International Journal of Web Engineering and Technology

ISSN online: 1741-9212 - ISSN print: 1476-1289
https://www.inderscience.com/ijwet

Deep learning-based task scheduling in edge computing

Bantupalli Nagalakshmi, Sumathy Subramanian

DOI: 10.1504/IJWET.2024.10061935

Article History:

Received:	03 May 2023
Last revised:	28 November 2023
Accepted:	06 December 2023
Published online:	29 April 2024

Deep learning-based task scheduling in edge computing

Bantupalli Nagalakshmi

School of Computer Science and Engineering,
Vellore Institute of Technology,
Vellore-632014, Tamil Nadu, India
Email: b.nagalakshmi2016@vitstudent.ac.in

Sumathy Subramanian*

School of Computer Science Engineering and Information Systems,
Vellore Institute of Technology,
Vellore-632014, Tamil Nadu, India
Email: ssumathy@vit.ac.in
*Corresponding author

Abstract: A potential paradigm called edge computing (EC) has recently come to light that supports internet of things (IoT) applications that are resource allocation with low latency services at the network edge. For scheduling the application tasks, the edge server's constrained processing capabilities present significant difficulties. The IoT-EC scenario is used in this research to study the task scheduling problem, and various jobs are scheduled to virtual machines (VMs) set up the edge server by maximising long-term task satisfaction. The proposed optimal task scheduling considers parameters like makespan, execution time, execution cost, and risk probability. Particularly, the risk probability estimation is done by the deep convolutional neural network (D-CNN). This estimation is based on task security and VM security. The scheduling of tasks is carried out via the new hybrid bald eagle Archimedes optimisation (HBEAO) by considering a multi-objective to minimise the makespan, execution time, execution cost, and risk probability. The proposed model is validated with existing models in terms of execution cost, execution time, fitness, makespan, risk probability, etc. It is observed that the HBEAO model attains less execution cost ($37.27), execution time (0.99 seconds), fitness (3.48%), risk probability (0.19%) and computation time (2,325.87 sec) respectively.

Keywords: task scheduling; deep learning; edge computing; server; optimisation; internet of things; IoT; deep convolutional neural network; D-CNN.

Reference to this paper should be made as follows: Nagalakshmi, B. and Subramanian, S. (2024) 'Deep learning-based task scheduling in edge computing', *Int. J. Web Engineering and Technology*, Vol. 19, No. 1, pp.20–43.

Biographical notes: Bantupalli Nagalakshmi is a research scholar in School of Computer Science and Engineering, Vellore Institute of Technology, Vellore. She pursed her BTech in Computer Science and Engineering from Sri Venkateswara University, Tirupati and MTech in Computer Science and

Engineering from JNTU University. Her area of research is algorithms and networks. She is a life member of the Computer Society of India and the Indian Society for Technical Education.

Sumathy Subramanian is a Professor at the School of Computer Science Engineering and Information systems, Vellore Institute of Technology, pursued her BE in Electronics and Communication Engineering, MTech in Computer Science and Engineering and PhD in Wireless Networks. She is guiding research scholars in the area of text mining, cloud computing and machine learning. She has published more than 50 papers in reputed journals and conferences at national and international level and has executed a DST-SERB project and British Council project in collaboration with Nottingham Trent University, UK. She is a life member of the Computer Society of India and the Indian Society for Technical Education.

This paper is a revised and expanded version of a paper entitled 'Self-improved pelican optimization for task scheduling in edge computing: neural network based risk probability estimation', presented at 8th International Symposium on Intelligent Informatics (ISI'23), Springer, 21 November 2023.

1 Introduction

The edge computing (EC) paradigm moved the processing capacity as close to the user as feasible by creating a new tier between the cloud and the field tier (Gezer and Wagner, 2021). This led to the emergence of the EC paradigm, in which a large number of small-scale computing facilities, known as edge servers, were built at the network edge along the path from the user end to the cloud to receive offloaded tasks from mobile devices. EC was intended to significantly reduce task reaction time in an energy-efficient manner while also improving security and privacy (Cai et al., 2021). Computational resources were moved closer to the sources of information generation in EC, which reduced the network latency and bandwidth use frequently associated with cloud computing (CC) (Zhao et al., 2021). The edge server in an EC system serves and processes task requests and produces data from internet of things (IoT) devices nearby. The edge server's proximity to the IoT device reduced the time it took to respond to requests for work compared to the centralised cloud data centre (Huang et al., 2020).

Scheduling jobs to suitable edge servers for execution in accordance with performance and resource needs was a workable solution. An effective work scheduling algorithm could actually greatly increase users' quality of experience while simultaneously increasing the efficiency of resources (Cai et al., 2021). However, task scheduling using FIFO might reduce the throughput of EC nodes due to a mismatch between the job and the hosting node (Ullah and Youn, 2020). The major emphasis of CC was how to distribute incoming task requests among numerous resource units of a cloud cluster because there was no difference in scheduling costs while sending work to different resource units (Huang et al., 2020). The length of time was taken for running processes to complete is clearly impacted by an edge server's excessive resource contention. To handle scenarios where an edge server is significantly overloaded, an effective work or task scheduling mechanism must be presented (Liu et al., 2019).

To optimise the number of successful jobs, a deep quantum network (DQN)-based task scheduling method was investigated in CC (Sheng et al., 2021). Collaborative task scheduling beats historical scheduling methods in terms of the deadline fulfilment ratio of time-critical workloads while maintaining deadlines for local tasks in IoT devices (Liu et al., 2019). With the help of the Trans EC-DQL algorithm, scheduling decisions were made after learning the vector representations of the nodes and tasks. In addition, to improve scalability, the algorithm's state was changed to a distribution dependent on resource levels (Tang et al., 2022). Traditionally, the K-means clustering technique is used to categorise the work in the KTCS scheme, and the node usage is evaluated. After that, tasks are allocated to the node that matches the requested resource. While real EC scenarios involve many heterogeneous nodes and tasks, current DL methods presume an environment with comparatively limited state sets and action sets. Processing a lot of data is necessary for this (Lv et al., 2021; Mukherjee et al., 2021; Yamuna and Usha Rani, 2022). Additionally, each node or task has its own characteristics, such as resource capabilities for nodes and resource requests, allocations, etc. for tasks. These make it harder to store all the data immediately and can also cause dimensional disasters. Consequently, it is a significant problem to schedule the tasks, conserve computing resources, and arrive at quick decisions (Zhou et al., 2021; Fang et al., 2019; Liu et al., 2021). It fully utilises the processing power and computer resources of the peer-to-peer network, decomposes complicated computational tasks, and distributes them to the selected node for shared computing, greatly improving the utilisation efficiency of the available computational resources and workflow execution as a whole. In order to employ the other EC nodes to process the delay-sensitive tasks, it is critical to plan the jobs effectively in order to ensure the load balance of the entire EC network and to complete the task as quickly as practical. The main contributions are:

1 Proposing an optimisation-based task scheduling process in IoT-EC with the consideration of multi-objectives like makespan, execution time, execution cost, and risk probability.

2 Applying the deep convolutional neural network (D-CNN) concept is for estimating the risk probability while scheduling the tasks, which will be based on task security and virtual machine (VM) security.

3 Proposes a new optimisation technique, hybrid bald eagle Archimedes optimisation (HBEAO) for scheduling the task with the above-mentioned constraints.

The organisation of the paper is: The review of previous research is shown in Section 2, a system model is explained in Section 3, the proposed optimisation for scheduling the task with deep learning-based risk probability prediction is explained in Section 4, HBEAO based scheduling process is shown in Section 5, result is discussed in Section 6, and references are shown in Section 7.

2 Literature review

In 2021, Sheng et al. (2021) have developed a policy-based REINFORCE method to address the problem of work schedule, and a fully connected neural network (FCN) was utilised to extract the features. The given deep reinforcement learning (DRL)-based task scheduling method outperforms the currently recommended methods in the literature,

according to simulation results that show average task satisfaction levels and success rates.

In 2021, Cai et al. (2021) have developed a task scheduling technique for a failure-resistant directed acyclic graph (DAG) to reduce the reaction time that the tasks encounter. An approach called context-aware greedy task scheduling (CaGTS) was presented in order to solve the DAG task allocation problem. A dependency aware task rescheduling (DaTS) technique was subsequently developed to address the edge server failure occurrence. Extensive tests had been performed on a Python-developed simulator to gauge how well the proposed methods performed. According to experimental results with a variety of parameter settings and DaTR could successfully prevent task scheduling interruptions brought on by server failure events.

In 2021, Zhao et al. (2021) have created the low-load distributed intrusion detection system (DIDS) task scheduling method for reinforcement learning, which is based on the Q-Learning approach. By altering scheduling techniques dynamically in reaction to network changes in the EC environment, this approach can balance the two diametrically opposed signs of low load and packet loss rate. The aim was to maintain a minimum overall DIDS load. Indicators like the rate at which malicious features are detected are not greatly affected by the proposed strategy, simulation trials demonstrate, and it performs better under low demand than existing scheduling strategies.

In 2021, Zhang et al. (2021) have proposed a joint task scheduling and containerising (JTSC) scheme for task scheduling. The resource usage of container operations was initially measured through experiments. To reflect the properties of task execution in containers on a network edge with multiple processors, system models for the system were then developed. Initially, tasks were scheduled without taking containerisation into account, which led to earlier timetables. Second, a number of containerisation methods were created to map jobs to containers using the system concepts and principles gleaned from the first schedules. Third, the task schedules were adjusted to reflect the updated task execution durations, which included the time needed for inter-container interactions. Numerous simulations were used to assess the JTSC method. According to the results, it significantly decreased wasteful container activities and increased application execution efficiency by 60%.

In 2019, Liu et al. (2019) have developed fuzzy clustering technique is utilised in this algorithm to narrow the search space range, hence lowering the complexity of the scheduling process and the number of repetitions. Additionally, the ant colony algorithm's powerful global search capability is used to identify the scheduling problem's ideal solution.

In 2020, Tang et al. (2022) have deployed DRL-based method was designed to accommodate dynamic changes in nodes and tasks as well as to address the issues faced by DRL methods due to the large number dimensions, when the input size is large. To be more precise,

1 They use representation methods of learning to categorise various EC nodes and activities. Nodes and jobs are mapped to compatible vector sub-spaces in order to minimise dimensions and store vector data effectively.

2 Scheduling decisions were made utilising the space remaining after image compression to learn the associated with the suggested nodes and jobs.

3 Real-world data was used in the studies, and the findings demonstrate that the suggested representation technique using a DRL-based procedure outperforms the baselines by 18.04% and 9.94%, accordingly, in terms of energy usage and service level agreement violation (SLAV).

In 2021, Abd Elaziz et al. (2021) have developed AEOSSA, a new artificial ecosystem-based optimisation (AEO)-based alternative task scheduling method for IoT requests in a cloud-fog context. This patch was built utilising the SSA operators in an effort to increase the AEO's ability to exploit data while looking for the best solution to the issue at hand. The usefulness of the proposed AEOSSA approach to solve the task scheduling problem is evaluated using a variety of synthetic and real-world datasets of different sizes.

In 2020, Shi and Shi (2020) have established a multi-node task scheduling with a multi-objective optimisation that took into account the impact of energy use, load balancing, and job completion time. To fulfil the needs of delay-sensitive activities, task scheduling was turned into a bidding system and a real-time dumping ground for subtasks. Their final argument was that the method for scheduling tasks over several nodes, which offers fresh suggestions for allocating EC work, was created through simulation testing. Table 1 shows the benefits and drawbacks of task scheduling techniques.

Table 1 Features and challenges of previous task scheduling techniques

Author [citation]	*Methodology*	*Benefits*	*Drawbacks*
Sheng et al. (2021)	REINFORCE algorithm	Good convergence performance	It is necessary to focus more on communication delay
Cai et al. (2021)	CaGTS	In server failure events, it effectively performs the task scheduling	The impact is a communication link failure
Zhao et al. (2021)	DIDS task scheduling Q-learning algorithm	It has better low-load performance	The q-table is very large and the update time was increased
Zhang et al. (2021)	JTSC	It improves execution efficiency and decreases ineffective container activities.	Applications that cannot be categorised as task workflows
Liu et al. (2019)	ACO	Achieved near-optimal task throughput	It requires more storage capacity
Tang et al. (2022)	DRL	Energy consumption, SLAV, and the cost are low	It needs a lot of data and a lot of computation
Abd Elaziz et al. (2021)	SSA	Improves the task completion ratio	Must be improved with a more complex load-balancing scheme
Shi and Shi (2020)	Multi-objective optimisation model	Energy consumption was low and task completion time was faster	It is necessary to take into account the relationships between the activities and challenging arrival scenarios.

Despite the fact that numerous algorithms have been developed for task scheduling, different meta-heuristic algorithms, and deep learning concepts play a significant part in

task scheduling. However, the existing models have a variety of shortcomings in terms of performance efficiency. A very quick multi-objective optimisation model was used to address the issue of energy consumption efficiency (Shi and Shi, 2020). But it does not take into account how the tasks are interdependent, or how difficult arrival scenarios can be. The MAR model must also take complicated load balancing into account. Some deep learning models like CNN, LSTM, and Bi-GRU reduce costs, but they still require improved convergence toward decision-making for the given task. To address the above-mentioned constraints, the HBEAO optimisation method is used to improve the scheduling of tasks.

3 System model

The edge server receives jobs produced by IoT apps for scheduling tasks in an EC systems. Edge server configured with several VMs. To keep things simple, we merely pay attention to the computational resources for job scheduling. Task sizes, expected finish durations, processing rates (measured in MIPS), and waiting periods are just a few of the information the scheduler keeps track of regarding incoming tasks and VMs that have an impact on scheduled decisions. Based on the data collected, (i.e., which VM is assigned to each job), the scheduler chooses when to schedule, (i.e., the order of the tasks in the schedule and their start timings) and where to schedule. Figure 1 shows the architecture of task scheduling in EC. The backlog of tasks in the queue and the waiting set inside the circle are the two groupings of work that are available for scheduling. Unlike the number of jobs in the backlog queue, which can only be viewed by the scheduler, the number of tasks in each waiting set, which occupy waiting slots that can be completely noticed, can be seen by the scheduler. At each scheduling time step, the scheduler chooses a maximum of one work from the waiting slot for scheduling. This research investigates the work scheduling in EC with a single deployed edge server.

3.1 Task scheduling mechanism

Tasks and VMs are the two primary elements of EC and are also crucial to task scheduling. IoT applications generate the tasks, which are then forwarded to edge servers for processing. While EC owns the edge servers, which offer computational resources. Let $S = (S_1, S_2, \ldots. S_N)$ and $VM = (VM_1, VM_2, \ldots. VM_n)$ denote the sets of tasks and nodes respectively. Where S_1 is a first task and N denotes the total number of tasks, VM_1 denoted the first VM; n denotes the total number of VM.

The defined objective is to raise the overall task satisfaction over the long term, which is represented in equation (1). Here G_1, G_2, G_3, G_4 indicates the makespan, execution time, execution cost, and risk probability respectively, w_1, w_2, w_3, w_4 indicates the weights, that are assigned to each parameter G_1, G_2, G_3, G_4.

$$O = \min\left(w_1 * G_1 + w_2 * G_2 + w_3 * G_3 + w_4 * G_4\right) \tag{1}$$

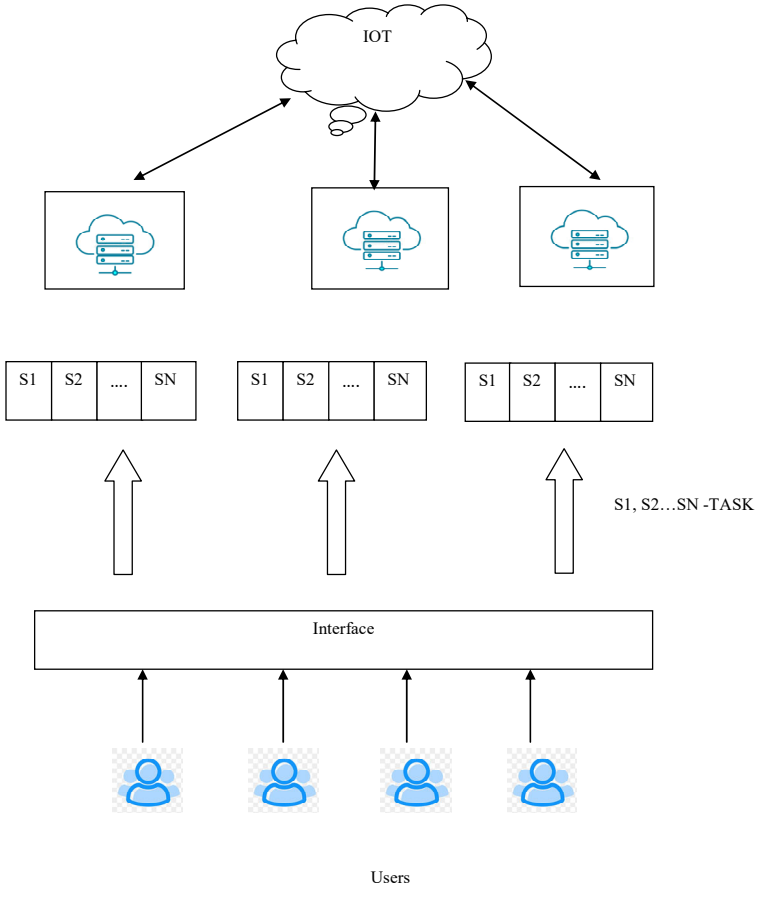
In equation (1), weights w are calculated using Dirichlet distribution. The Dirichlet distribution is defined as a distribution over vectors w satisfying the

constraints $w_i > 0 \sum_{i=1}^{n} w_i = 1$, where $n = 4$. The probability density function p of a Dirichlet-distributed random vector X is proportional to equation (2), where α is a vector containing the positive concentration parameters.

$$p(x) \propto \prod_{i=1}^{k} x_i^{\alpha_{i-1}} \tag{2}$$

The method uses the following property for computation: let Y be a random vector that has components that follow a standard gamma distribution, then $X = \frac{1}{\sum_{i=1}^{k} Y_i}$ is Dirichlet distributed.

Figure 1 Task scheduling architecture of EC (see online version for colours)



Task dependencies can be found by examining the connections between tasks and figuring out which ones need to be finished before beginning or ending. That is, you must take into account both your forebears and your successors. A start-to-finish (SF)

dependency occurs when you can only finish the prior task if the one after it has begun. One of the rarest kinds of dependencies in real-world settings, this kind of reliance typically arises from scheduling-related events involving the handoff of one task to another. Task dependencies show the sequence in which tasks must be finished. Dependencies allow you to find the optimal task sequencing that will get you through the research the quickest. While a VM is accessible, it is allocated a task; while it is busy, it is not assigned any tasks.

4 Proposed optimisation for scheduling the task with deep learning-based risk probability estimation

As stated above, the proposed task scheduling mechanism in EC considers parameters like makespan, execution time, execution cost, and risk probability. The risk probability can be predicted by using D-CNN. The defined parameters are as follows:

4.1 Makespan (G_1)

It is defined as the overall completion time needed to execute all tasks. Makespan (Mapetu et al., 2019) is represented in equation (3)

$$MP = \max_{1 \le i \le k} \{CT_i\} \tag{3}$$

where k indicates the number of VM. CT_i is represented in equation (4)

$$CT_i = \sum_{j=1}^{m} \frac{S_j.l}{VM_j.\mathrm{Pr} \times VM_j.mips} \tag{4}$$

where m denotes the number of tasks in VM, S denotes the task, l indicates the size of S, Pr indicates the number of processing elements in VM, $mips$ represents (million instruction per second) execution speed per processing element of a VM.

For example, the four jobs to process in parallel on two processors, and the processing times for each job are as follows: job 1: four units of time, job 2: three units of time, job 3: two units of time, job 4: five units of time. Using a scheduling algorithm, you might schedule the jobs as follows: processor 1: job 1 (four units), job 3 (two units), processor 2: job 2 (three units), job 4 (five units). The makespan in this case would be the time it takes for the last job to finish on processor 2: makespan = 3 units (job 2) + 5 units (job 4) = 8 units of time. So, the makespan for these parallel processing jobs on two processors is 8 units of time.

4.2 Execution time (G_2)

The duration of time taken by the VM to execute each task is denoted as execution time

4.3 Execution cost (G_3)

Execution costs are fees paid by the user to the provider in exchange for the usage of resources to carry out the task.

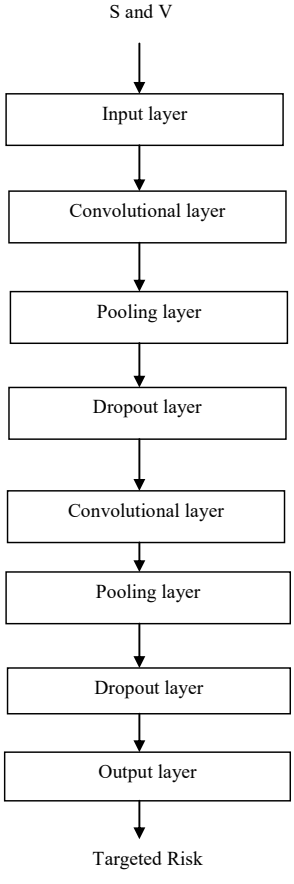
4.4 D-CNN-based risk probability estimation (G_4)

Risk probability (Verma, 2022; Li et al., 2021) can be predicted by using D-CNN, where the inputs are considered as task security (Hai et al., 2023) (S_s) and VM security (Hai et al., 2023) (V_s), based on which the model is trained to obtain the risk. A detailed explanation of the model is given below.

CNN is a deep feed-forward ANN class that is extensively utilised in computer vision issues like data classification. CNN differs from a 'simple' multilayer perception (MLP) network in that it employs convolutional layers, pooling, and nonlinearities like tanh, sigmoid, and ReLU (Teow, 2017).

The convolutional layer is made up of a filter that is used to 'slide' through the size of incoming data and generates task scheduling parameters. As a result, a 2-dimensional activation map made up of the filter's reactions at certain places will be created. The pooling layer then decreases the size of the data in accordance with the result of a convolution filter. This results in down-sampling, or scaling back of the parameterisation of the model. The proposed deep CNN is presented in Figure 2.

Figure 2 Layer-wise architecture of proposed deep CNN



4.4.1 Input layer

Task security (S_s) and VM security (V_s) are the inputs for deep CNN. S_s and V_s are generated randomly between the numbers 1 to 5.

4.4.2 Convolutional layer

When back propagation training is used to train the trainable convolution kernels in this layer, the kernel weights are automatically adjusted to take into account the features of the input data. The convolution layer's input features can be processed for use in other computational operations by the following algorithmic levels. This unsupervised convolutional feature learning method is neutrally motivated by the work on the recognition model and receptive field theory. The output is a convolved feature map (f_c) it was represented in equation (5) where ⊗ indicates a 2D discrete convolutional operator, convolutional kernel is represented in K spatially slides over the input data I to compute the element-wise multiplication and sum to produce an output, a convolved feature map (f_c).

$$\begin{aligned} f_c &= conv\left(T_s, V_s\right) \\ &= (I \otimes K)\left(S_s, V_s\right) = \sum_m \sum_n I(m,n) K\left(S_s - m, V_s - n\right) \end{aligned} \tag{5}$$

4.4.3 Pooling layer

In order to reduce the spatial dimensionality of the corrected feature map and produce a more compact feature representation for processing, a pooling layer subsamples the corrected feature map. (f_p) indicates the output of the pooled featured map. It is represented in equation (6).

$$f_p = pool\left(S_s, V_s\right) = \frac{1}{M} \sum_m x_{S_s, V_s} \tag{6}$$

4.4.4 Dropout layer

CNN has a lot of parameters, particularly in the fully connected layer, and having too many parameters can cause over fitting. Typically, a model combination made up of many distinct networks is trained to avoid over fitting. The fundamental principle of dropout is that at each training stage, a portion of units are randomly dropped with probability 1-p (or kept with probability p), and p can be established by experimentation. To put it another way, the networks are different from one another and become thinner than a typical neural network after dropout is used, which increases the model's resilience to over fitting and speeds up training. Each neuron within the combination of these thinning networks participates in prediction, and the model as a whole. Dropout hence can prevent overfitting.

4.4.5 Output layer

The output layer presents the risk probability result (P_{bj}) under varied condition r_k. This approach results in targeted risk, which has a range of 0 to 3. It is represented in

equation (7). If the output denotes 0, which means there is no risk, low risk if it is 1, high risk if it is 2, and medium risk if it is 3.

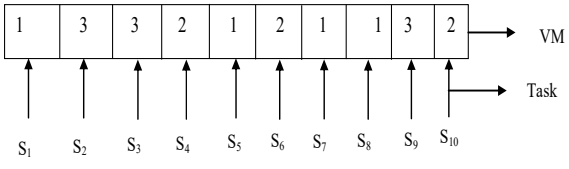
$$P_{bj} = \begin{cases} 0; & if\ r_k <= 0 \\ 1 - e^{\left((V_s - S_s)/2\right)}; & if\ r_k = 1 \\ 1 - e^{\frac{3(V_s - S_s)}{2}}; & if\ r_k = 2 \\ 1; & if\ r_k = 3 \end{cases} \tag{7}$$

5 HBEAO-based task scheduling process

5.1 Input solution and objective function

While scheduling the task using HBEAO, consider the inputs of the edge server have a certain number of VMs, and the number of tasks. Each task will be assigned to a particular VM for processing. For instance, then number of tasks SN is S_1, S_2, … S_N. Also, three numbers of VMs are used to assign the tasks. Based on the objective function in equation (1) (makespan, execution time, execution cost, and risk probability), task S_1 is assigned to the 1st VM, then task 2 is assigned to the 3rd VM, and so on till the10th task. Figure 3 shows the task scheduling process solution in HBEAO.

Figure 3 Task scheduling structure



The HBEAO method, or more particularly Archimedes' law, is based on physics. This algorithm's originality lies in the solution, which contains three pieces of auditory data for the core agents: volume (U), density (B), and acceleration (Gamma). As a result, in dim dimensions, the initial group of agents is generated at random (Hashim et al., 2021). As additive data, we provide random U, B, and Gamma values. The procedure of evaluating each object is then completed to determine which the best U_{best} is. To change the acceleration based on the idea of task collision throughout the HBEAO process, density and volume are updated. This step is crucial in identifying the innovative position of the present solution.

5.2 Initialisation

In this phase, we initialise the real population (according to this work, it is a task) of M objects using equation (8). Additionally, density (B_i), volume (U_i), and acceleration (Γ_i) of each object are constructed using the equations (9), (10), and (11). Where J_i indicates the i^{th} object, U_i^{Max} and U_i^{Min} are the search maximal spaces and minimal bounds, r_1, r_2, r_3 and r_4 are random numbers that range from $[0, 1]^{Dim}$.

$$J_i = J_i^{\min} + r_1 \times \left(J_i^{Max} - J_i^{Min} \right); i = 1, 2, ..., M \tag{8}$$

$$B_i = r_2 \tag{9}$$

$$U_i = r_3 \tag{10}$$

$$\Gamma_i = \Gamma_i^{Max} + r_4 \times \left(\Gamma_i^{Max} - \Gamma_i^{Min} \right); i = 1, 2, ..., M \tag{11}$$

To determine which object is the best (J_{best}) in the population, the best values of each candidate's density (B_{best}), volume (U_{best}), and acceleration (Γ_{best}) will be combined after a population analysis in which each candidate is given a score is completed.

5.3 Density and volume adjustments

At this stage, the density (Hashim et al., 2021) and volume (Hashim et al., 2021) values for each object are modified by selecting the optimal density and volume using equation (12) and equation (13). Initialise the volume and density for each i^{th} object.

$$J_i^{St+1} = J^{St} + s_1 \times \left(J_{Best} - J_i^{St} \right) \tag{12}$$

$$U_i^{It+1} = U_i^{It} + s_2 \times \left(U_{Best} - U_i^{It} \right) \tag{13}$$

where J_{best} and U_{best} are the volume and density for the optimal object. Here s_1 and s_2 indicates the random values between [0, 1].

5.4 Transfer coefficient and density scalar

Until the equilibrium state is established, this process involves object collisions. According to equation (14), the transition from the exploration phase to the exploitation phase is predominantly controlled by the transfer function (F_c). Over time, the (F_c) grows exponentially until it reaches 1. St indicates the current iteration; $St_{\max}$ indicates the maximum number of iterations. On the other hand, the global search employing equation (15) is expected to be converted to a local search by gradually decreasing the density factor bs.

$$F_c = \exp\left(\frac{St - St_{Max}}{St_{Max}} \right) \tag{14}$$

$$b_s^{St+1} = \exp\left(\frac{St - St_{Max}}{St_{Max}} \right) - \left(\frac{St}{St_{Max}} \right) \tag{15}$$

5.5 Proposed exploration phase

An assortment of materials chosen at random causes the agents to collide in this step. Consequently, the BES algorithm is used to update the acceleration objects when the transfer coefficient value is less than or equal to 0.5 (Alsattar et al., 2020). The BES algorithm, which replicates bald eagle hunting behaviour, justifies the results of each hunting step. The three parts of this method are picking the search space, looking inside the chosen search space, and swooping. It is shown by equation (16). Where $q(i)$ is

expressed in equation (17), $v(i)$ is expressed in equation (18). Where $qr(i)$ expressed in equation (19), $vr(i)$ expressed in equation (20), $\theta(i)$ expressed in equation (21) and $r(i)$ expressed in equation (22).

$$Z_{new} = Z_i + v(i) * (Z_i - Z_{i+1}) + q(i) * (Z_i - Z_{mean}) \tag{16}$$

$$q(i) = \frac{qr(i)}{\max(|qr|)} \tag{17}$$

$$v(i) = \frac{vr(i)}{\max(|vr|)} \tag{18}$$

$$qr(i) = r(i) * \sin(\theta(i)) \tag{19}$$

$$vr(i) = r(i) * \cos(\theta(i)) \tag{20}$$

$$\theta(i) = a * \pi * rand \tag{21}$$

$$r(i) = \theta(i) + R * rand \tag{22}$$

5.6 Proposed exploitation phase

As per the proposed logic, the trigonometric operator of the SCA algorithm is used for a position update. Here, the sine and cosine functions, as stated in equations (23) or (24), can be used to significantly improve the exploitation step of AOA by this operator. Where Z_{best} is the best object at St iteration. Z_i is the current solution, β_1, β_2, β_3 and β_4 are represented in equations (24) to (27). Where s is a constant value 2. Here $rand_1$, $rand_2$, $rand_3$ and $rand_4$ is calculated using the Chebyshev map,

$$\begin{aligned} Z_{new} &= Z_i + \beta_1 * \sin(\beta_2) * |\beta_3 . Z_{best} - Z_i| \; if \; \beta_4 < 0.5 \\ Z_{new} &= Z_i + \beta_1 * \cos(\beta_2) * |\beta_3 . Z_{best} - Z_i| \; if \; \beta_4 \geq 0.5 \end{aligned} \tag{23}$$

$$\beta_1 = s - t * \frac{s}{T_{\max}} \tag{24}$$

$$\beta_2 = 2 * Z_i * rand_2 \tag{25}$$

$$\beta_3 = 2 * rand_3 \tag{26}$$

$$\beta_4 = rand_4 \tag{27}$$

$$rand = \cos(0.5 * \cos^{-1} C_k) \tag{28}$$

5.7 The update process

For the exploration phase ($T_c \leq 0.5$), equation (29) modifies the position of the i^{th} object in iteration $St + 1$. Where F indicates the transfer operator which is updated by the BES swooping stage. It is represented in equation (30).

$$X_{new} = X_{best} + G * C_2 * rand * acc_{i-norm}^{++1} * d * \left(F * x_{best} - x_i^t\right) \tag{29}$$

$$F = \theta(i) + R * rand * \sinh[\theta(i)] \tag{30}$$

$$X_{new} = x_i^t + G * C_1 * rand * acc_{i-norm}^{++1} * d * \left(x_{rand} - x_i^t\right) \tag{31}$$

The pseudo-code of HBEAO is as follows:

Pseudo code of HBEAO
Step 1: Initialisation
Initialise the total number of populations, lower bound, upper bound, maximum iteration, chromosome length
Step 2: Initial solution
Generate population positions in a random manner
Compute densities and volume via equations (4), (5), and (6)
Step 3: Fitness function
Compute the fitness of each solution and choose the best fitness value
Step 4: Current iteration = 1
Step 5: While the current iteration is less than or equal to the maximum iteration
For each object
Update the density and volume of each object via equation (12) and equation (13)
Evaluate the transfer operator and density decreasing factor via equation (14) and equation (15)
If *transfer function* ≤ 0.5
Update the acceleration via the BES concept as per equation (16)
Update the positions of each object via the proposed equation (29)
Else
Update the acceleration via the BES concept as per equation (23)
Update the position of each object via equation (31)
End if
End for
Step 6: Compute the fitness of each updated object and select the best fitness
Step 7: Set iteration = iteration + 1
Step 8: End while
Step 9: Return the optimal best solution
Step 10: Stop

6 Results

6.1 Simulation procedure

Python was used to execute the suggested task scheduling in IoT EC. The HBEAO was compared with the various algorithms, namely, deep neural network (DNN) (Chen et al.,

2020), CHIMP, slime mould algorithm (SMA), Aquila optimisation (AO), tunicate swarm algorithm (TSA), Archimedes optimisation algorithm (AROA), K-means clustering-based task classification and scheduling (KTCS) (Ullah and Youn, 2020), and bald eagle search (BES), respectively. We can gather, visualise, and analyse real-time data streams in the cloud using a service provided by the IoT analytics platform called Thing Speak. Thing Speak instantaneously visualises the data we send from our devices to it. The network of interconnected devices and the technology that allows for communication both between the devices and the cloud as well as among them are collectively referred to as the 'internet of things', or IoT. Internet or networks are both referred to as 'clouds'. This technology replaces local drives with distant servers on the internet for online data storage, management, and access. There are many other types of data that can be used, including files, photos, documents, audio, and video.

6.2 Analysis of HBEAO for execution cost ($)

The computation of the execution cost of the HBEAO is analysed with conventional techniques, is represented in Figure 4 and Table 2. The HBEAO achieved the lowest execution cost of 37.27 whilst the DNN is 109.25, KTCS is 99.19, CHIMP is 76.61, SMA is 84.93, AO is 97.33, TSA is 50.24, AROA is 50.06, ACO is 88.06, AEOSSA is 63.89 and BES is 70.81, respectively. Furthermore, at the 20th VM, the HBEAO generated an execution cost of 37.39, even though the execution cost of 109.52, 98.81, 76.86, 85.45, 97.13, 50.32, 57.28, 87.83, 63.59 and 71.19, respectively. Generally, it is corroborated that the HBEAO works more flawlessly in the task of scheduling.

Figure 4 Estimation of execution cost of HBEAO and standard approaches (see online version for colours)

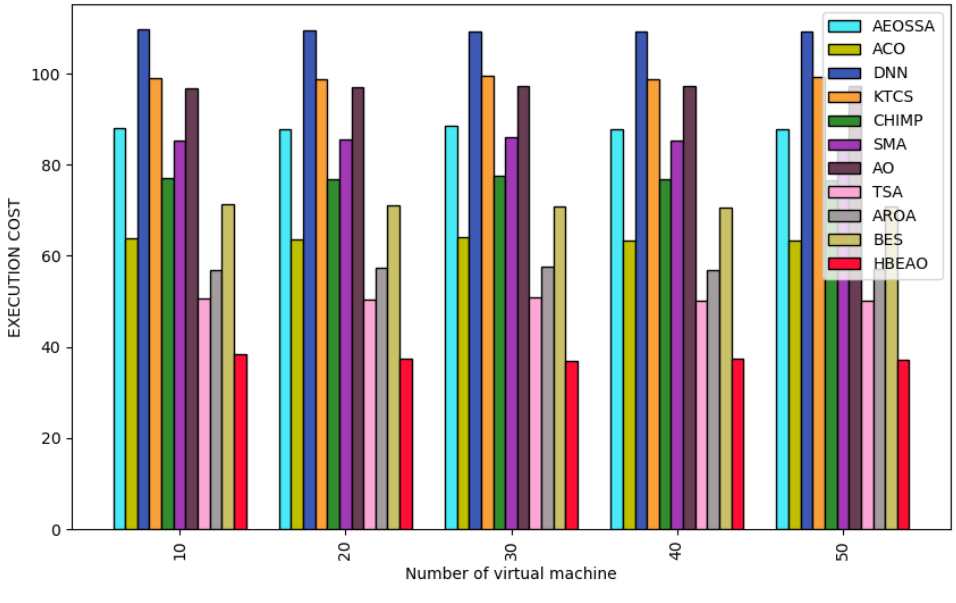


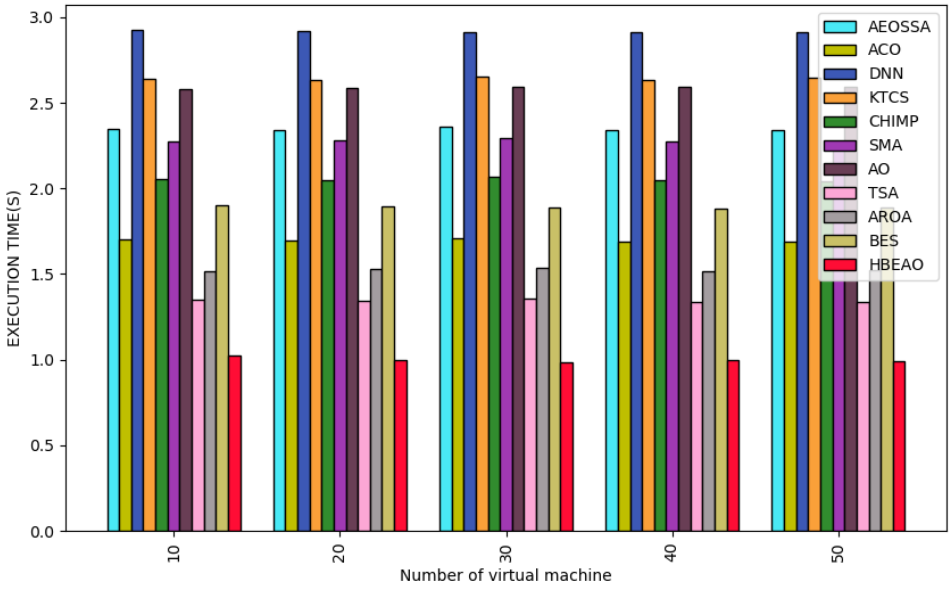
Table 2 Execution cost

Metrics	*DNN*	*KTCS*	*CHIMP*	*SMA*	*AO*	*TSA*	*AROA*	*BES*	*AEOSSA*	*ACO*	*HBEAO*
10	109.73	99.00	77.12	85.26	96.77	50.67	56.80	71.31	88.06	63.89	38.31
20	109.52	98.81	76.87	85.46	97.14	50.32	57.29	71.20	87.83	63.59	37.39
30	109.33	99.51	77.60	85.97	97.22	50.78	57.58	70.88	88.55	64.18	36.91
40	109.31	98.81	76.83	85.17	97.25	50.07	56.99	70.69	87.82	63.45	37.53
50	109.26	99.19	76.61	84.93	97.33	50.24	57.06	70.81	87.90	63.42	37.27

6.3 Analysis of execution time during task scheduling process of HBEAO and other methods

An execution time analysis is reviewed and computed over the standard approaches to demonstrate the viability of the HBEAO. Also, Figure 5 and Table 3 present the pertinent results. The minimal execution time obtained in the HBEAO is 0.99 towards the 50th VM, in contrast, the DNN = 2.91, KTCS = 2.64, CHIMP = 2.04, SMA = 2.26, AO = 2.59, TSA = 1.34, AROA = 1.5, AEOSSA = 2.34, ACO = 1.70 and BES = 1.89, respectively. Likewise, the HBEAO maintained 1.02, 0.1, 0.9, and 1.0 execution times while scheduling the tasks in 10, 20, 30, and 40 VMs. Therefore, the HBEAO provides progressive performances for task scheduling with minimal execution time.

Figure 5 Estimation of execution time of HBEAO and standard approaches during the scheduling process (see online version for colours)



6.4 Analysis of HBEAO-based task scheduling with respect to an objective

The HBEAO is conflicted with previous methodologies for numerous VMs to fitness. The findings for the fitness measure are portrayed in Figure 6 and Table 4. For the 30th

VM, the HBEAO scored a fitness of 6.01, although the current methods have obtained a fitness of 49.53, 75.38, 20.90, 50.06, 85.34, 22.75, 59.04, 45.05, 31.58 and 20.65, respectively. This proves that the proposed method obtains minimal fitness that ensures better convergence in scheduling the tasks. The HBEAO achieved diminished fitness to 39.48, meanwhile, the DNN is 88.97, KTCS is 42.55, CHIMP is 34.14, SMA is 45.74, AO is 10.48, TSA is 28.67, AROA is 53.39, AEOSSA is 51.29, ACO is 46.34 and BES is 32.13, respectively when it experiments with 50 VM.

Table 3 Execution time (seconds)

Metrics	*DNN*	*KTCS*	*CHIMP*	*SMA*	*AO*	*TSA*	*AROA*	*BES*	*AEOSSA*	*ACO*	*HBEAO*
10	2.92	2.64	2.06	2.27	2.58	1.35	1.51	1.9	2.34	1.70	1.02
20	2.92	2.63	2.05	2.28	2.59	1.34	1.53	1.90	2.34	1.69	0.12
30	2.92	2.65	2.07	2.29	2.59	1.35	1.54	1.89	2.36	1.71	0.987
40	2.91	2.63	2.05	2.27	2.59	1.34	1.52	1.89	2.34	1.69	1.00
50	2.91	2.65	2.04	2.26	2.60	1.33	1.52	1.89	2.34	1.69	0.99

Figure 6 Estimation of fitness of HBEAO and standard approaches (see online version for colours)

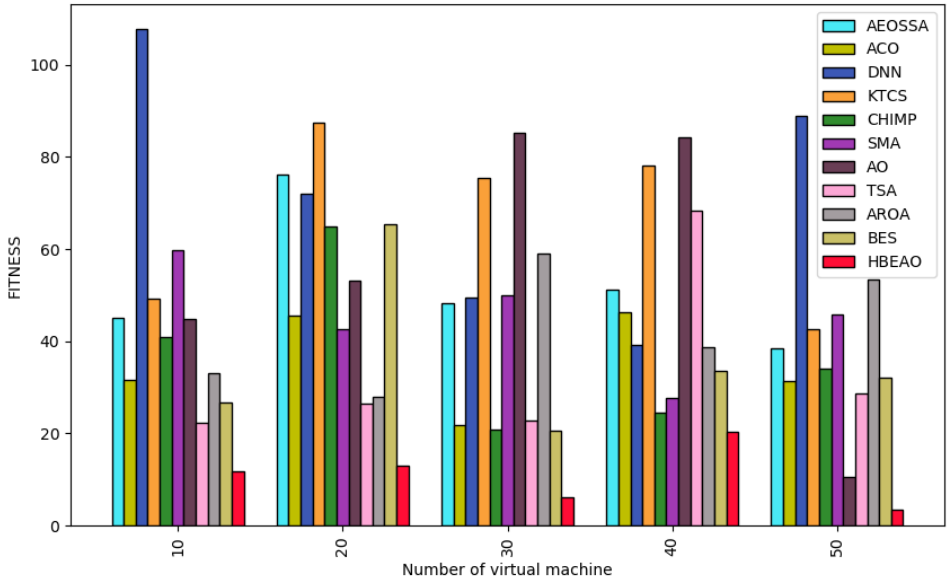


Table 4 Fitness

Metrics	*DNN*	*KTCS*	*CHIMP*	*SMA*	*AO*	*TSA*	*AROA*	*BES*	*AEOSSA*	*ACO*	*HBEAO*
10	107.77	49.21	40.90	59.85	44.86	22.28	33.15	26.78	45.05	31.58	11.65
20	72.13	87.54	64.99	42.77	53.22	26.35	27.90	65.40	76.26	45.67	12.99
30	49.54	75.39	20.90	50.06	85.34	22.75	59.04	20.65	48.14	21.82	6.016
40	39.14	78.16	24.44	27.56	84.34	68.25	38.59	33.61	51.29	46.34	20.42
50	88.97	42.55	34.14	45.74	10.47	28.67	53.39	32.13	38.34	31.40	3.48

6.5 Analysis of HBEAO-based task scheduling with respect to makespan

The resemblance of the HBEAO makespan to those of the established methods for the distinctive VM is shown in Figure 7 and Table 5. The maximal makespan attained by the KTCS in the 10th VM is 144.05, accompanied by a DNN is 127.6921 and AO is 110.9954, however, the HBEAO acquired the makespan of 37.57. Furthermore, the HBEAO gained the makespan of 36.16, which is superior to DNN = 127.58, KTCS = 144.02, CHIMP = 90.39, SMA = 106.19, AO = 111.40, TSA = 62.05, AROA = 70.18, AEOSSA = 117.29, ACO = 76.29, and BES = 90.83, respectively. Consequently, the findings imply that the HBEAO has outperformed other alternative plans by a significant margin with less makespan for task scheduling in IoT EC.

Figure 7 Estimation of makespan of HBEAO and standard approaches (see online version for colours)

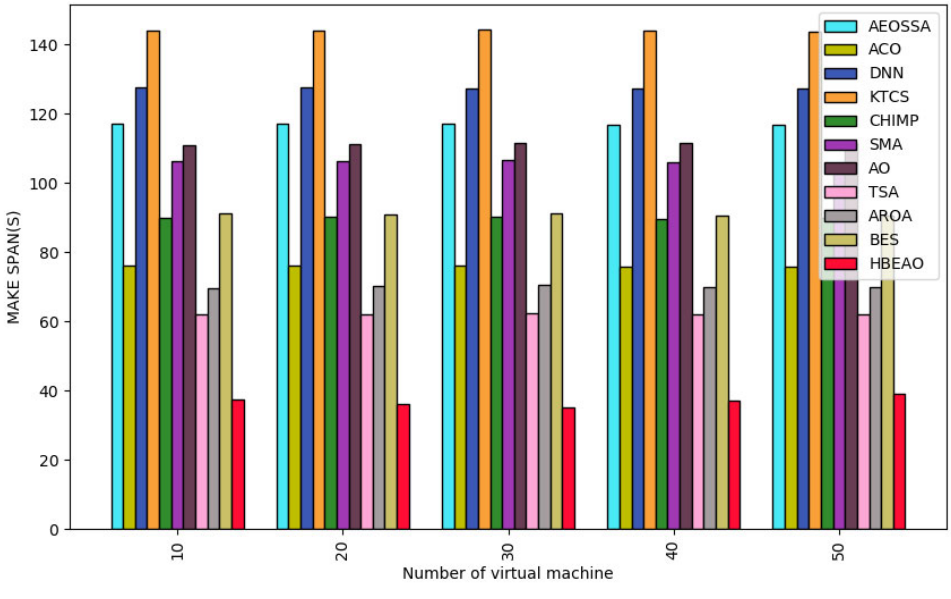


Table 5 Makespan

Metrics	*DNN*	*KTCS*	*CHIMP*	*SMA*	*AO*	*TSA*	*AROA*	*BES*	*AEOSSA*	*ACO*	*HBEAO*
10	127.69	144.05	90.09	106.25	110.99	62.09	69.57	91.39	117.20	76.21	37.57
20	127.58	144.02	90.39	106.19	111.40	62.05	70.18	90.83	117.29	76.29	36.15
30	127.50	144.43	90.16	106.75	111.49	62.43	70.59	91.10	117.20	76.21	35.15
40	127.44	144.24	89.70	105.97	111.70	61.90	69.91	90.75	116.96	75.80	36.96
50	127.45	143.92	89.81	105.51	111.63	61.93	69.85	90.63	116.86	75.869	38.97

6.6 Analysis of HBEAO-based task scheduling with respect to risk probability

To substantiate the feasibility of the HBEAO for task scheduling in IoT EC persistence over the traditional methods are described in Figure 8 and Table 6. The risk probability of the HBEAO for the 40th VM is 0.17, even though the DNN is 0.29, KTCS is 0.34, CHIMP is 0.32, SMA is 0.32, AO is 0.35, TSA is 0.33, AROA is 0.33, AEOSSA is 51.29, ACO is 46.34 and BES is 0.32, respectively. The HBEAO provides impressive results with lower risks because its performance is more consistent than that of other traditional algorithms. By considering the optimal parameters of optimisation-based task scheduling, the HBEAO optimisation technique is used to improve the scheduling of tasks. The task scheduling of EC performs with lesser time, less makespan, and minimum amount of execution cost by considering the proposed optimisation method. The validation of present field of research has focused on using cutting-edge optimisation and deep learning approaches to help with task scheduling. The proposed method produces better results in task scheduling in terms of optimal parameters.

Figure 8 Estimation of risk probability of HBEAO and standard approaches (see online version for colours)

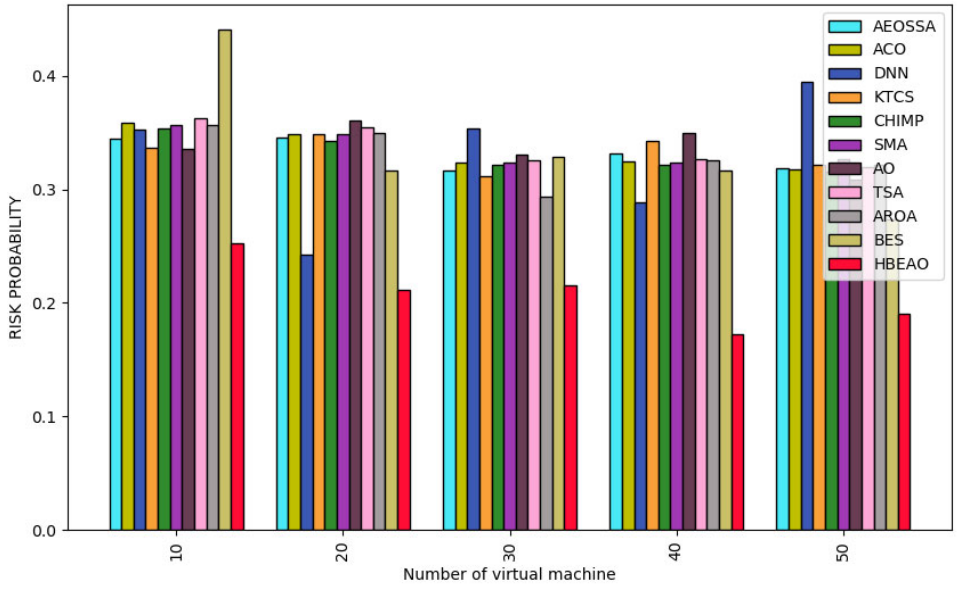


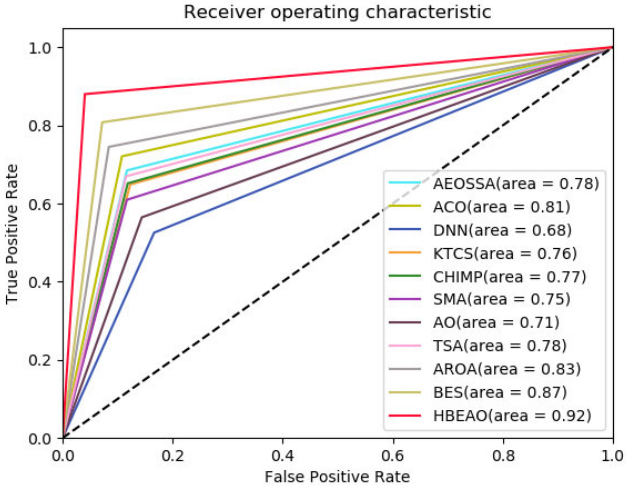
Table 6 Risk probability (%)

Metrics	*DNN*	*KTCS*	*CHIMP*	*SMA*	*AO*	*TSA*	*AROA*	*BES*	*AEOSSA*	*ACO*	*HBEAO*
10	0.35	0.34	0.35	0.36	0.34	0.36	0.36	0.44	0.34	0.34	0.25
20	0.24	0.35	0.34	0.35	0.36	0.36	0.35	0.32	0.31	0.32	0.21
30	0.35	0.31	0.32	0.32	0.33	0.33	0.30	0.33	0.33	0.32	0.21
40	0.29	0.34	0.32	0.32	0.35	0.33	0.33	0.32	0.31	0.31	0.17
50	0.39	0.32	0.31	0.33	0.31	0.32	0.32	0.27	0.34	0.35	0.19

6.7 Convergence analysis on HBEAO-based task scheduling

The convergence assessment was carried out to determine the performance of the HBEAO in comparison to other methods in terms of minimising multi-objective function. The outcomes are depicted in Figure 9. During the initial iteration, the HBEAO acquired the convergence rate of 11.2, whereas the TSA is 11.3, SMA is 12.4, CHIMP is 12.6, BES is 12.5, AEOSS is 12.7, ACO is 12.4 and AROA is 16.2, respectively. However, the HBEAO still has the lowest convergence value of 16.2 at the final iteration. As a consequence, it can be interpreted that the enhancement in the HBEAO approach minimises the makespan, execution cost, execution time, and risk probability and it applicable for task scheduling in IoT EC.

Figure 9 Convergence study on HBEAO and conventional methodologies (see online version for colours)



6.8 Statistical analysis

Despite its random nature, the method of optimisation is exposed to several runs in order to ascertain the final results in terms of statistical metrics. The proposed statistical analysis for the HBEAO system is compared with more conventional approaches in Table 7. Five different case studies were used to evaluate the minimum, the maximum, the mean, the median, and the standard deviation. The proposed model has a mean of 84.9%, which is better than the average for standard methods which are 10.41%, 9.88%, 9.69%, 8.87%, 9.30%, and 9.50%, respectively. The HBEAO approach yields the least mean value. The CHIMP approach yields the cost function of 8.58 %, followed by the BES method at 8.37% and the AROA method at 8.31%, while the proposed HBEAO model achieved the minimum cost function of 8.18%. Similarly, the proposed HBEAO model's median is 8.33%. The maximum values of the existing methods and proposed technique are 11.6%, 8.48%, 10.55%, 0.80%, 43.83%, and 11.6%. Consequently, the statistical analysis demonstrates the suggested algorithm's hopeful performance in choosing the best attributes for accurate task scheduling.

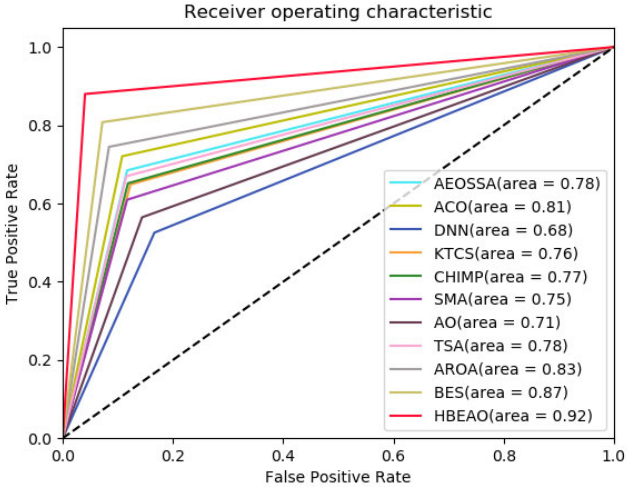
6.9 ROC-AUC analysis

Figure 10 displays the analysis of the ROC and AUC curves. The AUC-ROC curve is an efficacy measure for problems with classification at various threshold values. The y-axis analysis shows the genuine positive rate, and the X-axis analysis shows the false positive level. The AUC obtained, 0.86, is almost 1. In comparison to some of the past efforts, the value of AUC is accessed. It is evident that between the upper-left corner and the curve diagonal, the test data cover a bigger area than the training dataset.

Table 7 Statistical analysis of cost function

Metrics	*Mean*	*Median*	*Standard deviation*	*Min*	*Max*
CHIMP	10.41	11.21	1.58	8.58	12.50
SMA	9.88	9.48	1.47	8.51	12.15
AO	9.69	8.45	1.53	8.45	12.38
TSA	8.87	8.57	0.89	8.48	11.29
AROA	9.30	8.31	1.76	8.31	16.11
BES	9.50	8.69	1.06	8.37	12.51
ACO	10.54	10.39	1.62	9.03	13.15
AEOSSA	11.22	11.95	1.78	8.69	12.94
ACO	10.54	8.45	9.48	8.31	11.39
HBEAO	8.49	8.33	0.81	8.18	11.20

Figure 10 FROC and AUC curve analysis (see online version for colours)



6.10 Time complexity analysis

Table 8 displays the time complexity of the proposed HBEAO method over existing models. The existing Chimp optimisation (CHIMP), SMA, AO, TSA, AROA, BES methods are inferior by 34%, 72%, 24%, 24%, 19%, 22%, 12% and 6% respectively compared to the proposed HBEAO model's computational time.

Table 8 Computational time analysis

Methods	*Computation time (seconds)*
CHIMP	3,137.204
SMA	4,016.772
AO	2,899.112
TSA	2,891.345
AROA	2,777.992
BES	2,483.078
AEOSSA	2,798.081
ACO	2,435.123
HBEAO	2,325.872

6.11 Discussion

The IoT-EC concept is used in this research to analyse the task scheduling problem. By optimising long-term work satisfaction, a variety of jobs are scheduled to VMs deployed at the edge server. Considerations for the suggested ideal job scheduling include makespan, execution time, execution cost, and risk likelihood. In particular, the D-CNN performs the risk probability estimation. Based on task security and VM security, the novel HBEAO is used to schedule activities. The HBEAO achieved the lowest execution time of 0.99 for the 50th VM of all achieved execution times of 2.91, 2.59, 1.34, 1.5, 2.30, 2.10, 2.12, 2.17 and 1.89, respectively. Finally, the effectiveness of the projected job is compared to the existing methods to task scheduling. The efficacy of the proposed work is then contrasted with that of the conventional task scheduling methods.

7 Conclusions

As the IoT gains traction, data collection is rapidly expanding. Jobs should be transferred to edge servers in order to offer quick response and balanced loads because end devices are computationally limited. The uneven distribution of devices in the actual environment would cause hot spots to form. There may be too many devices connected to some servers, which could cause jobs to run over their scheduled completion times. The proposed approach for task scheduling using deep learning-based risk probability prediction is offered as a solution to this issue. EC's ideal job scheduling takes into account variables like risk likelihood, execution cost, execution time, and makespan. Risk probability was assessed using DCNN. The assignment was finally scheduled using the HBEAO. The created model has a fitness score of 3.484123, which is 96.08% better than DNN, 91.811% better than KTCS, 89.79% better than Chimp, 92.38% better than SMA, 66.73% better than AO, 87.84% better than TSA, 93.47% better than AROA, AEOSSA is 51.29, ACO is 46.34 and 89.15% better than BES. The end result demonstrates that the suggested method performs better than alternative benchmark methods. The focus of recent research has been on using cutting-edge optimisation and deep learning approaches to help with task scheduling. Future research will concentrate on workload

prediction, such as overload, underload, or balanced load, with enhanced algorithms to improve job scheduling.

References

Abd Elaziz, M., Abualigah, L. and Attiya, I. (2021) 'Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments', *Future Generation Computer Systems*, Vol. 124, pp.142–154.

Alsattar, H.A., Zaidan, A.A. and Zaidan, B.B. (2020) 'Novel meta-heuristic bald eagle search optimisation algorithm', *Artif. Intell. Rev.*, Vol. 53, pp.2237–2264, https://doi.org/10.1007/s10462-019-09732-5.

Cai, L., Wei, X., Xing, C., Zou, X., Zhang, G. and Wang, X. (2021) 'Failure-resilient DAG task scheduling in edge computing', *Computer Networks*, Vol. 198.

Chen, Z., Hu, J., Chen, X., Hu, J., Zheng, X. and Min, G. (2020) 'Computation offloading and task scheduling for DNN-based applications in cloud-edge computing', in *IEEE Access*, Vol. 8, pp.115537–115547, DOI: 10.1109/ACCESS.2020.3004509.

Fang, J., Li, K. and Ma, A. (2019) 'Latency aware online tasks scheduling policy for edge computing system', *Journal of Physics: Conference Series*, Vol. 1325, No. 1, IOP Publishing.

Gezer, V. and Wagner, A. (2021) 'Real-time edge framework (RTEF): task scheduling and realisation', *J. Intell. Manuf.*, Vol. 32, pp.2301–2317, https://doi.org/10.1007/s10845-021-01760-9.

Hai, T. et al. (2023) 'Task scheduling in cloud environment: optimization, security prioritization and processor selection schemes', *Journal of Cloud Computing*, Vol. 12, No. 1, p.15.

Hashim, F.A., Hussain, K., Houssein, E.H. et al. (2021) 'Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems'. *Appl. Intell.*, Vol. 51, pp.1531–1551, https://doi.org/10.1007/s10489-020-01893-z

Huang, J., Li, S. and Chen, Y. (2020) 'Revenue-optimal task scheduling and resource management for IoT batch jobs in mobile edge computing', *Peer-to-Peer Netw. Appl.*, Vol. 13, pp.1776–1787, https://doi.org/10.1007/s12083-020-00880-y.

Li, Z., Xu, W., Shi, H., Zhang, Y. and Yan, Y. (2021) 'Security and privacy risk assessment of energy big data in cloud environment', *Computational Intelligence and Neuroscience*, Vol. 2021, p.11.

Liu, J. et al. (2019) 'An ant colony optimization fuzzy clustering task scheduling algorithm in mobile edge computing', *Security and Privacy in New Computing Environments: Second EAI International Conference, SPNCE 2019*, Tianjin, China, 13–14 April, Proceedings 2, Springer International Publishing.

Liu, P., Lyu, S., Ma, S. and Wang, W. (2021) 'Optimization algorithm of wireless surveillance data transmission task based on edge computing', *Computer Communications*, Vol. 178, pp.14–25.

Lv, Z., Chen, D., Lou, R. and Wang, Q. (2021) 'Intelligent edge computing based on machine learning for smart city', *Future Generation Computer Systems*, Vol. 115, pp.90–99.

Mapetu, J.P.B., Chen, Z. and Kong, L. (2019) 'Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing', *Applied Intelligence*, Vol. 49, pp.3308–3330.

Mukherjee, D., Nandy, S., Mohan, S., Al-Otaibi, Y.D. and Alnumay, W.S. (2021) 'Sustainable task scheduling strategy in cloudlets', *Sustainable Computing: Informatics and Systems*, Vol. 30, pp.1–10.

Sheng, S., Chen, P., Chen, Z., Wu, L. and Yao, Y. (2021) 'Deep reinforcement learning-based task scheduling in IoT edge computing', *Sensors*, Vol. 21, No. 5, pp.1–19.

Shi, Z. and Shi, Z. (2020) 'Multi-node task scheduling algorithm for edge computing based on multi-objective optimization', *Journal of Physics: Conference Series*, Vol. 1607, No. 1, IOP Publishing.

Tang, Z., Jia, W., Zhou, X., Yang, W. and You, Y. (2022) 'Representation and reinforcement learning for task scheduling in edge computing', in *IEEE Transactions on Big Data*, 1 June, Vol. 8, No. 3, pp.795–808, DOI: 10.1109/TBDATA.2020.2990558.

Teow, M.Y. (2017) 'Understanding convolutional neural networks using a minimal model for handwritten digit recognition', in *2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, IEEE, pp.167–172.

Ullah, I. and Youn, H.Y. (2020) 'Task classification and scheduling based on K-means clustering for edge computing', *Wireless Pers Commun*., Vol. 113, pp.2611–2624, https://doi.org/10.1007/s11277-020-07343-w.

Verma, G. (2022) 'Secure VM migration in cloud: multi-criteria perspective with improved optimization model', *Wireless Personal Communications*, May, Vol. 124, No. 5, pp.1–28.

Yamuna, R. and Usha Rani, M. (2022) 'Priority based task scheduling and delay optimization in mobile edge computing', *International Journal of Computer Engineering in Research Trends*, Vol. 9, No. 1, pp.1–6.

Zhang, J., Zhou, X., Ge, T., Wang, X. and Hwang, T. (2021) 'Joint task scheduling and containerizing for efficient edge computing', in *IEEE Transactions on Parallel and Distributed Systems*, 1 August, Vol. 32, No. 8, pp.2086–2100, DOI: 10.1109/TPDS.2021.3059447.

Zhao, X., Huang, G., Gao, L., Li, M. and Gao, Q. (2021) 'Low load DIDS task scheduling based on Q-learning in edge computing environment', *Journal of Network and Computer Applications*, Vol. 188.

Zhou, J., Fan, J. and Wang, J. (2021) 'Task scheduling for mobile edge computing enabled crowd sensing applications', *International Journal of Sensor Networks*, Vol. 35, No. 2, pp.88–98.