BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **24**, No 3
Sofia • 2024 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2024-0023

Deep Learning-Driven Workload Prediction and Optimization for Load Balancing in Cloud Computing Environment

Syed Karimunnisa[1], *Yellamma Pachipala*[2]

[1]*Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation Vaddesvaram, AP, 522302, India*
[2]*Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation Vaddesvaram, AP, 522302, India*
E-mails: karimun1.syed@gmail.com pachipala.yamuna@gmail.com

***Abstract*:** *Cloud computing revolutionizes as a technology that succeeds in serving large-scale user demands. Workload prediction and scheduling tend to be factors dictating cloud performance. Forecasting the future workload in due to avoid unfair resource allocation, emerges to be a crucial inspecting feature for enhanced performance. The aforementioned issues of interest are addressed in our work by soliciting a Deep Learning driven Max-out prediction model, which efficiently forecasts the future workload by providing a balanced approach for enhanced scheduling with the Tasmanian Devil-Bald Eagle Search* (*TDBES*) *optimization algorithm. The results obtained proved that the TDBES scored efficacy in makespan with 16.75%, migration cost with 14.78%, and a migration efficiency rate of 9.36% over other existing techniques like DBOA, WACO, and MPSO, with additional error analysis of prediction performance using RMSE, MAP, and MAE, among which our contributed approach overrides traditional methods with least error.*

***Keywords*:** *Cloud computing, Task scheduling, Workload prediction, Virtual Machine, Migration.*

1. Introduction

Recently, researchers have been majorly concentric around cloud platforms [1, 2]. It is recognized as the best method for controlling data use, resource allocation, and the provision of different computing IT services. Numerous web-based programs have started to employ cloud computing due to its boundless computational services and the Pay-as-you-go model. By utilizing virtualization strategies and by running several virtual instances on physical computers in data centres, a cloud system [3, 4] can offer more features. Utilizing this tactic, cloud computing providers can make better use of their infrastructure while spending less on energy utilization.

The needs of cloud users to store their data and computational resource requirements are considered as their workload [5]. A desktop workload, for instance, accommodates several users registering into interactive desktop sessions. Several

cloud apps have their unique properties and metrics. The service required by the cloud user and the type of application dictates the type of cloud service needed. With proper workload prediction [6, 7], the cloud can operate more effectively with cost. To estimate the workloads in the cloud environment, several machine learning algorithms, such as Long Short-Term Memory (LSTM) and Deep Belief Network (DBN), are opted as common approaches in current times. Time series models are predicted using a variety of prediction techniques, including the AutoRegressive Integrated Moving Average (ARIMA) model, Support Vector Machine (SVM), Neural Network, Bayesian model, and Moving Average (MA) model. There is yet another category of prediction techniques for figuring out cloud workload trends [8] [9]. Eventually, due to the variation in cloud services, it is difficult to elevate the accuracy of prediction methods since the models are not flexible enough to encounter each service.

The workload prediction model [10] to generate the scheduling algorithm is indigent to handle the solid mapping between the tasks of the users and the resources that are available. This is the major issue in this field, as [11, 12] balancing the computation and scheduling in the finest way is difficult for ascertaining high reliability and improved response time. Different optimization algorithms [13, 14] are in progress to deal with the optimum scheduling by considering it as the NP-hard problem. Meanwhile, the task of scheduling needs [15, 16] have to be automated and improved in terms of several factors like speed, energy efficiency, and optimal scheduling [17-19]. This paper proposes a system with workload prediction and scheduling, where the deep learning algorithm is insisted to train with features like Virtual Machine (VM) capacity as well as task capacity to make the scheduling appropriate via an optimization procedure [20, 21]. The proposed work is carried out in the following manner:

- Proposing an improved Deep learning strategy, with an enhanced Deep Max-out model for predicting workloads, trained in terms of the capacity of respective tasks and Virtual Machines.
- The prediction paves the way for the optimum scheduling of tasks via the optimization strategy named Tasmanian Devil Bald Eagle Search (TDBES) algorithm, which ensures the attainment of objectives like time, cost, and efficiency.

Structure of this suggested work: Section 1 presents an introduction followed by Section 2, where an intense literature review of traditional methods is addressed. Section 3 addresses the proposed cloud computing system model and workload prediction model, and also describes optimal load balancing and task scheduling. Section 4 handles the results and discussions of the contributed model. The conclusion of the work is summarized in Section 5.

2. Related work

In 2021 Matoussi and Hamrouni [22] developed a workload forecasting strategy to enhance throughput planning, assure efficient management of resources, and subsequently serve SLA consensus with cloud users. In this situation, the author put up a novel method for anticipating the number of requests that would be

flooding a SaaS service and setting aside virtual resources for fulfilling the requests of users. The technology would be used to simultaneously acquire two benefits: accurate forecast findings and response time optimization.

In 2021, Meyer et al. [23] developed a Machine Learning-driven classification method permitting optimized cloud resource allocation, majorly interference-aware. The goal of the work is to validate how resource scheduling is impacted by classification algorithms and the changes they impact in workload parameters. The author started by looking at the way the hardware components respond to various apps bearing different dynamic requests. The author investigated several interference classification schemes and assessed their effectiveness while taking into consideration the cloud workload dynamic nature.

In 2020, Grzegorowski et al. [24] introduced a novel technique to construct a robust clustering technique using cloud resources that were tailored to the specific data processing need. The infrastructure-as-a-code framework was used in the given architecture to enable the configuration of clusters and manage them dynamically. It starts by determining the ideal cluster size for completing a task in the allotted amount of time. Further optimization of execution time is carried out to take advantage of the cloud spot market's discounted rates by observing the price range patterns of spot instances and utilizing ARIMA models.

In 2023, Min Cao et al. [25] presented three new approaches for an energy-aware EIS. To save energy, the author first chose the ideal schedule for each activity to execute on respective resources. Secondly, the EIS allocates workflow stack based on the optimal time for every task execution, thereby saving energy by lowering frequency and voltage factors. Further, the EIS takes advantage of the lapsed time created due to shortfalls of task precedence for minimizing dynamic energy usage and satisfying the deadline constraints of workflows.

In 2021, Jing Bi et al. [26] used a logarithmic procedure to lower the SD before applying resource sequences and workloads. An improved filter was used to eliminate extreme spots and noise interference. The data was scaled using the Min-Max method. For time series forecasting, a deep learning integrated technique was developed that uses Machine Learning network models, including Grid-Long Short-Term Memory and Bidirectional-LSTM networks, to generate good-quality predictions of workload arriving and resource requirements at regular time Intervals.

In 2021, Kumar, Singh and Buyya [27] developed an SDWF approach that divulges current predictions to compute the forecasting error trend and aid in improving future prediction efficiency and accuracy. For training neurons, the model makes use of an enhanced heuristic methodology depending on the black hole phenomenon. Additionally, Friedman and Wilcoxon's ranking tests were used in the statistical analysis to confirm the accuracy of the recommended forecasting model.

In 2021, Kaixuan Ji et al. [28] developed a Joint Energy Efficiency Optimization Scheme that jointly considers and co-optimizes the energy used by the cooling system and the servers. JEES includes the dynamic and optimal online task scheduling approach depending on the assessment of the marginal cost measure; a strategy called resource management integrates the workload forecasting model for

managing the set of resources as well as the task-migration approach using the evaluation of marginal cost. The overall energy usage of data centers can be lowered by utilizing the suggested approaches.

In 2019, Zheng Xiao et al. [29] integrated the VM allocation as well as task scheduling, depending on the workload features. The workload is stochastic and time-varying in practice. The author showed that the dataset for the acquired workload has a Markov property that can be represented by a Markov chain. Further, three major operators that characterize the workload are extracted: recurrence, entropy, and persistence. These operators assess the approximate burst times, predictability, and stability of the user requests, accordingly. It is established that the relationship among the VM amount and load characteristic operators is nonlinear to assess the approximate burst times, predictability, and stability of the user requests, accordingly. It is established that the relationship between the VM amount and load characteristic operators is nonlinear.

Chandrasekar et. al. [30], in 2023 contributed a hybrid weighted Ant colony Optimization model that uses the solution function and pheromone update function for optimal task scheduling. The authors' heuristic approach projects the overall improved performance over existing algorithms by evaluating parameters like resource management and execution time.

Sharma, Beniwal and Garg [31] 2020 proposed a QoS-driven Scheduling algorithm based on Ant colony optimization that employed a neural network approach to handle multi-objective scheduling efficiently. The results of the proposed approach have overridden the existing algorithms in terms of metrics like execution time and scheduling cost of user tasks.

Chaudhary et al. [32], in 2023 came up with a modified particle swam optimization to handle high scheduling time and computational expenses during the scheduling process. The MPSO lowers the objective function of cost and make-span that aid in avoiding premature convergence by enhancing the efficiency of local search. The performance of the contributed model showcases the rise in the graph when compared against traditional approaches.

Husseinzadeh et al. [33], in 2021 presented a discrete butterfly optimization algorithm that is based on the levy fight approach to overcome resource wastage and improve the consequence speed of the algorithm. The proposed approach also succeeds in presenting local optima problems by prioritizing the tasks, followed by DBOA that aids in intensive workflow scheduling with enhanced performance. S. Mangalampalli, S. K. Swain and V. K. Mangalampalli [34], in 2023 contributed an enhanced task scheduling method based on cat swam optimization that works by considering task priorities and words it for scheduling. The proposed algorithm works based on the behavior of cats, which overtakes the existing baseline algorithm concerning QOS parameters like spam and resource usage.

3. Methodology

Considering the cloud C having N sets of users U_i, $i=\{1, 2, \dots, N\}$, as well as the M set of tasks generated from users U_i, represented as Tsk_j, $j=\{1, 2, \dots, M\}$, scheduling

the set of tasks to the resources is the major objective, i.e., feasible virtual machines vm_i, where i={vm_1, vm_2, …, vm_n} as well as physical machine pm_i where i={pm_1, pm_2, …, pm_k}. Tasks are scheduled according to their priority as stated by this paradigm. In prior, the workloads are predicted via deep learning strategy.

Workload prediction by Improved Deep Max-out model: An accurate and precise workload forecast plays a crucial role in enhancing cloud performance. It is also discovered that this may increase the effectiveness and lower operating costs. Taking this feature into consideration, this paper adopts the usage of a deep learning model for forecasting the workloads (subsequent tasks), for which, an improved Deep Max-out model is introduced that trains with features like CPU, hours-of-day and day-of-week, for predicting the task as the target label. The proposed prediction model is described in Fig. 1.

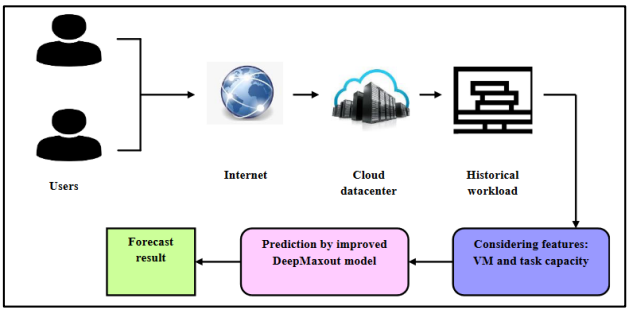


Fig. 1. Workload-prediction scenario

The mathematical modeling of improved deep max out is from [35]:

(1) $$\text{SoftMax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}},$$

(2) $$G - \text{SoftMax} = \frac{\exp(x_i + g(x))}{\sum_{j=1} \exp(x_j + g(x))},$$

$$g(x) = \frac{1}{2}\text{erf}\left(-\frac{\sqrt{2}(\mu_i - x_i)}{2\sigma_i}\right) + \frac{1}{2},$$

$$\text{erf} = \frac{1}{\sqrt{\pi}}\int_{-z}^{z} e^{-t^2}\, dt, t^2 = x_i,$$

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

In Equation (1), a traditional SoftMax activation function that can operate multitudinous classes is provided. It is divided by the sum of the normalized outputs for each class, which range from 0 to 1. The i/p layer, embedded layer, max pooling layer, dropout layer, convolution network layer, and dense layers work using max-out and activation functions to make up the network structure of the Deep Max-out model.

Equation (2) gives an improved SoftMax activation function($G - \text{SoftMax}$) which enhances the model robustness, where x_i is activation and $g(x)$ is Gaussian distributed term, μ, σ is mean and SD.

Equation (3) illustrates this network's max out unit, where $R_{yz} = M \cdot \lambda_{yz} + \gamma_{yz}$, γ is bias, M is input features which include task and virtual machine capacity, η is feature map, and λ is weight.

The feature M is initially subjected to the input layer, and the output derived from M is sent to the embedding layer, where the computation of output takes place. After the operation of the convolution layer, the dropout layer is continued and from the 3rd convolution layer, the outcome is received. After the convolution layer, the result is produced with the help of a max-pooling layer. The dropout layer follows the dense layer in order of appearance. The max-out module uses the input received at the previous dropout layer to compute its output, by later adding the resultant to the dense layer for the final result. The activation function then computes the output of the classification model using the dense layer's output as a last step.

$$E_{\mathrm{he}} = \mathrm{Ls}_1 \frac{E_{\mathrm{SE}}}{\mathrm{Max}_{\mathrm{SE}}} + \mathrm{Ls}_2 \frac{E_{\mathrm{CE}}}{\mathrm{Max}_{\mathrm{CE}}}, \tag{4}$$

Here

$$E_{\mathrm{CE}} = -\frac{1}{\mathrm{Output\_Size}} \sum_{i=1}^{\mathrm{Output\_Size}} y_i \log \hat{y_i} + (1 - y_i) \times 1 \times \log\left(1 - \hat{y_i}\right),$$

$$E_{\mathrm{he}} = \mathrm{Ls}_1 \frac{E_{\mathrm{SE}}}{\mathrm{Max}_{\mathrm{SE}}} + Ls_2 \frac{E_{\mathrm{CE}^*}}{\mathrm{Max}_{\mathrm{CE}^*}}, \tag{5}$$

$$E_{\mathrm{CE}^*} = -\frac{1}{\mathrm{Output\_Size}} \sum_{i=1}^{\mathrm{Output\_Size}} \sum_{j=1}^{c} w_j^{(1-p_{ij})} y_i \log \hat{y_i} + (1 - y_i) \times 1 \times \log\left(1 - \hat{y_i}\right), \tag{6}$$

$$w_j = \log\left(\frac{\max(n_j | j \in c)}{n_j}\right) + 1. \tag{7}$$

The hybrid loss functions [36] provided in Equation (4) are utilized to calculate a model's error. The Sum Squared Error SE(E_{SE}) loss function and Cross Entropy Error CE loss function(E_{CE}) are two often-used error measures for assessing a classification model's efficacy. According to the proposed model, improved hybrid loss function calculation is given in Equation (5), where E_{CE^*} in the dynamically weighted balanced CE (Equation (6)), y_i is an actual label and $\hat{y_i}$ is the predicted label, w_j is equivalent to the frequency log. Equation (7) shows the ratio of n_j class frequency and majority class (computed over training dataset), Ls_1, Ls_2 are the scalar values which are the proportions provided to loss functions, in which $\mathrm{Ls}_1 + \mathrm{Ls}_2 = 1$.

Enhanced Task Scheduling and Load Balancing: An effective and enhanced load balancing strategy can presciently monitor the workload of the VMs and assign tasks to them in accordance. Consider the task Tsk_j as 1000 (Tsk_j will be varied) and consider the physical machine pm_Kas 50 and the virtual machine vm_N between the range 20 and 25. Each pm_K has some vm_N which is in the range between 20 and 25. Then we have to randomly allocate each task into a pm_K. In this, workload prediction model, the task takes place by using the aforementioned improved Deep Maxout model, which is addressed, in Section 4. Here, the Virtual Machine capacity and Task requirements are taken as features given to the Improved Deep max out, and the target labels of deep max out are 0, 1 and 2. The conditions for underload, equal load, and overload are as follows:

- When the capacity of the VM overrules the task needs the label of the target class is defined to 0, i.e., the machine is underloaded.
- When the capacity of the VM equals the task requirements the label of the target class is defined to 1, i.e., the machine is equally loaded.

- When the capacity of the VM fails to handle the task requirements the label of the target class is defined to 2, i.e., the machine is overloaded.

The optimal scheduling carried out is evaluated against performance metrics like Makespan ($\boldsymbol{F_1}$), Migration cost ($\boldsymbol{F_2}$), and Migration efficiency ($\boldsymbol{F_3}$). The model of scheduling is illustrated in Fig. 2.

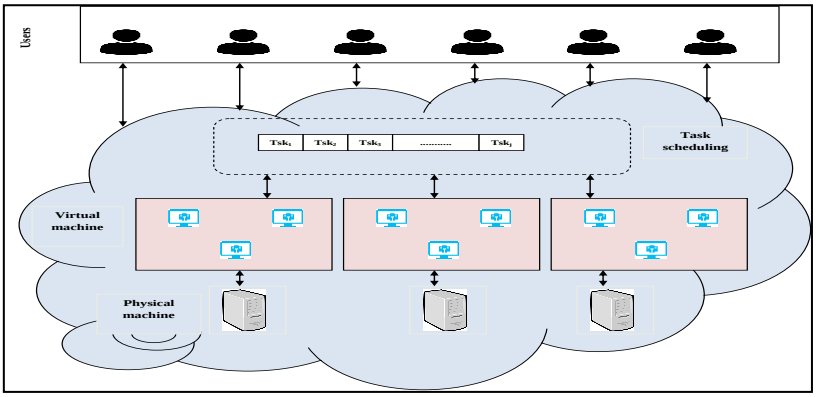


Fig. 2. Scheduling of tasks in cloud computing

Formulating lower bound and upper bound as per input solution for loading

$$\mathrm{Lb} = 0\left(\mathrm{Zeros}\left(\mathrm{len}(\mathrm{Machine\_underload})\right)\right),$$

$$\mathrm{Ub} = 1\left(\mathrm{Ones}\left(\mathrm{len}(\mathrm{Machine\_underload})\right)\right),$$

and the problem size of TDBES is the count of the underload machine. The next equation gives the formulation of the objective function, where w_1, w_2, w_3 are the random weights in [0, 1]:

$$\mathrm{Obj} = (w_1 \times F_1) + (w_2 \times F_2) + (w_3 \times F_3). \tag{8}$$

Makespan (F_1) is coined as the overall Computation Time CT_i (Equation (10)) to execute the task, in which N is the set of virtual machines:

$$F_1 = \underset{1 \le i \le N}{\mathrm{Max}}\{\mathrm{CT}_i\}, \tag{9}$$

$$\mathrm{CT}_i = \sum_{j=1}^{n} \frac{T_j \times \mathrm{length}}{\mathrm{vm}_N \times \mathrm{Pes\_Num} \times X\mathrm{vm}_N}. \tag{10}$$

Migration cost (F_2): The $M \times M$ matrix, whose columns and rows indicate the physical machine pm_K migration cost, is used to compute the virtual machine vm_N migration cost. Here, the matrix has identical row and column components, i.e., (1, 1) and (2, 2) both exhibiting lower migration costs.

Migration efficiency F_3 is calculated by using the migration value:

$$F_3 = \frac{1}{F_1}. \tag{11}$$

Load balancing process: In Table 1, 0 means underloaded, 1 means equally loaded, and 2 means overloaded. Here, the underload machine (physical machine pm_K) is (2, 5), and the overload machine (physical machine pm_K) is (4, 1). For example, in Table 2, consider the solution with the range of (0, 1) is (0.2, 0.1) with index (1, 2). We will perform an index-wise sort on this response in Table 3. According to the index sorted, the underloaded machine is arranged as (5, 2), where

5 is the second machine and 2 is the first machine. Here, two tasks are overloaded. As machine 5 only has one task space, one load task is assigned to it, while a second overload task is sent to machine 2, which has two task spaces. Finally, the load gets balanced.

Table 1. Load balancing process

pm_K	vm_N	Tsk_j	Load prediction
1	3	4	2
2	4	3	0
3	2	2	1
4	1	2	2
5	5	4	0

Table 2. Index and solution before sorting

Index	1	2
Solution	0.2	0.1

Table 3. Index and solution after index-wise sort

Index	2	1
Solution	0.2	0.1

TDBES Algorithm for optimal task scheduling. This section explains the contributed meta-heuristic TDBES, that fusions TDO [37] and BES [38], and presents its mathematical modelling. The T-devil (Tasmanian devil) uses two approaches for feeding: live prey attack or feasting on the carrion of deceased animals. This behavior is simulated in TDBES. Based on the problem's restrictions, the agent's random population is created initially. The problem-solving area search agent is the TDBES population members who propose the problem variables' candidate values depending on the search area location. As a result, it is possible to conceptualize each member of a population functionally in vector form whose elements correspond to the variables count in the issue. To model the set of TDBES members, a matrix X in the next equation can be employed, where X is the Tasmanian population, N is the searching count of T-devil, X_i is the i-th candidate solution, as well as m depicts the count of the given problem variables:

$$X = \begin{bmatrix} X_1 \\ : \\ X_i \\ : \\ X_N \end{bmatrix} = \begin{bmatrix} x_{1,1} \ldots x_{1,j} \ldots x_{1,m} \\ : \\ x_{i,1} \ldots x_{i,j} \ldots x_{i,m} \\ : \\ x_{N,1} \ldots x_{N,j} \ldots x_{N,m} \end{bmatrix}_{N \times m} . \quad (12)$$

The function of the defined objective can be determined by putting the readings of alternative solutions into objects of the defined objective function. In the next equation, a vector F is utilized to model the values attained for the defined objective function:

(13) $$\mathrm{F} = \begin{bmatrix} \mathrm{F}_1 \\ : \\ \mathrm{F}_i \\ : \\ \mathrm{F}_N \end{bmatrix} = \begin{bmatrix} F(X_1) \\ : \\ F(X_i) \\ : \\ F(X_N) \end{bmatrix}_{N\times 1} .$$

Occasionally, the T-devil hunts for local carrion, whereas other raptors that hunt for large prey but are unable to complete the tasks preside above the T-devil. Another population member's position is taken into account as Carrion puts for each T-devil in the TDBES design. The next equation simulates one of these instances' random selection, where C_i depicts the carrion that is chosen; depending on C_i, the T-devil is given a new location in the search area:

(14) $$C_i = X_k, i = 1, 2, \ldots, N, \text{ and } k \in \{1, 2, \ldots, N | k \neq i\}.$$

Equation (12) models the T-devil's motion style, where $I \in (1, 2)$ depicts random numbers, $r \in (0,1)$ depicts random numbers, and $x_{i,j}^{\mathrm{new},S1}$, gives T-devil's current updated position depending on aforementioned strategy. Following the calculation of the T-devil's updated position in Equation (15), if the goal function value is more at the newly updated position, then the position is acceptable; if not, the T-devil remains where it was. In Equation (16), according to the proposed concept, new update of Tasmanian is done by hybridizing the TDO and proposed BES update, where RF is the Random Factor (Equation (18)) [39], $r_2 \in (0, 1)$ is the random value, $\mathrm{IT_{Max}}$ denotes the Maximum number of Iteration, and $T$is the current iteration, Levy denotes levy flight update with Equation (19). The conventional BES update equation is in Equation (20), where α is the position change controlling parameter.

(15) $$x_{i,j}^{\mathrm{new},S1} = \begin{cases} x_{i,j} + r \times \left(C_{i,j} - I \times x_{i,j}\right) \text{ if } F_{C_i} \prec F_i, \\ x_{i,j} + r \times \left(x_{i,j} - C_{i,j}\right) \text{ otherwise,} \end{cases}$$

(16) $$X_i = \begin{cases} X_i^{\mathrm{new},S1} \text{ if } F_i^{\mathrm{new},S1} \prec F_i, \\ X_i \text{ otherwise,} \end{cases}$$

(17) $$x_{i,j}^{\mathrm{new},S1} = \begin{cases} x_{i,j} + \mathrm{RF} \times \left(C_{i,j} - I \times x_{i,j}\right) \text{ if } F_{C_i} \prec F_i, \\ x_{\mathrm{new}} = x_{\mathrm{best}} + \alpha \times r(x_{\mathrm{mean}} - x_i) \times L\text{evy otherwise,} \end{cases}$$

(18) $$\mathrm{RF} = r_1 \times \sin(r_2), r_1 = \frac{1.5\times(\mathrm{IT_{Max}} - t + 1)}{\mathrm{IT_{Max}}},$$

(19) $$\mathrm{Levy} = \frac{T(1+\beta)\times\left(\sin\left(\frac{\pi\beta}{2}\right)\right)^{1/\beta}}{T\left(\frac{1+\beta}{2}\right)\times\beta\times\left(2^{\left(\frac{\beta-1}{2}\right)}\right)},$$

(20) $$\mathrm{x_{new}} = x_{\mathrm{best}} + \alpha \times r(x_{\mathrm{mean}} - x_i),$$

(21) $$P_i = X_k, i = 1, 2, \ldots, N, k \in \{1, 2, \ldots, N | k \neq i\}.$$

During the process of *i*-th T-devil updating, another population member position is taken as the location of prey. Equation (21) simulates the process of choosing prey, where $k \in (1, N)$ is a natural random integer, P, and depicts the prey that is chosen. Equation (22) calculates a newer position for the T-devil based on the precise position of the prey. T-devil new location calculation replaces the prior location when it

enhances the objective function value. Equation (23) serves as an example of the second strategy.

$$x_{i,j}^{\text{new},S2} = \begin{cases} x_{i,j} + r \times \left(P_{i,j} - I \times x_{i,j}\right) \text{ if } F_{P_i} \prec F_i, \\ x_{i,j} + r \times \left(x_{i,j} - P_{i,j}\right) \text{ otherwise.} \end{cases} \tag{22}$$

$$X_i = \begin{cases} X_i^{\text{new},S2} \text{ if } F_i^{\text{new},S2} \prec F_i, \\ X_i \text{ otherwise.} \end{cases} \tag{23}$$

Equation (24) can be used to get the radius R of neighborhood, which represents the area where the T-devil follows its prey. So, using Equation (25), which simulates the T-devil's chasing behaviour numerically, a new position can be determined. If the updated position offers an improved position than the existing position, the T-devil algorithm agrees upon it. The position updating process of T-devil is shown in Equation (26). As per the contributed model in Equation (27), the T-devil algorithm update process is done by introducing [40] the proposed factor PF (Equation (28)).

$$R = 0.01\left(1 - \frac{t}{\text{IT}_{\text{Max}}}\right), \tag{24}$$

$$x_{i,j}^{\text{new}} = x_{i,j} + (2r - 1) \times R \times x_{i,j}, \tag{25}$$

$$X_i = \begin{cases} X_i^{\text{new}} \text{ if } F_i^{\text{new}} \prec F_i, \\ X_i \text{ otherwise,} \end{cases} \tag{26}$$

$$X_i = \begin{cases} X_i^{\text{new}} \times \text{PF} \text{ if } F_i^{\text{new}} \prec F_i, \\ X_i \text{ otherwise,} \end{cases} \tag{27}$$

$$\text{PF} = \exp\left(\frac{-i}{\delta \times \text{IT}_{\text{Max}}}\right). \tag{28}$$

Shuffle crossover [41] aids in the development of offspring who are not dependent on the parents' crossover points.

Shuffle Crossover Operation

Step 1. Select two parents: parent A and parent B from the parent pool

Step 2. Two offspring: C^{t+1}, D^{t+1} are created

Step 3. Shuffle the genes randomly in both the parents.

Step 4. Select the 1-point crossover point cp randomly from the group $\{1, 2, \ldots, n-1\}$

Step 5. For i=1 to cp do

Step 6. $C_i^{t+1} = a_i^t$

Step 7. $D_i^{t+1} = b_i^t$

Step 8. end do

Step 9. For $i = cp + 1$ to n do

Step 10. $C_i^{t+1} = b_i^t$

Step 11. $D_i^{t+1} = a_i^t$

Step 12. end do

Algorithm. **Pseudocode of proposed TDBES for optimal load balancing and task scheduling**

Initialize t (iterations) and N (member of the population), Set the T-devil's location to its initial value and assess the objective function.

Step 1. For *t* in 1 to *N*

Step 2. For *t* in 1 to *N*

Step 3. If $\text{probb} < 0.5, \text{probb} = \text{rand\_value}$.

Step 4. Choose the carrion prey for *i*-th T-devil by using Equation (14)

Step 5. Evaluate an updated value of the T-devil by Equation (16)

Step 6. The *i*-th T-devil proposed update is done via Equation (17)

Step 7. Else

Step 8. Choose the prey_value of *i*-th T-devil using Equation (21)

Step 9. Evaluate the new modified value of the T-devil by Equation (22)

Step 10. *i*-th T-devil update is done via Equation (23)

Step 11. Upgrade R via Equation (24)

Step 12. Evaluate new update of *i*-th T-devil in x_i neighborhood by using Equation (25)

Step 13. *i*-th T-devil proposed update is done via Equation (27)

Step 14. End If

Step 15. End For

Step 16. Shuffle crossover operation

Step 17. Save the optimal suggested solution so far

Step 18. End For

4. Results and discussions

The proposed workload prediction and scheduling of tasks are implemented in Cloud-sim. Google Cluster Workload Traces, about the 2019 dataset was considered and assembled from [42]. The efficiency of the TDBES work compared with the conventional strategies, like Discrete Butterfly Optimization Algorithm (DBOA), Weighted Ant Colony Optimization (WACO), and Modified Particle Swarm Optimization (MPSO), correspondingly. Moreover, it is assessed by evaluating parameters like Execution Time, Communication cost, Fitness, Migration cost, and so on. The analysis is also done while ranging the task count from 500 up to 2000.

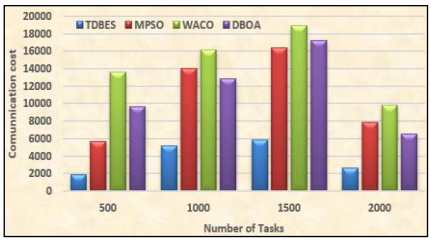
Dataset. The data set considered for our work is the Google Cluster Workload Traces 2019 dataset includes data, such as:

- Utilization of CPU histograms throughout for every 5-minute duration;
- Information related to allocation sets.

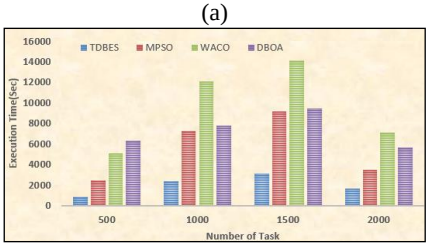
4.1. Evaluation of execution time and communication cost

The performance of the TDBES is computed over the WACO, MPSO, and DBOA for workload prediction, and scheduling the tasks about transfer cost and time of execution is displayed in Fig. 3. The communication cost and execution time ought to be lower for the proposed TDBES when compared to other approaches.

In particular, the TDBES yielded a minimized communication cost with a 500th task count against other compared algorithms with task counts of 1000, 1500, and 2000. Moreover, the TDBES obtained an execution time of 1648 s with task counts of 2000, which is considerably minimized. Owing to the employment of improved deep max-out with a hybrid optimization model (TDO and BES), enhanced outcomes are achieved.



(a)



(b)

Fig. 3. Communication cost over number of Tasks (a); Execution time over number of Tasks (b)

4.2. Evaluation of makespan

The assessment of performance metrics of the TDBES over the WACO, MPSO, and DBOA subjected to predicting workloads and task scheduling as depicted in Fig. 4 tends to project a lowered make-span for the TDBES approach than the conventional methodologies, when the count of tasks is tuned at a count of 500. Hence, the findings indicate the reliability of the TDBES approach effective in workload prediction and task scheduling with a lowered makespan.

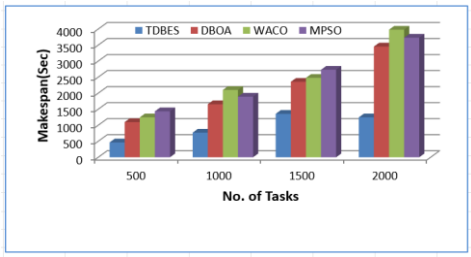
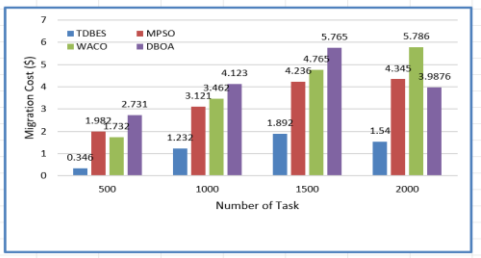


Fig. 4. Makespan over number of Tasks

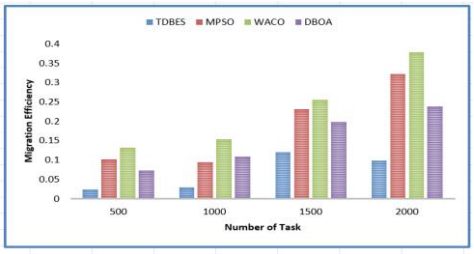
4.3. Evaluation of migration cost and efficiency

Fig 5a and b depict the migration cost and efficiency of the TDBES inspected over the existing methods for the prediction of workload and task scheduling. The performance analysis is carried out by varying the set of tasks. The cost and efficiency

of task migration tend to be improved for the TDBES approach with a lower migration cost of 1.55, when scrutinized against the traditional methods bearing readings, DBOA=4.987, WACO=6.786, and MPSO=5.345, correspondingly. Additionally, the migration efficiency of the TDBES (0.0989) is greater than the DBOA (0.0238), MPSO (0.325), and WACO (0.375), respectively(task=2000). Thus, the betterment of the TDBES is proved in terms of migration cost and migration efficiency.



(a)



(b)

Fig. 5. Migration cost over number of tasks (a); Migration efficiency vs number of tasks (b)

4.4. Convergence evaluation

The convergence analysis on TDBES is compared to the existing techniques, such as DBOA, WACO, CSO (cat swap optimization), and MPSO, which are presented in Fig 6.

During the 0th iteration, the TDBES and the other methods generated the highest cost value, as that drastically came down with increased iterations. However, the TDBES acquired the minimized cost rate, mainly at the 26th iteration; the TDBES scored the least cost value of 4.427. Based on the graph, the TDBES convergence rate improves rapidly over the existing schemes, which leads to a reduction in cost and exact prediction of workload and task scheduling.

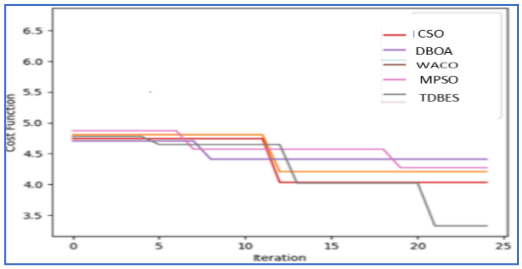
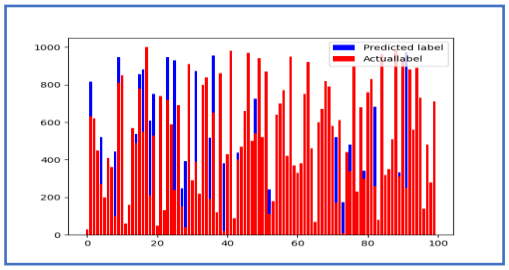


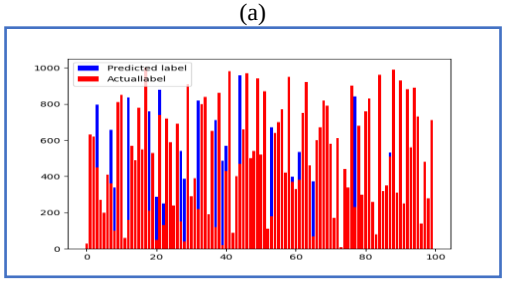
Fig. 6. Cost function vs number of iterations

4.5. Regression evaluation

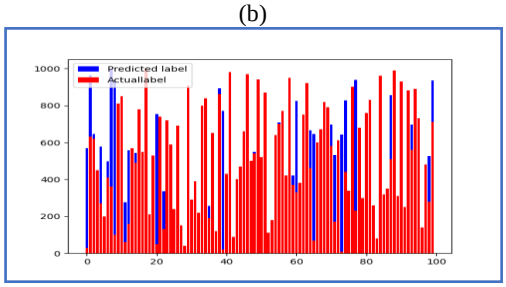
The evaluation of the regression metric of the contributed model against other existing models for the prediction of workloads is presented in Fig. 7. In this assessment, we have presented the actual label and predicted label for the TDBES and the existing methodologies like RNN, CNN, and NN. The figures depict that our proposed model generated more identical values in both the actual and predicted labels, while the RNN, NN, and CNN tend to produce more discrepancy values.



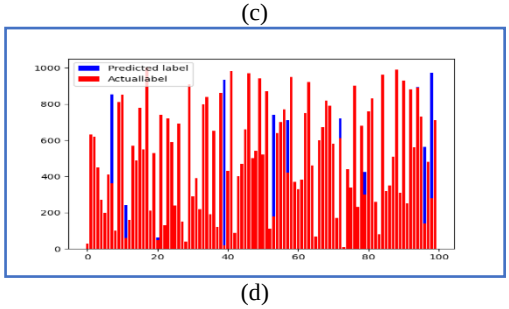
(a)



(b)



(c)



(d)

Fig. 7. Regression analysis of CNN (a); regression analysis of NN (b); regression analysis of RNN (c); regression analysis of TDBES (d)

4.6. Error analysis of Improved Deep Max-out

Table 4 describes the error analysis of the proposed algorithm over legacy approaches like NN, RNN, CNN, Bi-GRU, and Conventional Deep Max-out about MAP, RMSE, MAE, and MALE. Here, the improved Deep Max-out has accomplished the lowest error measure ratings. Similarly, the RMSE of the improved Deep Max-out is 0.102, which is much lesser over the CNN (0.261), RNN (0.344), NN (0.163), Bi-GRU (1.182) and Conventional Deep Max-out (1.152), correspondingly. Simultaneously, the proposed recorded the minimized MAE=0.023, and MSLE=0.216, respectively.

Table 4. Error rate evaluation

Models	MAP	RMSE	MAE	MALE
NN	0.323	0.163	0.058	0.446
CNN	0.531	0.261	0.152	0.521
RNN	0.452	0.344	0.143	0.688
Bi-GRU	2.183	1.182	0.655	1.403
Improved Deep Max-out	0.122	0.102	0.023	0.216

4.7. Prediction analysis

Table 5 illustrates the prediction assessment of TDBES against RNN, NN, and CNN models in terms of success rate for predicting over-predictions and under-predictions for the considered amount of workload. Here, the TDBES has accomplished superior prediction values than the RNN, NN, and CNN algorithms.

Table 5. Prediction evaluation

Models	Success rate	Over-prediction	Under-prediction
RNN	54.7	48	52
NN	50.5	52	48
CNN	49.5	52	48
TDBES	96.5	15	26

5. Conclusion

The presented work successfully contributes an efficient prediction model that predicts the upcoming workload arriving at a machine depending on the data patterns of the previous workload using the Deep Max-out prediction model. The workload arriving at the machines that tend to unbalance the VM is balanced and scheduled with an optimal TDBES Algorithm that performs migration of tasks among VMs and aids in efficient scheduling considering evaluation metrics like makespan, Migration cost, and efficiency. The statistical analysis conducted has demonstrated that the proposed method effectively lowers the communication cost over conventional methods that exhibit high communication costs. Moreover, the analysis of prediction results highlights the importance of considering constraints like time of execution, Fitness, migration cost, and makespan of tasks, all of which are significantly lower in the proposed model compared to conventional methods. Notably, the execution time is minimized, whereas traditional methods exhibit poor performance with longer execution times.

References

1. Ebadifard, F., S. M. Babamir. Autonomic Task Scheduling Algorithm for Dynamic Workloads through a Load Balancing Technique for the Cloud-Computing Environment. – Cluster Computing, 2020. DOI: 10.1007/s10586-020-03177-0.
2. Devi, K. L., S. Valli. Multi-Objective Heuristics Algorithm for Dynamic Resource Scheduling in the Cloud Computing Environment. – The Journal of Supercomputing, 2020. DOI: 10.1007/s11227-020-03606-2.
3. Kaur, G., A. Bala. Prediction-Based Task Scheduling Approach for Food Plain Application in Cloud Environment. – Computing, Vol. **103**, 2021, pp. 895-916. DOI: 10.1007/s00607-021-00936-8.
4. Kaur, G., A. Bala. OPSA: An Optimized Prediction-Based Scheduling Approach for Scientific Applications in Cloud Environment. – Cluster Computing, 2021. DOI: 10.1007/s10586-021-03232-4.
5. Li, H., Y. Zhao, S. Fang. CSL-Driven and Energy-Efcient Resource Scheduling in the Cloud Data Center. – The Journal of Supercomputing. DOI: 10.1007/s11227-019-03036-9.
6. Karimunnisa, S., Y. Pachipala. Task Classification and Scheduling Using Enhanced Coot Optimization in Cloud Computing. – International Journal of Intelligent Engineering and Systems, 2023. DOI: 10.22266/ijies2023.1031.43.
7. Peng, Z., J. Lin, D. Cui, Q. Li, J. He. A Multi-Objective Trade-Off Framework for Cloud Resource Scheduling Based on the Deep Q-Network Algorithm. – Cluster Computing, 2019. DOI: 10.1007/s10586-019-03042-9.
8. Shishira, S. R., A. Kandasamy. A Novel Feature Extraction Model for Large-Scale Workload Prediction in Cloud Environment. – SN Computer Science, 2021. DOI: 10.1007/s42979-021-00730-5.
9. Tarafdar, A., M. Debnath, S. Khatua, R. K. Das. Energy and Makespan Aware Scheduling of Deadline Sensitive Tasks in the Cloud Environment. – Journal of Grid Computing, 2021. DOI: 10.1007/s10723-021-09548-0.
10. Pachipala, Y., D. B. Dasari, V. V. R. M. Rao, P. Bethapudi, T. Srinivasarao. Workload Prioritization and Optimal Task Scheduling in Cloud: Introduction to Hybrid Optimization Algorithm. – Wireless Networks, 2024. DOI: 10.1007/s11276-024-03793-3.
11. Nabi, S., M. Ahmed. OG-RADL: Overall Performance-Based Resource-Aware Dynamic Load-Balancer for Deadline Constrained Cloud Tasks. – The Journal of Supercomputing, 2020. DOI: 10.1007/s11227-020-03544-z.
12. Singh, H., A. Bhasin, P. R. Kaveri. QRAS: Efficient Resource Allocation for Task Scheduling in Cloud Computing. – SN Appl. Sci., Vol. **3**, 2021, 474. DOI: 10.1007/s42452-021-04489-5.
13. Leka, H. L., Z. Fengli, A. T. Kenea, N. W. Hundera, T. G. Tohye, A. T. Tegene. PSO-Based Ensemble Meta-Learning Approach for Cloud Virtual Machine Resource Usage Prediction. – Symmetry, Vol. **15**, 2023.
14. Karimunnisa, S., Y. Pachipala. An AHP Based Task Scheduling and Optimal Resource Allocation in Cloud Computing. – International Journal of Advanced Computer Science and Applications, 2023. DOI: 10.14569/ijacsa.2023.0140317.
15. Menon, S. M., P. Rajarajeswari. A Hybrid Machine Learning Approach for Drug Repositioning. – International Journal of Computer Science and Network Security, Vol. **20**, 2020, Issue 12, pp. 217-223.
16. Umbarkar, A. J., P. D. Sheth. Crossover Operators in Genetic Algorithms: A Review. – ICTACT Journal on Soft Computing, Vol. **6**, October 2015, Issue 1. DOI: 10.21917/ijsc.2015.0150.
17. Dickson, M. C., A. S. Bosman, K. M. Malan. Hybridised Loss Functions for Improved Neural Network Generalisation. – arXiv:2204.12244v1 [cs.LG] 26 April 2022.
18. Ramadan, A., S. Kamel, M. H. Hassan, T. Khurshid, C. Rahmann. An Improved Bald Eagle Search Algorithm for Parameter Estimation of Different Photovoltaic Models. – Processes 2021, Vol. **9**, 1127. DOI: 10.3390/pr9071127.
19. Ilankumaran, A., S. J. Narayanan. An Energy-Aware QoS Load Balance Scheduling Using Hybrid GAACO Algorithm for Cloud. – Cybernetics and Information Technologies, Vol. **23**, 2023, No 1, pp. 161-177.

20. S r i v a s t a v a, V., K. D w i v e d i, A. K. S i n g h. Cryptocurrency Price Prediction Using Enhanced PSO with Extreme Gradient Boosting Algorithm. – Cybernetics and Information Technologies, Vol. **23**, 2023, No 2, pp. 170-187.
21. G u l i a s h k i, V., L. K i r i l o v, A. N u z i. Optimization Models and Strategy Approaches Dealing with Economic Crises, Natural Disasters, and Pandemics – an Overview. – Cybernetics and Information Technologies, Vol. **23**, 2023, No 4, pp. 3-25.
22. M a t o u s s i, W., T. H a m r o u n i. 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.
23. M e y e r, V. D., F. K i r c h o f f, M. L. D a S i l v a, C. A. F. D e R o s e. ML-Driven Classification Scheme for Dynamic Interference-Aware Resource Scheduling in Cloud Infrastructures. – Journal of Systems Architecture, Vol. **116**, 2021.
24. G r z e g o r o w s k i, M., E. Z d r a v e v s k i, A. J a n u s z, P. L a m e s k i, C. A p a n o w i c z, D. S l e z a k. Cost Optimization for Big Data Workloads Based on Dynamic Scheduling and Cluster-Size Tuning. – Big Data Research, Vol. **25**, 2021.
25. C a o, M., Y. L i, X. W e n, Y. Z h a o, J. Z h u. Energy-Aware Intelligent Scheduling for Deadline-Constrained Workflows in Sustainable Cloud Computing. – Egyptian Informatics Journal, Vol. **24**, 2023, Issue 2.
26. B i, J., S. L i, H. Y u a n, M. C. Z h o u. Integrated Deep Learning Method for Workload and Resource Prediction in Cloud Systems. – Neurocomputing, Vol. **424**, 2021, No 1.
27. K u m a r, J., A. K. S i n g h, R. B u y y a. Self-Directed Learning Based Workload Forecasting Model for Cloud Resource Management. – Information Sciences, Vol. **543**, 2021.
28. J i, K., F. Z h a n g, C. C h i, P. S o n g, B. Z h o u, A. M a r a h a t t a, Z. L i u. A Joint Energy Efficiency Optimization Scheme Based on Marginal Cost and Workload Prediction in Data Centers. – Sustainable Computing: Informatics and Systems, Vol. **32**, 2021.
29. X i a o, Z., B. W a n g, X. L i, J. D u. Workload-Driven Coordination between Virtual Machine Allocation and Task Scheduling. – Advances in Parallel and Distributed Computing for Neural Computing, Neural Computing and Applications, 2019. DOI: 10.1007/s00521-019-04022-1.
30. C h a n d r a s h e k a r, C., P. K r i s h n a d o s s, V. K. P o o r n a c h a r y, B. A n a n t h a k r i s h n a n, K. R a n g a s a m y. HWACOA Scheduler: Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing. – Applied Sciences, Vol. **13**, 2023, No 6, p. 3433. DOI: 10.3390/app13063433.
31. S h a r m a, N., S. B e n i w a l, P. G a r g. Ant Colony Based Optimization Model for QoS-Based Task Scheduling in Cloud Computing Environment. – SSRN Electronic Journal, 2022. DOI: 10.2139/ssrn.4237751.
32. C h a u d h a r y, S., V. K. S h a r m a, R. N. T h a k u r, A. R a t h i, P. K u m a r, S. S h a r m a. Modified Particle Swarm Optimization Based on Aging Leaders and Challengers Model for Task Scheduling in Cloud Computing. – Mathematical Problems in Engineering, Vol. **2023**, 2023, No 1. DOI: 10.1155/2023/3916735.
33. H o s s e i n z a d e h, M., et al. Improved Butterfly Optimization Algorithm for Data Placement and Scheduling in Edge Computing Environments. – Journal of Grid Computing, Vol. **19**, March 2021, No 2, DOI: 10.1007/s10723-021-09556-0.
34. M a n g a l a m p a l l i, S., S. K. S w a i n, V. K. M a n g a l a m p a l l i. Multi Objective Task Scheduling in Cloud Computing Using Cat Swarm Optimization Algorithm. – Arabian Journal for Science and Engineering, Vol. **47**, September 2021, No 2, pp. 1821-1830. DOI: 10.1007/s13369-021-06076-7.
35. P e t a, J,. S. K o p p u. An IoT-Based Framework and Ensemble Optimized Deep Maxout Network Model for Breast Cancer Classification. – Electronics, Vol. **11**, 2022, 4137. DOI: 10.3390/electronics11244137.
36. D i c k s o n, M. C., A. S. B o s m a n, K. M. M a l a n. Hybridised Loss Functions for Improved Neural Network Generalisation. – arXiv:2204.12244v1 [cs.LG] 26 April 2022.
37. D e h g h a n i, M., T. P. H u b a l o v s k y, P. T r o j o v s k y. Tasmanian Devil Optimization: A New Bio-Inspired Optimization Algorithm for Solving Optimization Algorithm. – Digital Object Identifier, Vol. **10**, 2022. DOI: 10.1109/Access.2022.3151641.

38. A l s a t t a r, H. A., A. A. Z a i d a n, B. B. Z a i d a n. Novelmeta-Heuristic Bald Eagle Search Optimization Algorithm. – Artificial Intelligence Review, Vol. **53**, 2020, No 6. DOI: 10.1007/s10462-019-09732-5.
39. R a m a d a n, A., S. K a m e l, M. H. H a s s a n, T. K h u r s h i d, C. R a h m a n n. An Improved Bald Eagle Search Algorithm for Parameter Estimation of Different Photovoltaic Models. – Processes, Vol. **9**, 2021, 1127. DOI: 10.3390/pr9071127.
40. Y a n g, X., J. L i u, Y. L i u, P. X u, L. Y u, L. Z h u, H. C h e n, W. D e n g. A Novel Adaptive Sparrow Search Algorithm Based on Chaotic Mapping and T-Distribution Mutation. – Appl. Sci., Vol. **11**, 2021, 11192. DOI: 10.3390/app112311192.
41. U m b a r k a r, A. J., P. D. S h e t h. Crossover Operators in Genetic Algorithms: A Review. – ICTACT Journal on Soft Computing, Vol. **6**, 2015, Issue 1. DOI: 10.21917/ijsc.2015.0150.
42. **https://research.google/tools/datasets/google-cluster-workload-traces-2019/**

Received: 11.12.2023; Second Version: 05.08.2024; Accepted: 10.08.2024