to adjust to changing task needs and dynamic situations. Heuristic Scheduling Methods: These techniques, which include Genetic Algorithms (GA) and Particle Swarm Optimization (PSO), mainly rely on prior knowledge and pre-established guidelines for scheduling decisions. They cannot optimize globally or make adjustments in real time, even though they take job priorities and resource efficiency into account.

Methods for Load Balancing Scheduling: These techniques seek to balance workloads among computing resources resulting in improved performance and resource usage. Random scheduling, round-robin (RR) scheduling, and queue-length-based scheduling are a few examples. They perform well in workload distribution overriding job specifications.

Scheduling process with evolutionary demands questioning the efficiency of cloud performance [6]. First of all, the sheer number of tasks demanded and their effective management raises the bar for computational demands adding complexity. Second, the cloud environment is dynamic, task arrivals, departures, and resource requirements are always changing. This dynamic nature renders the requirement for task scheduling algorithms that can be flexible and adaptive in real time to quickly adjust to changing demands [7]. Finally, to accomplish load balancing and maximize resource utilization, efficient task scheduling that relies on the appropriate distribution and application of resources is needed. If this equilibrium is not reached, system performance may suffer and resources may be wasted.

At the core of cloud computing lies the allocation of computing tasks to a shared resource pool of resources comprising diverse virtualized servers or virtual machines (VMs) [8]. Operating akin to a market-driven utility, cloud computing endeavors to enable providers and users to optimize their profits with enhanced returns on investment. As a result, the adoption of sophisticated scheduling strategies becomes essential to facilitate the management of software, user applications, tasks, and workflows within this environment. Scheduling, in its essence, plays a crucial role in shaping system performance, influencing both resource utilization efficiency and operational costs, thus underscoring its mark in the domain of cloud computing [9].

Due to the dynamic provisioning and management capabilities of virtual machines (VMs) [10], challenges in cloud scheduling generally manifest in two layers. Firstly, the task scheduling phase involves aligning user-submitted tasks concerning available VM resources. Secondly, a vital VM-to-host mapping process, which facilitates VM creation or migration [11, 12, 13]. Our main emphasis is on optimizing the former, as it directly influences the processing capabilities of a cloud computing system. Improving task scheduling has the potential to notably enhance system efficiency in terms of both time and cost [14, 15].

The above-mentioned challenges and issues that impact cloud performance are considered and addressed by introducing a framework that encompasses operations like workload prediction and scheduling, resulting in the design of a deep learning algorithm trained on features such as VM capacity and task capacity to optimize the scheduling process. The contributions of this work are delineated as follows:

- Introducing an enhanced Deep Learning approach, named Improved Deep Maxout, to predict workloads by training on both VM and task capacities.
- The prediction process facilitates optimal task scheduling through the TES algorithm, guaranteeing the achievement of objectives such as time efficiency, cost-effectiveness, and overall system efficiency.

The paper begins by introducing the concepts and challenges associated with cloud computing in Section I. Section II presents a thorough literature review along with the analysis and discussions of researchers' findings. In Section III, the paper describes the architecture and system flow of the proposed methodology in detail. Results and discussion is given in Section IV. The conclusion and future directions for enhancements are discussed in Section VI, shedding light on potential future developments.

II. Related Work

Several Researchers have made significant contributions to resolving task scheduling and resource allocation challenges. However, our work builds upon these efforts by addressing overlooked aspects and introducing enhancements that improve overall performance.

Wiem Matoussi and Tarek Hamrouni [16] developed workload forecasting techniques in 2021 to support capacity planning, guarantee effective resource allocation, and uphold SLA agreements with end users. Their methodology offered a novel way to forecast the surge of requests to a SaaS service and distribute virtual resources to satisfy user needs. The dual goals of this technique were response time optimization and accurate forecast forecasts.

Vinícius Meyer et al. [17] presented a machine learning-based classification technique in 2021 to provide the best possible resource allocation in cloud environments that are aware of dynamic interference. The main objective was to demonstrate how categorization techniques affect resource allocation so that it better accommodates variations in workload. The study began by looking at how different apps with different dynamic requests are handled by hardware components. The effectiveness of several interference classification techniques was investigated and assessed, taking into account the dynamic nature of cloud workloads.

Marek Grzegorowski et al. [18] presented a revolutionary method in 2020 for building a reliable clustering methodology with cloud resources customized to the particular data processing requirement. The provided architecture made use of the infrastructure-as-a-code framework to allow for dynamic cluster configuration and administration. It begins by figuring out which cluster size is best for finishing a task within the given time frame. The execution time was then optimized by using ARIMA models and examining the price history of spot instances to benefit from the lower prices offered by the cloud spot market.

Min Cao et al. [19] introduced three fresh methods for an energy-aware EIS in 2023. The author initially determined the best time for each task to run on a certain resource to save energy. Second, to reduce frequency factor and voltage and conserve energy, the EIS allows workflow slack time according to the optimal time for each task's execution. Moreover, the EIS minimizes dynamic energy consumption and satisfies workflow deadline limitations by utilizing the lapsed time caused by task priority deficiencies.

A logarithmic method was used in 2021 by Jing Bi et al. [20] to lower standard deviation before workloads and resource sequences were applied. They used the Min-Max approach to scale the data and an improved filter to remove noise interference and outliers. They have created an integrated deep learning method for time series forecasting, which makes use of ML-network models like Grid-LSTM and Bi-LSTM networks to produce accurate forecasts of resource demands and workload arrival at regular intervals.

The SDWF (Standard Deviation Weighted Forecasting) approach was developed in 2021 by Jitendra Kumar et al. [21]. It uses the deviation in the present predictions to calculate the trend of forecasting errors and improve the accuracy of future predictions. For training neurons, the model uses a sophisticated heuristic approach motivated by the black hole phenomenon. In addition, a statistical analysis was carried out to verify the accuracy of the suggested forecasting model, using Friedman and Wilcoxon signed ranking tests.

JEES (Joint Energy Optimisation and Scheduling), which simultaneously tackles and optimizes energy consumption in both cooling systems and servers, was introduced in 2021 by Kaixuan Ji et al. [22]. In addition to a resource management strategy that integrates workload forecasting models for resource allocation and a task-migration approach that makes use of marginal cost evaluation, JEES also includes a dynamic online task-scheduling technique based on the evaluation of marginal cost. The combined effect of these strategies is to lower data centers' overall energy usage.

Taking into account the unpredictable and time-varying nature of the workload, Zheng Xiao et al. [23] combined VM allocation with task scheduling in 2019. A Markov chain can be used to simulate the Markov property that the acquired workload dataset exhibits, as the study showed. In addition, recurrence, entropy, and persistence were found to be the three main operators that best described the workload. These operators assess the stability, predictability, and approximate burst timings of user requests, in that order. It was discovered that there is a nonlinear link between workload characteristic operators and virtual machine allocation.

A hybrid weighted Ant Colony Optimisation model with solution and pheromone updating functions was presented by Chirag Chandrasekar et al. [24] in 2023 for the best job scheduling. They showed that their meta-heuristic method outperformed current algorithms in terms of metrics like execution time and resource management.

Neetu Sharma, Sonal, and Puneet Gala [25] introduced a QoS-based Ant Colony Optimisation scheduling system in 2020 that used a neural network technique for effective multi-objective scheduling [37]. Their suggested approach outperformed current algorithms in terms of user task scheduling costs and execution times.

A modified Particle Swarm Optimisation (MPSO) was suggested in 2023 by Shikha Chaudhary et al. [26] to overcome the issues of long scheduling times and high computational costs during the scheduling process. By minimizing early convergence and improving local search efficiency, the MPSO optimizes the objective function about cost and makespan. An increase in the graph indicates that the performance of the suggested model is superior to that of conventional methods.

To solve resource waste and improve the algorithm's speed of convergence, Medhi Hussein Zadeh et al. [27] created a discrete Butterfly Optimisation Algorithm (DBOA) in 2021 that was modeled after the Levy flight technique. Their method of task prioritization and DBOA successfully addresses local optima concerns and allows for more efficient scheduling of intense workflows.

A task scheduling technique using Cat Swarm Optimisation was introduced in 2023 by Sudheer Mangalampalli et al. [28]. This algorithm prioritizes tasks and arranges them for scheduling. Inspired by the behavior of cats, this algorithm outperforms baseline methods that are currently in place in terms of QoS metrics like makespan and resource utilization.

Abdul Rajak [36] addressed an intelligent approach that benefits the real-world agricultural environment. Md shohel Sayeed et.al [38] proposed a real-time system for parking vehicles using a weighted K-nearest neighbour approach by selecting an optimal scheduler. Gousteris et.at [39] have come up with a secure approach for cloud storage using blockchain technology for efficient performance with heterogeneous input data.

III. METHODOLOGY

The work progresses by considering user requests as tasks appertaining to N users symbolized as U_i, where $i=\{1,2,...,N\}$. These tasks designated as Tsk_j with $j=\{1,2,.....M\}$ are assigned to feasible virtual machines VM_i where $i=\{vm_1,vm_2,.....vm_n\}$ and physical machines PM_i where $i=\{pm_1,pm_2,.....pm_k\}$. The proposed paradigm proceeds based on a priority scheme for scheduling using Deep learning techniques resulting in improved forecast of workloads.

Improving cloud performance and cutting operating costs need accurate workload forecasts. To precisely predict workloads, a deep learning model is used in this article, which increases efficiency and lowers costs. The proposed improved Deep Maxout model successfully learns to anticipate jobs as target labels by using factors like CPU utilization, day of the week, and time of day.

The enhanced Deep Maxout model's mathematical formulation is explained in [29]. To improve model robustness, it adds the modified softmax activation function (G-SM) to Eq. (2). In this case, the activation value is denoted by si and $gd(s)$ Gaussian distributed term with mean μ and standard deviation σ. The classic softmax activation function, which can handle multiple classes by normalizing the outputs for each class, ranging from 0 to 1, is presented by Eq. (1).

The input layer, embedding layer, max pooling layer, dropout layer, convolution layer, and dense layers that make use of activation and maxout functions constitute the Deep Maxout model's network structure. The maxout unit of this network is shown in Eq. (3), here $K_{yz} = N \cdot \lambda_{yz} + \gamma_{yz}$, γ serves as the bias, with N standing for input features such as task and virtual machine capacity, η serving as the feature map, and λ serving as the weight.

The softmax function *SM* is coined as follows:

$$SM(s)_i = \frac{e^{s_i}}{\sum_{j=1}^{k} e^{s_j}} \quad (1)$$

This formula defines a probability distribution across *k* possible outcomes. Here s_i represents the input value corresponding to outcome *I* and the denominator summates the exponentials of all input values across all outcomes.

$$G - SM = \frac{exp(s_i + gd(s))}{\sum_{j=1} exp(s_i + gd(x))} \quad (2)$$

$$gd(s) = 0.5 * er\_f\left(-\frac{\sqrt{2}(\mu_i - s_i)}{2\sigma_i}\right) + 0.5$$

where, error function er_f can be coined as.

$$(1/\sqrt{\pi}) \int_{-z}^{z} e^{-t^2}\, dt, t^2 = s_i$$

$$Q(M) = \underset{z\in[1,\eta]}{Maximum}(R_{yz}) \quad (3)$$

The input layer receives the input feature *N*, and the embedding layer processes its output to calculate the outcome. The dropout layer and convolution layer are the next two layers that process the output further. The final output is obtained starting with the third convolution layer. The final output is then produced by the max-pooling layer following the convolution layer. The dense layer is followed in order by the dropout layer.

The output of the max-out module is calculated based on the input pipelined from the dropout layer and is combined with the dense layer to create the final product. Using the output of the dense layer as the last step, the activation function computes the classification result as Deep Max_{out}. To evaluate the accuracy parameter of the model, hybrid loss functions which are given in Eq. (4) and detailed in [33] are used. Typically, error measures such as the *SE(E_SE)* and *CE* loss functions *(E_CE)* are employed to assess the efficacy of a classification model.

The enhanced hybrid loss function calculation for the proposed model is provided in Eq. (5), where the dynamically weighted balanced *CE* is represented by $E_{\_CE^*}$ (as in Eq. (6)). In this case, $y_i\hat{}$ stands for the anticipated label, and y_i indicates the actual label. w_lb_j, which is calculated as the ratio of class frequency n_j and the majority class (as determined across the training dataset), is equal to the class frequency log (as in Eq. (7)). The proportions allocated to the loss functions are represented by the scalar values Lf_1 and Lf_2, where $Lf_1+Lf_2=1$.

$$E_{\_he} = Lf_1 \frac{E\_SE}{Max_{SE}} + Lf_2 \frac{E\_CE}{Max_{CE}} \quad (4)$$

Here

$$E_{\_CE} = -\frac{1}{Output\_Size} \sum_{i=1}^{Output\_Siz} y_i \cdot log\, \hat{y_i} + (1 - y_i) * 1 * log\left(1 - \hat{y_i}\right)$$

$$E_{\_he} = Lf_1 \frac{E\_SE}{Max_{SE}} + Lf_2 \frac{E\_CE^*}{Max_{CE^*}} \quad (5)$$

$$E_{\_CE^*} = -\frac{1}{Output\_Size} \sum_{i=1}^{Output\_Siz} \sum_{j=1}^{c} w\_lb_j^{(1-p_{ij})}\, y_i \cdot log\, \hat{y_i} + (1 - y_i) * 1 * log\left(1 - \hat{y_i}\right) \quad (6)$$

$$w\_lb_j = log\left(\frac{max(n_j | j \in c)}{n_j}\right) + 1 \quad (7)$$

1) Optimal load balancing and task scheduling: An effective load-balancing approach proactively aids in monitoring the workload of the virtual machines (VMs') and allocating jobs to them appropriately. An example job would be Tsk_j, which has a range of 1000 tasks. The physical machines would be pm_k, with 50 machines, and the virtual machines would be vm_N, with 20–25 computers. There are multiple vm_N within this range for every pm_k. The pm_k machines are randomly assigned tasks.

The enhanced Deep Maxout model is used in this workload prediction model. The target label of the Deep Maxout model is set to 0, 1, and 2, and the features supplied to the Improved Deep Maxout model are Task capacity and VM capacity. The following are the definitions of under load, equal load, and overload conditions:

- The target label is set to 0, indicating that the machine's workload prediction is under load, if the task capacity is smaller than the virtual machine's capacity.
- The target label is set to 1, indicating that the machine's workload prediction is at equal load, if the task capacity and the VM capacity are equal.
- The target label is set to 2, meaning that the machine's workload prediction is overloaded, if the task capacity exceeds the virtual machine capacity.

Constraints including Makespan (Fn_1), Migration cost (Fn_2), and Migration efficiency (Fn_3) are taken into account throughout the scheduling process considered. Fig. 1 shows the model of scheduling. The input solution assigns a lower bound of Lb=0, *(Zeros(len(Machine_underload)))*, and an upper bound of Ub=1, *(Ones(len(Machine_underload)))* to the variables. The number of underloaded machines is equal to the problem size of TES.

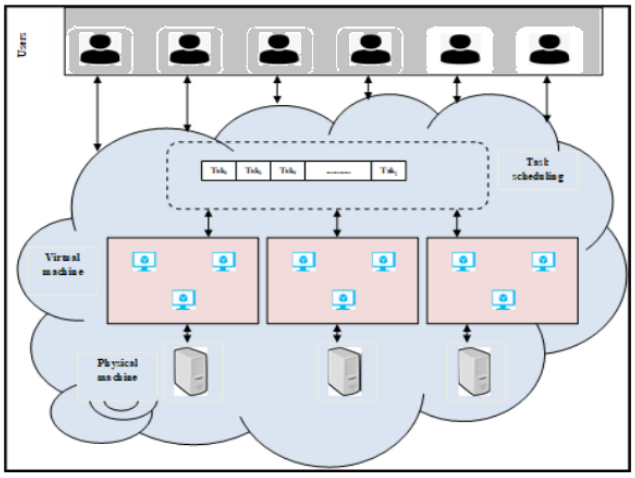

Fig. 1. Scheduling of task in cloud computing.

The objective function is expressed in Eq. (8), where random weights in the interval [0, 1] are represented by the variables w_1, w_2, and w_3.

$$Obj = (w_1 * Fn_1) + (w_2 * Fn_2) + (w_3 * Fn_3) \quad (8)$$

1) Makespan(Fn_1): Eq. (9), which defines the makespan, shows the total computing time *(CT_i)* (as given in Eq. (10)) needed to complete a task, where N is the number of virtual machines.

$$Fn_1 = \underset{1 \le i \le N}{Max}\{CT_i\} \quad (9)$$

$$CT_i = \sum_{j=1}^{n} \frac{T_j * length}{vm_N * Pes\_Num * Xvm_N} \quad (10)$$

2) Migration cost(Fn_2) : A *M*M* matrix is used to calculate the migration cost, where the rows and columns of the matrix indicate the migration costs of virtual machines (vm_N) and physical machines (pm_K), respectively. The (1, 1) and (2, 2) columns in this matrix indicate reduced migration costs, while the remaining entries suggest higher migration costs.

3) Migration efficiency(Fn_3) : Based on the migration value, the migration efficiency is computed, as shown in Eq. (11).

$$Fn_3 = \frac{1}{Fn_1} \quad (11)$$

This section presents the mathematical formulation of the proposed meta-heuristic TES, which combines TD [30] with ES [31]. The Tasmanian devil feeds on live prey by attacking them or by scavenging carrion from dead animals. This behavior is simulated by TES. First, a random population of agents is generated according to the limitations of the challenge. Based on the location of their search region, the population members of TES, which are problem-solving search agents, suggest potential values for problem variables. Functionally, each member of the population can be thought of as a vector, with the number of elements representing the number of variables in the problem. A matrix in Eq. (12), where P is the Tasmanian population, N is the number of Tasmanian devils seeking, P_i is a potential solution, and m is the number of specified problem variables, can be used to simulate the set of TES members.

$$P = \begin{bmatrix} P_1 \\ : \\ P_i \\ : \\ P_N \end{bmatrix} = \begin{bmatrix} p_{1,1} \dots p_{1,j} \dots p_{1,m} \\ : \\ p_{i,1} \dots p_{i,j} \dots p_{i,m} \\ : \\ p_{N,1} \dots p_{N,j} \dots p_{N,m} \end{bmatrix}_{N \times m} \quad (12)$$

By changing the values of potential solutions into the defined objective function's objects, the objective function can be assessed. In Eq. (13) the values obtained for the defined objective function are represented by a vector V.

$$V = \begin{bmatrix} V_1 \\ : \\ V_i \\ : \\ V_N \end{bmatrix} = \begin{bmatrix} V(P_1) \\ : \\ V(P_i) \\ : \\ V(P_N) \end{bmatrix}_{N \times 1} \quad (13)$$

The Tasmanian devil algorithm does not always hunt; sometimes it prefers to eat the carrion that is around. The area above the Tasmanian devil is home to additional predators that hunt enormous prey but are unable to finish it. Each Tasmanian devil considers the position of another population member to be carrion bait under the TES design. Cr_i denotes the selected carrion in Eq. (14) which shows the random selection of one of these cases. The Tasmanian devil is moved to a new location inside the search area based on Cr_i.

$$Cr_i = P_k, i = 1,2, \dots, N,$$
$$x \in \{1,2, \dots, N | x \neq i\} \quad (14)$$

A random number is represented by $In \in (1,2)$ and a random number is represented by $r \in (0,1)$, while the Tasmanian devil's current update based on the first strategy is represented by $p_{i,j}^{new,S1}$ in Eq. (12). Eq. (15) is used to calculate the Tasmanian devil's new position. If the objective function value at the new position is higher than its previous position, the position is considered acceptable; otherwise, the Tasmanian devil process retains its position.

The new update for the Tasmanian devil is obtained by merging the TD and proposed ES updates in Eq. (16), by the proposed methodology. In this case, the random factor is represented by *RandF* (as in Eq. (18)) [34], the random value is $r_2 \in (0,1)$ the maximum iteration is It_{Max} , the current iteration is C_It and the levy flight update is represented by *Levy_fun* (as in Eq. (19)). Eq. (20) provides the typical (eagle search) ES update equation, where α is the parameter governing the position change.

Update to the new value as $p_{i,j}^{new,S1}$ with $p_{i,j} + r \times \left(Cr_{i,j} - In \times p_{i,j}\right)$ when the criteria $V_{Cr_i} \prec V_i$ is met alternatively

$$p_{i,j} + r \times \left(p_{i,j} - Cr_{i,j}\right) \quad (15)$$

The algorithm considers $P_{i,j}^{new,S1}$the updated if the fitness offered is better than current else retains the current value as follows

$$P_i = \begin{cases} P_i^{new,S1}; V_i^{new,S1} \prec V_i \\ P_i; Otherwise \end{cases} \quad (16)$$

$$p_{i,j}^{new,S1} = \begin{cases} p_{i,j} + RanF \times \left(Cr_{i,j} - In \times p\right); \\ if V_{Cr_i} \prec V_i \\ p_{best} + \alpha * r(p_{mean} - p_i) * Levy\_fun; \\ else \end{cases} \quad (17)$$

$$RanF = r_1 * sin(r_2), \quad (18)$$

where

$$r_1 = \frac{1.5 \times (It_{Max} - t + 1)}{It_{Max}}$$

$$Levy_{fun} = \frac{C_{It(1+\beta)} * \left(sin\left(\frac{\pi\beta}{2}\right)\right)^{1/\beta}}{C_{It\left(\frac{1+\beta}{2}\right)} * \beta * \left(2^{\left(\frac{\beta-1}{2}\right)}\right)} \quad (19)$$

$$p_{new} = p_{best} + \alpha * r(p_{mean} - p_i) \quad (20)$$

$$Pry_i = P_k, i = 1,2, \ldots, N,$$

$$x \in \{1,2, \ldots, N | x \neq i\} \quad (21)$$

The position of other population members is taken into account as the location of prey during the updating procedure of the i^{th} Tasmanian devil. Prey selection is modeled by Eq. (21) where Pry_i indicates the selected prey and $x \in (1, N)$ is a natural random number.

Eq. (22) uses the exact location of the prey to calculate the Tasmanian devil's new position. The Tasmanian devil's location is adjusted to this new position if the new location increases the target function value. Eq. (23) provides an example of this second strategy phase.

$$p_{i,j}^{new,S2} = \begin{cases} p_{i,j} + r \times \left(Pry_{i,j} - In \times p_{i,j}\right); \\ if V_{Pry_i} \prec V_i \\ p_{i,j} + r \times \left(p_{i,j} - Pry_{i,j}\right); \\ else \end{cases} \quad (22)$$

$$P_i = \begin{cases} P_i^{new,S2}; V_i^{new,S2} \prec V_i \\ P_i; Otherwise \end{cases} \quad (23)$$

The radius *R* of the neighborhood that is, the region where the Tasmanian devil tracks its prey can be found using Eq. (24). Thus, a new position for the Tasmanian devils can be established by quantitatively simulating their pursuit behavior using Eq. (25). The Tasmanian devil will accept the newly calculated position if it provides a better value for the goal function than the previous position. Eq. (26) describes the process of updating the position of the Tasmanian devil. The Tasmanian devil's position update procedure is carried out, by the model presented in Eq. (27), by incorporating the factor P_Fact as suggested in Eq. (28) [32].

$$Rd = 0.01\left(1 - \frac{t}{It_{Max}}\right) \quad (24)$$

$$p_{i,j}^{new} = p_{i,j} + (2r - 1) \times Rd \times p_{i,j} \quad (25)$$

$$P_i = \begin{cases} P_i^{new}; if\ V_i^{new} \prec V_i \\ P_i; else \end{cases} \quad (26)$$

$$P_i = \begin{cases} P_i^{new} \times P\_Fact; \\ if\ V_i^{new} \prec V_i \\ P_i; else \end{cases} \quad (27)$$

$$P\_Fact = exp\left(\frac{-i}{\delta \times It_{Max}}\right) \quad (28)$$

Algorithm 1: TES (Tasmanian Devil-assisted Bald Eagle Search) for optimal load balancing and task scheduling.

Input: Set of VMs = {VM_1,VM_2….,VM_n} and Tasks={T_1,T_2….,T_n}
Initialize: No of iterations (*T*) and no of members of the population (*N*).
While t=1: T
For i=1: N
IF $prob \prec 1/2, prob = rand.$
Select carrion for the i th Tasmanian devil by Eq. (14)
Calculate the new status of the Tasmanian devil by Eq. (16)
Update the i th Tasmanian devil apply Levy flight strategy by Eq. (17)
Else
Select prey of i^{th} the Tasmanian devil using Eq. (21)
Assess the updated value of the Tasmanian Devil using Eq. (22).
The Tasmanian Devil's update is executed using Eq. (23).
Update Rd by Eq. (24)
Assess the new updated value of i^{th} Tasmanian Devil in the neighborhood using Eq. (25).
Update proposed i^{th} Tasmanian devil via Eq. (27)
End For
End While
Output: The best solution obtained for a given optimization problem.

IV. Results and Discussion

A. Experimental Setup and Simulation

We employed simulations to evaluate the effectiveness of our proposed scheduling methods. Cloudsim is widely utilized as simulation software for evaluating optimization techniques. It replicates components of cloud systems such as data centers, tasks, and virtual machines, while also supporting task scheduling strategies and diverse energy usage models for simulating various workloads. In this study, we simulated a cloud model based on a single data center, akin to Infrastructure as a Service (IaaS). The simulations were performed on a computer equipped with an Intel(R) Core(TM) i5-8265U CPU @ 1.80 GHz processor, 16 GB RAM, and a 64-bit Windows 11 Operating System.

The configuration of the simulated cloud data center is shown in Tables I, II and III.

TABLE I. CONFIGURATION OF HOST IN DATACENTER

Host Parameters	*Value*
Processing Element (PE)	2-10
Processing capacity	20000-35000 MIPS
RAM capacity	8GB,16GB,32GB

TABLE II. CONFIGURATION OF VMS

VM Parameters	*Value*
Processing Element (PE) in each VM	1
CPU computing capacity	600-4000 MIPS
RAM capacity	512-4196 MB

TABLE III. TASK PARAMETERS

Task Parameters	*Value*
Task Length	15000-900000 MI
Size of Task	60-3000 KB

B. Performance Metrics

The Google Cluster Workload Traces dataset from 2019 sourced from [35] is subjected to Cloudsim for workload prediction and job scheduling. The TES approach is evaluated against conventional strategies such as DBOA, MPSO, and WACO. The key parameters considered for evaluating the performance of the proposed method pertain to Communication cost, Execution time, Makespan, Migration Cost and Migration Efficiency are compared against existing approaches. The analysis is carried out with task counts ranging from 500 to 2000.

1) Communication cost and execution time: Fig. 2 and Fig. 3 depicts the comparison of TES performance to that of WACO, MPSO, and DBOA in terms of communication cost and execution time for workload prediction and task scheduling. The results achieved present an increased drift in performance by the proposed TES in terms of reduced execution times and communication costs over other approaches. To be more precise, when handling tasks with tasks of various counts, the TES attained notably lowered communication costs than other algorithms handling. The result of the proposed approach renders the utmost values when task count is 500, when compared against task counts of 1000, 1500, and 2000. Furthermore, the TES demonstrated a significant improvement with an execution time of 1648 seconds with 2000 tasks.

2) Makespan analysis: The workload prediction and task scheduling performance metrics comparison of the TES vs. WACO, MPSO, and DBOA is shown in Fig. 4. Indicating that when job/task count is set at 500, the findings show a shorter makespan for the TES approach as compared to conventional methods. This demonstrates how accurate the TES is at forecasting workload and allocating jobs facilitating the shortest makespan.

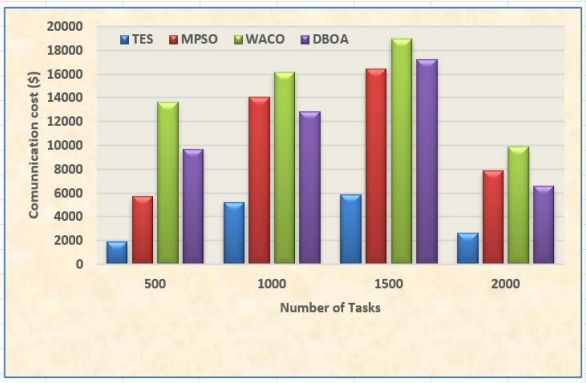

Fig. 2. Communication cost vs. number of task.

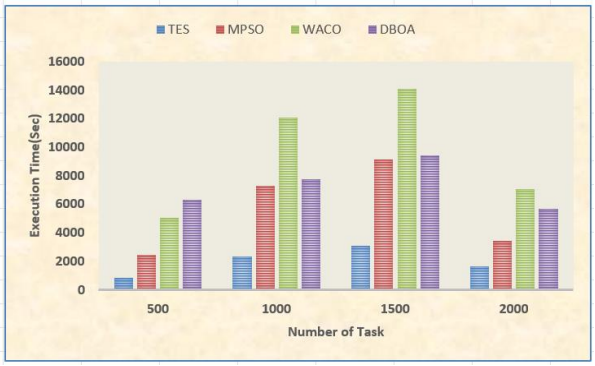

Fig. 3. Execution time vs. number of task.

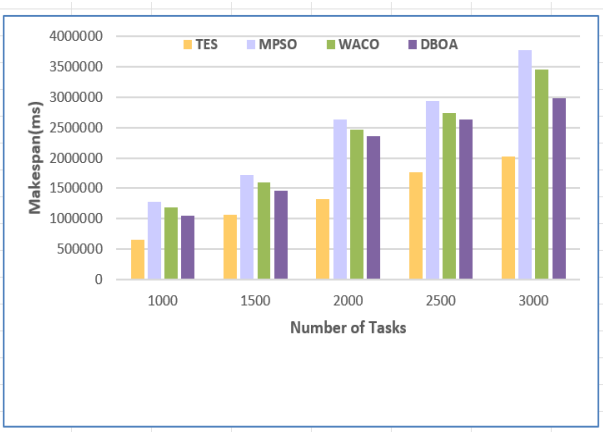

Fig. 4. Makespan vs. number of tasks.

3) Migration cost and migration efficiency analysis: The analysis of the TES's efficiency and migration cost of other task scheduling and workload prediction techniques is shown in Fig. 5 and Fig. 6. The number of tasks is changed to conduct the analysis. With a migration cost of 1.54 (with 2000 tasks), the TES methodology outperforms previous approaches with WACO=5.786, MPSO=4.345, and DBOA=3.987, respectively, in terms of efficiency and cost. Furthermore, TES's migration efficiency (0.0987) outperforms MPSO's (0.322), WACO's (0.378), and DBOA's (0.0239), with 2000 tasks. These findings show that TES is more effective and less expensive during task migrations.

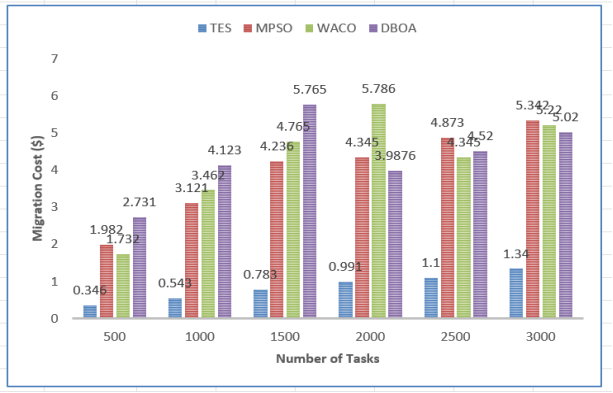

Fig. 5. Migration cost vs. number of tasks.

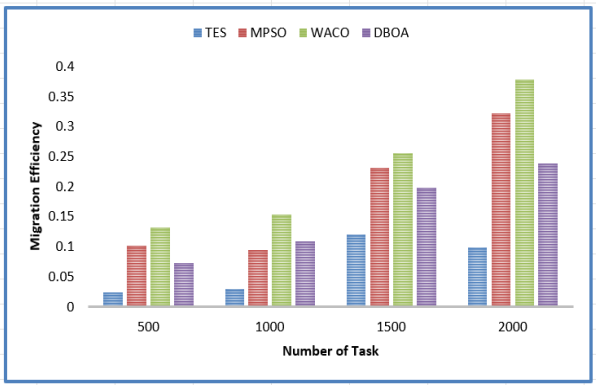

Fig. 6. Migration efficiency vs. number of task.

4) Convergence evaluation: As seen in Fig. 7, the convergence analysis of TES is contrasted with other methods currently in use, such as WACO, MPSO, and DBOA. The cost values that were initially produced by TES and other methods were high at the 0th iteration, with a considerable downtrend following successive iterations. The cost parameter values attained are comparatively low using TES, especially on the 25th iteration when it obtained a minimal cost value of 3.427. According to the convergence graph, TES converges more rapidly than other schemes, bringing down costs and producing enhanced accuracy in forecasting workload about user tasks.

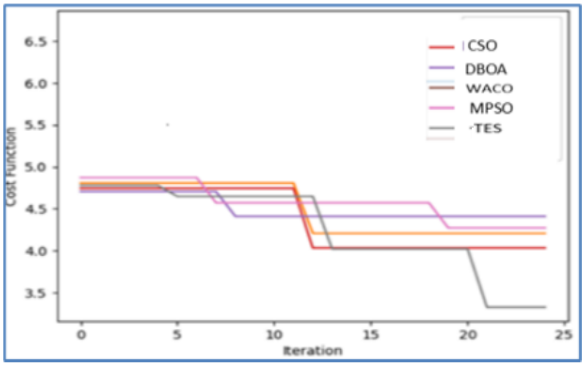

Fig. 7. Cost function vs. iterations.

5) Regression analysis: The regression evaluation of the suggested approach and the current approaches (CNN, NN, and RNN) for workload prediction is shown in Fig. 8, 9, 10 and Fig. 11. The results demonstrate that TES in contrast to CNN, NN, and RNN, which typically provide more divergent values, produces more consistent values in both real and classified labels.

6) Error analysis of improved deep maxout: Table IV presents an error analysis of the suggested algorithm concerning conventional Deep Maxout, CNN, RNN, NN, Bi-GRU, RMSE, MAE, and MSLE. The lowered error rates attained with the enhanced Deep Maxout technique reflect the e. In particular, the RMSE of the enhanced Deep Maxout is 0.100, which is substantially less than that of the Conventional Deep Maxout (1.150), RNN (0.340), CNN (0.260), and NN (0.160) in Bi-GRU (1.180), respectively. Furthermore, the suggested approach demonstrated enhanced performance in terms of MSE=0.120, MAE=0.021, and MSLE=0.215, all of which were minimized.

7) Prediction analysis: Table compares the prediction evaluation of TES with CNN, NN, and RNN addressing the prediction efficiency of TES that outperformed CNN, NN, and RNN algorithms in terms of prediction accuracy.

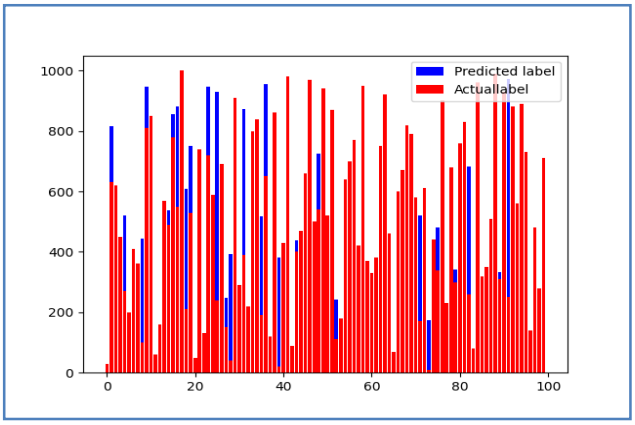

Fig. 8. Regression evaluation of CNN.

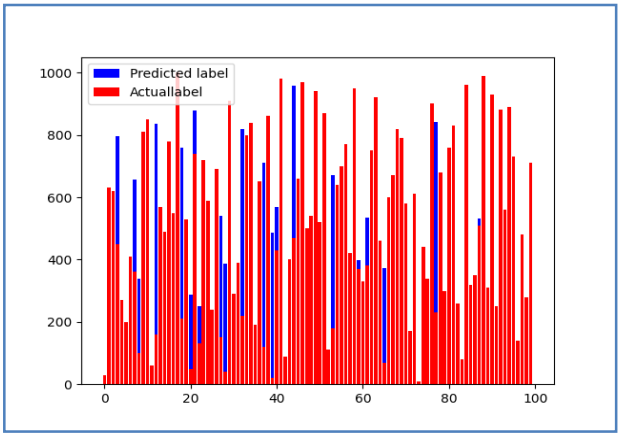

Fig. 9. Regression evaluation of NN.

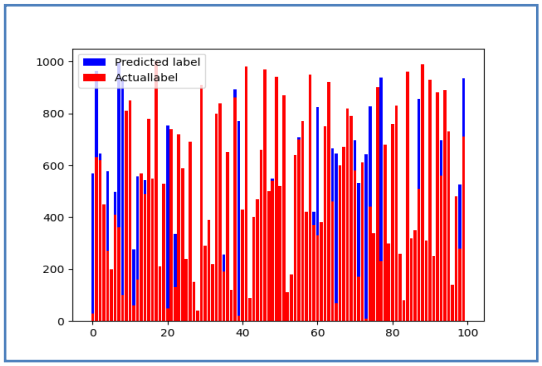

Fig. 10. Regression evaluation of RNN.

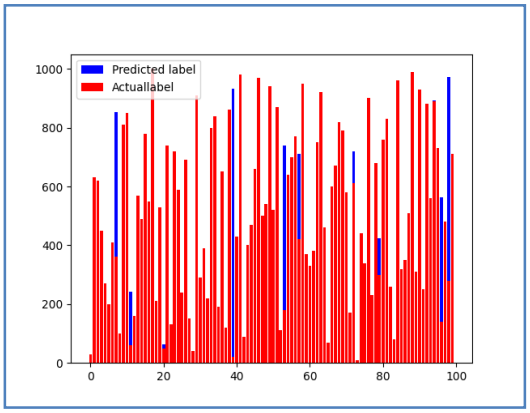

Fig. 11. Regression evaluation of TES.

TABLE IV. ERROR EVALUATION

Methods	*MSE*	*RMSE*	*MAE*	*MSLE*
CNN	0.521	0.260	0.150	0.520
RNN	0.460	0.340	0.140	0.678
NN	0.320	0.160	0.057	0.447
Bi-GRU	2.180	1.180	0.652	1.404
Conventional Deep Maxout	4.120	1.150	0.224	1.320
Improved Deep Maxout	0.120	0.100	0.021	0.215

TABLE V. PREDICTION EVALUATION OVER THE TRADITIONAL APPROACHES

Methods	*Success Rate*	*Over Prediction*	*Under Prediction*
CNN	49	51	49
NN	49	51	49
RNN	53	47	52
TES	96	14	25

V. CONCLUSION

The proposed study offers an effective workload prediction model that forecasts future workload based on trends seen in historical data by utilizing the Deep Max-out prediction model. This model uses the TES Algorithm to optimize scheduling while efficiently balancing the workload on machines. By ensuring that jobs are moved among virtual machines (VMs) in the best possible way, this method produces effective scheduling that takes into account metrics like make-span, migration cost, and migration efficiency. Comparing the suggested model to conventional approaches, promisingly results in high communication costs, with statistical analysis showcasing that the new model greatly reduces communication costs (2647.835). Moreover, the comparison of forecast outcomes emphasizes how crucial it is to take into account limitations such as job make-span, fitness, migration-cost, and execution time. When comparing the suggested model to conventional techniques, these constraints are noticeably less. In particular, the execution time is reduced to around 817, demonstrating how well the suggested model performs in comparison to traditional techniques that require larger execution cycles.

REFERENCES

[1] Fatemeh Ebadifard and SeyedMortezaBabamir, "Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment", Cluster Computing, 2020, https://doi.org/10.1007/s10586-020-03177-0.

[2] K. Lalitha Devi and S. Valli, "Multi-objective heuristics algorithm for dynamic resource scheduling in the cloud computing environment", The Journal of Supercomputing, 2020, https://doi.org/10.1007/s11227-020-03606-2.

[3] Mahendra Bhatu Gawali and Subhash K. Shinde, "Task Scheduling and resource allocation in cloud computing using a heuristic approach", Gawali and Shinde Journal of Cloud Computing: Advances, Systems, and Applications (2018),7:4.

[4] S. R. Shishira and A. Kandasamy, "A Novel Feature Extraction Model for Large-Scale Workload Prediction in Cloud Environment", SN Computer Science (2021), https://doi.org/10.1007/s42979-021-00730-5.

[5] Anurina Tarafdar, Mukta Debnath, Sunirmal Khatua, Rajib K. Das, "Energy and Makespan Aware Scheduling of Deadline Sensitive Tasks in the Cloud Environment", Journal of Grid Computing (2021), https://doi.org/10.1007/s10723-021-09548-0.

[6] Yonghua Zhu, Weilin Zhang, Yihai Chen, and Honghao Gao, "A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment", EURASIP Journal on Wireless Communications and Networking (2019), https://doi.org/10.1186/s13638-019-1605-z.

[7] Habte Lejebo Leka, Zhang Fengli, Ayantu Tesfaye Kenea, Negalign Wake Hundera, Tewodros Gizaw Tohye, and Abebe Tamrat Tegene, "PSO-Based Ensemble Meta-Learning Approach for Cloud Virtual Machine Resource Usage Prediction", Symmetry, vol.15, 2023.

[8] M. Yadav and A. Mishra, "An enhanced ordinal optimization with lower scheduling overhead based novel approach for task scheduling in the cloud computing environment," Journal of Cloud Computing, vol. 12, no. 1, Jan. 2023, doi: 10.1186/s13677-023-00392-z.

[9] X. Zhang, "A fine-grained task scheduling mechanism for digital economy services based on intelligent edge and cloud computing," Journal of Cloud Computing, vol. 12, no. 1, Mar. 2023, doi: 10.1186/s13677-023-00402-0.

[10] G. Saravanan, S. Neelakandan, P. Ezhumalai, and S. Maurya, "Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing," Journal of Cloud Computing, vol. 12, no. 1, Feb. 2023, doi: 10.1186/s13677-023-00401-1.

[11] P. Shukla and S. Pandey, "MOTORS: Multi-Objective Task Offloading and Resource Scheduling Algorithm for Heterogeneous Fog-Cloud Computing Scenario," Jul. 2023, doi: 10.21203/rs.3.rs-3124031/v1.

[12] S. R. K. B. and S. R. E., "Improved Context Aware PSO Task Scheduling in Cloud Computing," Webology, vol. 19, no. 1, pp. 3709–3721, Jan. 2022, doi: 10.14704/web/v19i1/web19244.

[13] H. Zhang, "A Cloud Computing Task Scheduling Method Based on Genetic Algorithm," Proceedings of the 2nd International Conference on Information Economy, Data Modeling and Cloud Computing, ICIDC 2023, June 2–4, 2023, Nanchang, China, 2023, doi: 10.4108/eai.2-6-2023.2334608.

[14] Dharma s and K. P, "An Efficient Task Allocation Using Resource Shortest Scheduling and Fairness Firefly Algorithm In Cloud Computing," Feb. 2023, doi: 10.21203/rs.3.rs-2020473/v1.

[15] H. Zhou, "A Novel Approach to Cloud Resource Management: Hybrid Machine Learning and Task Scheduling," Journal of Grid Computing, vol. 21, no. 4, Nov. 2023, doi: 10.1007/s10723-023-09702-w.

[16] Wiem Matoussi and Tarek Hamrouni, "A new temporal locality-based workload prediction approach for SaaS services in a cloud environment", Journal of King Saud University – Computer and Information Sciences, 2021, doi:10.1016/j.jksuci.2021.04.008.

[17] Vinícius Meyer, Dionatrã F. Kirchoff, Matheus L. Da Silva, Cesar A.F. De Rose, "ML-driven classification scheme for dynamic interference-aware resource scheduling in cloud infrastructures", Journal of Systems Architecture, vol 116, 2021.

[18] Marek Grzegorowski, EftimZdravevski, Andrzej Janusz, Petre Lameski, Cas Apanowicz, Dominik Slezak, "Cost Optimization for Big Data Workloads Based on Dynamic Scheduling and Cluster-Size Tuning", Big Data Research, vol 25, 2021.

[19] Min Cao, Yaoyu Li, Xupeng Wen, Yue Zhao, Jianghan Zhu, "Energy-aware intelligent scheduling for deadline-constrained workflows in sustainable cloud computing", Egyptian Informatics Journal, Volume 24, Issue 2, July 2023.

[20] Jing Bi, Shuang Li, Haitao Yuan, MengChu Zhou, "Integrated deep learning method for workload and resource prediction in cloud systems", Neurocomputing, Volume 424, 1 February 2021.

[21] Jitendra Kumar, Ashutosh Kumar Singh, Rajkumar Buyya, "Self-directed learning based workload forecasting model for cloud resource management", Information Sciences, vol 543, 2021.

[22] Kaixuan Ji, Fa Zhang, Ce Chi, Penglei Song, Biyu Zhou, Avinab Marahatta, Zhiyong Liu, "A joint energy efficiency optimization scheme based on marginal cost and workload prediction in data centers", Sustainable Computing: Informatics and Systems, vol 32, 2021.

[23] Zheng Xiao, Bangyong Wang, Xing Li, Jiayi Du, "Workload-driven coordination between virtual machine allocation and task scheduling", Advances in Parallel and Distributed Computing for Neural Computing, Neural Computing, and Applications, 2019, https://doi.org/10.1007/s00521-019-04022-1.

[24] C. Chandrashekar, P. Krishnadoss, V. Kedalu Poornachary, B. Ananthakrishnan, and K. Rangasamy, "HWACOA Scheduler: Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing," Applied Sciences, vol. 13, no. 6, p. 3433, Mar. 2023, doi: 10.3390/app13063433.

[25] N. Sharma, S. Beniwal, and P. Garg, "Ant Colony Based Optimization Model for Qos-Based Task Scheduling in Cloud Computing Environment," SSRN Electronic Journal, 2022, doi: 10.2139/ssrn.4237751.

[26] S. Chaudhary, V. K. Sharma, R. N. Thakur, A. Rathi, P. Kumar, and S. Sharma, "Modified Particle Swarm Optimization Based on Aging Leaders and Challengers Model for Task Scheduling in Cloud Computing," Mathematical Problems in Engineering, vol. 2023, pp. 1–11, Jun. 2023, doi: 10.1155/2023/3916735.

[27] M. Hosseinzadeh et al., "Improved Butterfly Optimization Algorithm for Data Placement and Scheduling in Edge Computing Environments," Journal of Grid Computing, vol. 19, no. 2, Mar. 2021, doi: 10.1007/s10723-021-09556-0.

[28] S. Mangalampalli et al., "Prioritized Task-Scheduling Algorithm in Cloud Computing Using Cat Swarm Optimization," Sensors, vol. 23, no. 13, p. 6155, Jul. 2023, doi: 10.3390/s23136155.

[29] Jyothi Peta and Srinivas Koppu, "An IoT-Based Framework and Ensemble Optimized Deep Maxout Network Model for Breast Cancer Classification", Electronics 2022, 11, 4137. https://doi.org/10.3390/electronics11244137.

[30] MOHAMMAD DEHGHANI, 'T…PÁN HUBALOVSKY, AND PAVEL TROJOVSKY, "Tasmanian Devil Optimization: A New Bio-Inspired Optimization Algorithm for Solving Optimization Algorithm", Digital Object Identifier, VOLUME 10, 2022, doi: 10.1109/ACCESS.2022.3151641.

[31] H. A. Alsattar, A. A. Zaidan, B. B. Zaidan, "Novelmeta-heuristic bald eagle search optimization algorithm", Artificial Intelligence Review, https://doi.org/10.1007/s10462-019-09732-5.

[32] Xiaoxu Yang, Jie Liu, Yi Liu, Peng Xu, Ling Yu, Lei Zhu, Huayue Chen, and Wu Deng, "A Novel Adaptive Sparrow Search Algorithm Based on Chaotic Mapping and T-Distribution Mutation", Appl. Sci. 2021, 11, 11192. https://doi.org/10.3390/app112311192.

[33] Matthew C. Dickson, Anna S. Bosman, and Katherine M. Malan, "Hybridised Loss Functions for Improved Neural Network Generalisation", arXiv:2204.12244v1 [cs.LG] 26 Apr 2022.

[34] Abdelhady Ramadan, Salah Kamel, Mohamed H. Hassan, Tahir Khurshid, and Claudia Rahmann, "An Improved Bald Eagle Search Algorithm for Parameter Estimation of Different Photovoltaic Models", Processes 2021, 9, 1127. https://doi.org/10.3390/pr9071127.

[35] https://research.google/tools/datasets/google-cluster-workload-traces-2019/.

[36] A. R. A. Rajak, "Emerging Technological Methods for Effective Farming by Cloud Computing and IoT," Emerging Science Journal, vol. 6, no. 5, pp. 1017–1031, Aug. 2022, doi: 10.28991/esj-2022-06-05-07.

[37] Supriya Menon, M. & Rajarajeswari, P., "A Novel Approach for Multi Variant Classification of Medical Data in Short Text ", in Journal of Scientific and Industrial Research, 2021, Volume 80, 0975-1084, pp. 457 – 462.

[38] M. S. Sayeed, H. Abdulrahim, S. F. Abdul Razak, U. A. Bukar, and S. Yogarayan, "IoT Raspberry Pi Based Smart Parking System with Weighted K-Nearest Neighbours Approach," Civil Engineering Journal, vol. 9, no. 8, pp. 1991–2011, Aug. 2023, doi: 10.28991/cej-2023-09-08-012.

[39] S. Gousteris, Y. C. Stamatiou, C. Halkiopoulos, H. Antonopoulou, and N. Kostopoulos, "Secure Distributed Cloud Storage based on the Blockchain Technology and Smart Contracts," Emerging Science Journal, vol. 7, no. 2, pp. 469–479, Feb. 2023, doi: 10.28991/esj-2023-07-02-012.

AUTHORS' PROFILE

Syed Karimunnisa received an M.Tech (CSE) from JNTUA and currently pursuing a Ph.D. from Koneru Lakshmaiah Education Foundation, Guntur, Andhra Pradesh. She has published both international conferences and Journals. She is a Life Member of ISTE and CSTA. Her areas of Interest include Cloud C omputing, Artificial Intelligence, Machine Learning, Deep Learning, Data Mining, and the Internet of Things.

Pachipala Yellamma is working as an Associate Professor in the Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Guntur, Andhra Pradesh. Completed her PhD in Data security in cloud computing. Her research interests include cloud computing, the Internet of Things, Network security, Data compression, Cryptography, and Data security. She is a Life Member in ISTE and IEEE and has some free memberships. She has published several national and international articles in Scopus, web of Science, and SCI. She has three patent publications.