*International Journal on Recent and Innovation Trends in Computing and Communication*
***ISSN: 2321-8169 Volume: 11 Issue: 9***
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

# Self-Improved Pelican optimization for Task Scheduling in Edge Computing: Neural Network based risk probability estimation

**Bantupalli NagaLakshmi[1] , Dr. Sumathy Subramanian[2]**
[1]School of Computer Science and Engineering,Vellore Institute of Technology,Vellore,India
[2]School of Information Technology and Engineering ,Vellore Institute of Technology,Vellore,India
[*]Corresponding Author E-Mail : ssumathy@vit.ac.in , b.nagalakshmi2016@vitstudent.ac.in.

**Abstract**— The aim of edge computing is to drastically speed up the task response time while using less energy. The computational resources in EC are situated in closer proximity to the information generation sources, resulting in lower network latency and bandwidth utilization related to cloud computing (CC). In an EC system, the edge server handles and manages the requests of task and generated information from adjacent IoT machines. The task's schedule is regarded as the optimization problem. Thus, this paper aims to provide a novel task scheduling model that considers Risk probability, Execution cost, Execution time, and Makespan in to account. The Neural Network specifically estimates risk probabilities while taking task security and virtual machine security into consideration. This work suggests a new Paetro distribution-based pelican optimization algorithm (PDPOA) model for optimum scheduling of tasks. Results from the proposed system are examined and compared to existing methods via certain measures including Makespan, Execution time, Execution cost, Risk probability, etc.

**Keywords- *Task Scheduling*; *Edge Computing*; *Neural Network*; *Risk Probability*; *PDPOA Model.***

**Nomenclature**

| Abbreviation | Description |
|---|---|
| CC | Cloud Computing |
| PDPOA | Patero Distribution Based Pelican Optimization Algorithm |
| AR | Augmented Reality |
| FIFO | First-In-First-Out |
| BES | Bald Eagle Search Optimization |
| DPR | Dynamic Partial Reconfiguration |
| CaGTS | Context-Aware Greedy Task Scheduling |
| DaTR | Dependency-Aware Task Rescheduling |
| EC | Edge Computing |
| JTSC | Joint Task Scheduling And Containerizing |
| VMs | Virtual Machines |
| POA | Pelican Optimization Algorithm |
| SSA | Sparrow Search Algorithm |
| BOA | Butterfly Optimization Algorithm |
| KTCS | K-Means Clustering-Based Task Classification And Scheduling |
| HHO | Harris Hawks Optimizer |
| HBA | Honey Badger Algorithm |

## I. Introduction

A novel computer paradigm called EC extends cloud functionality to the network edge. Through edge servers, it offers customers computing resources, such as CPUs, memory, storage, etc. Edge computing has various benefits over cloud computing, such as strong user data privacy, reduced bandwidth requirements for communication networks, and low service-access latency [1]. As a result, edge computing has shown to be effective in many common applications, including smart homes, video analysis, intelligent transportation, augmented reality, etc. Moreover, an edge server must handle a variety of functions for an application, for example, a smart-home application like device management, sensor data analysis, video processing, object identification, etc. These tasks rely on diverse software elements (packages, libraries, etc.), some of which may clash with one another [2] [3]. For instance, two tasks may require the same component in two distinct versions. Virtualization is frequently used to create separated environments for work in order to prevent these software-dependency issues. Additionally, virtualization can improve system robustness by shielding the system from failure due to the failure of a single job. Containers are more portable than standard virtual machines, making them suitable for edge servers with minimal resources. Containers have been used by several ECmodel including Azure IoT Edge from Microsoft, Greengrass from Amazon, etc. Research work [4] has shown the viability of containers in EC.

The distributed platform is made up of heterogeneous computer nodes by EC. Here, the tasks are categorized as I/O-intensive, CPU-intensive, or Communication-intensive based on the amount of resources they demand [5]. Diverse processes in the distributed computing environment are conducted concurrently on the nodes such as EC and cloud. When the workload of task managers differs depending on the kind of conservation of the reservoir, it is crucial because of the feasibility of the present reservoir and the state of the load on the nodes [6] [7]. The jobs are often scheduled using the FIFO principle by task managers. However, FIFO task scheduling may cause the EC nodes' performance to decrease because of the discrepancy between the job and the hosting node [8]. Task scheduling and resource management have been prominent issues in both the academic and industrial communities ever since the dawn of CC [9]. The main focus is on allocating incoming job requests among the several resource units in a cloud cluster; scheduling costs do not change when workloads are sent to other resource units. However, in the context of EC, the job scheduling and resource management problem appears different. The challenge of job scheduling & resource management appears to be different in the framework of EC.

Edge tasks [10]are generated dynamically When the client needs a service then the server of edge offers many procedures that generate one request from the service. This problem is considered  to be the optimization problem, and this paper aims to propose a self-improved optimization for solving the scheduling process of task.

- ✓ The risk probability estimation is carried out by the Neural Network that considers the VMsecurity and task security.
- ✓ Implements a Patero distribution-based pelican optimization algorithm (PDPOA) modelforoptimal scheduling of tasks.

Section II determines a review on scheduling of task in edge EC. The framework of optimal scheduling of task in EC is discussed in Section III. Sections IV &V explains the results& conclusions.

## II. LITERATURE REVIEW

### A. *Related works*

In 2020, Ullah *et al.* [11] has established a unique technique known as KTCS that, prior to being sent to the edge node, classifies the job depending on the kind of resource demand based onI/O, COMM, or CPU. The suggested approach effectively schedules and allocates the job to maximize the usage of the edge devices with the waiting lines of M/M/c by modeling the K-means model. Finally, the outcomes shown the suggested model considerably increases the efficiency of resource use & job execution time for edge nodes.

In 2019, Zhu *et al.* [12] has implemented a task-scheduling scheme on the DPR platform. They fully account for the task switching overhead & predictable hardware task execution durations in DPR, as well as cut down on the number of active tasks and task switching times to increase scheduling effectiveness. According to experimental findings, all situations have efficiencies > 98% and approach 90%-98% in execution time surpasses the cost of reconfiguration by a factor of two.

In 2021, Ling *et al.* [13] has proposed an approach for scheduling DAG activities that can be failure-resistant to cut down on the reaction time of the tasks experience. A CaGTS approach was then presented after computing the DAG task scheduling problem. Subsequently a DaTR method has been created to handle the edge server failure event. The effectiveness of the offered methods has been evaluated by a series of thorough experiments on a Python simulator. Owing to testing results with various parameter settings, DaTR can effectively stop server failure-related disruptions in job scheduling and CaGTS may lower finished times by at least 10.47% when evaluated to standards.

In 2021, Xu *et al.* [14] has developed a lesser load DIDS task scheduling approach depending upon the Q-Learning model in reinforcement models that could alter scheduling models in response to network alteration in the EC environment to maintain a low total load of DIDS, it carries on equilibrium between the two opposite parameters of least load and packet loss value. Simulations tests confirm that the recommended technique performs superior than other scheduling strategies according to low demand, and that indicators like the speed at which harmful characteristics occur cannot be significantly impacted.

In 2021, Zhang *et al.* [15] has developed a JTSC system in this article. To measure the usage of energy of container operations, experiments were conducted. The characteristics of execution of task in containers on an server of edge with many CPUs are then captured by system models. Without taking containerization into account, tasks have been initially planned, leading to initial schedules. Second, a number of containerization algorithms are created depending on system methods and principles revealed from the original schedules. According to the findings, it significantly improves application execution efficiency by reducing wasteful container operations by 60%.

## III. FRAMEWORK OF OPTIMAL TASK SCHEDULING MODEL IN EC

The aim of this study is to offer an optimal task scheduling model while taking Execution Cost, Risk Probability, Execution Time, and Makespan into account. In particular, the NN estimates risk probability while taking task security and virtual machine security into consideration. This paper suggests a novel PDPOA paradigm for efficient scheduling of tasks.

### A. *System Model*

IoT applications that schedule tasks are provided to edge servers in an edge computing architecture. Multiple VMs are setup on the edge server. They restrict the target to the computation resources for task scheduling to make it easier. The scheduler makes the various constraints based on virtual machines (VMs) and approaching tasks that impact the resolution for scheduling like the size of the task, time period

***Article Received:*** *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

for completing the project, the speed for processing (measured in MIPS), and time for waiting. The scheduler makes decisions about when and where to make the process based on the inspection. Fig 1 displays the framework of EC scheduling of tasks.

The initial set of jobs is an excess item in the queue, but the second collection is a waiting time in the circle. Here, the availability of the schedulers is in the two groups of jobs. The scheduler knows the number of tasks in every waiting time that cover the whole noticeable solts for waiting; similarly, the scheduler is conscious about the number of jobs in the backlog queue. In every time scheduling procedure, the scheduler selects an increase of jobs to be organized from the waiting period. The task scheduling in EC is developed in one edge server that is explored in this work.

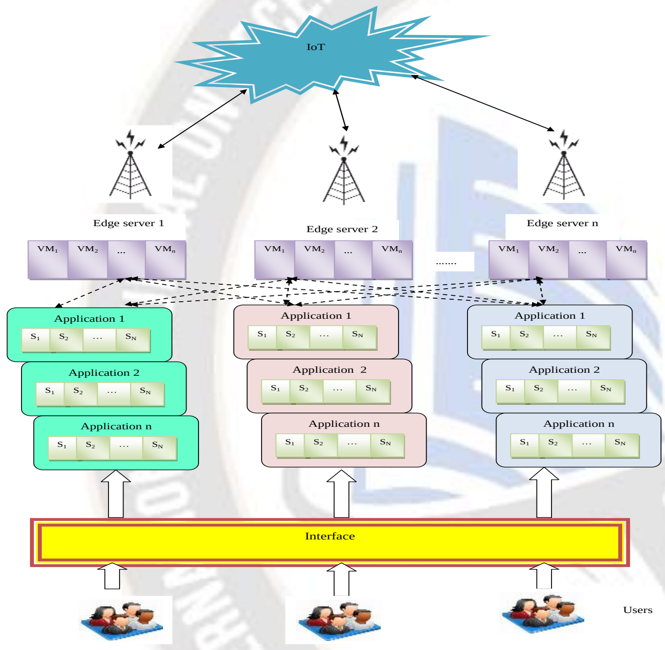

Figure 1. Architecture of Task Scheduling of EC

## B. *Task Scheduling Mechanism*

The two main components of EC are tasks and VMs both of which are essential to task scheduling. The tasks are created by IoT applications and sent to edge servers to be processed. The edge servers, which provide computing resources, are owned by EC. Let $S=(S_1,S_2,....S_N)$ and $VM=(VM_1,VM_2,....VM_n)$ stand for the corresponding collection of tasks and nodes. Where $S_1$ is a initial task and $N$ signifies the overall count of tasks, $VM_1$ the initial virtual machin , and $n$ signifies the overall count of $VM$ virtual machines.

According to Eq. (1), the objective function is established. The dirichlet distribution is used in this calculation to determine the weight.

$$OF=(w_1*F_1)+(w_2*F_2)+(w_3*F_3)+(w_4*F_4)\ (1)$$

$$\sum_{j=1}^{m} w_j = 1 (2)$$

Where, $m=4$ , $w_1,w_2,w_3,w_4$ indicates the weights, makespan is denoted by $F_1$, execution time is indicated by $F_2$ , execution cost is signified by $F_3$ , and $F_4$ is the risk probability.

A Dirichlet-distributed random vector's $X$ probability $p$ density function is proportional to Eq. (3), wherein $\alpha$ is a vector holding the positive constraints of concentration.

$$p(x)\infty\prod_{i=1}^{k} x_i^{\alpha_{i-1}} \qquad (3)$$

The technique computes using the below function: If is dirichlet distributed, then $X=\frac{1}{\sum_{i=1}^{k} Y_i}$ , $Y$ random vector whose components all have the same gamma distribution.

## C. *Task scheduling with NN based risk probability estimation*

As previously mentioned, the suggested task scheduling algorithm in EC takes Execution cost, Makespan, Execution time, and Risk likelihood into account. NN may be used to forecast the risk probability. The following are the specified parameters:

**Makespan $F_1$ :** This term refers to the total amount of time required to complete all jobs. It is given in Eq. (4) where $n \rightarrow$ number of VMs, and $CT_i$ is described in Eq.(5) where $N \rightarrow$ count of tasks in $VM$ , $S_s \rightarrow$ task, $l \rightarrow$ size of S, $\Pr \rightarrow$ processing elements count in $VM$ , and the $MIPS$ speed for execution per processing element of a $VM$ (million instructions per second).

$$MP=\max_{1\le r<n}\{CT_r\}\ (4)$$

$$CT_r=\sum_{s=1}^{N}\frac{S_s l}{VM_s.\Pr\times VM_s.mips}\ (5)$$

**Execution Time $F_2$ :** Execution time refers to the quantity of time each job requires the VM to complete.

**Execution Cost $F_3$ :** Fees for the use of resources to finish the task are paid by the user to the supplier as execution costs.

**Risk Probability Estimation based on NN $F_4$ :** NN may be used to estimate risk probability [16] [17], with task security $S_{\sec}$ and virtual machine security $VM_{\sec}$ as inputs. The model is trained to calculate risk based on these inputs. The model's complete description is provided below

Multiple layers with information and connections between them make up an artificial NN [18]. It allows input and output layers of arbitrary length. Multiple layers of data and connections between them make up an artificial neural network. It allows input and output layers of arbitrary length.The nodes in the artificial neural network are arranged in layers. Each layer's node values are saved as vectors, $M_a$ where the number of the layer is $a$ . The input to the neural network is represented by the zeroth layer $M_0$ . $\sigma_a$ is a function that the layer $a$ uses for activation. $W_a$ are weight matrices, $B_a$ are scalar values known as bias. The number of rows in $W_a$ the matrix and the columns count in the $W_a$ matrix are both equal to the nodes count in $M_a$ layer, and $M_{a-1}$ . The identical activation functions $\sigma(z)=z$ , the hyperbolic tangent $\sigma(z)=\tanh(z)$, & sigmoid $\sigma(z)=(1+\exp(-z))^{-1}$ . The algebraic formulas use biases the and weights of the network to propagate the input through the network to the output layer. Allow the network's layer count to be $b+1$ and the output layer to be $M_b$ . Its value is calculated using

$$M_b=\sigma_b\left(B_b+\ldots+W_4\sigma_3\begin{pmatrix}b_3+W_3\sigma_2\\(b_2+W_2\sigma_1(b_1+W_1\sigma_0))\end{pmatrix}\right)\text{(6)}$$

The NN must be trained, or determine the biases and ideal weights in each data set, before it can be used. Consider that $t_b$ is the output layer and that $t_0$ is the input layer. The NN generates the output $M_b$ if $t_0$ is used as the input. Based on the input $M_0=t_0$ , they define the error vector as $e_b=t_b-M_b$ for the output layer; where $M_b$ is the network's output. The error norm $L_2$ of NN is defined as follows for a single training pair:

$$L_2=e_b.e_b \text{ (7)}$$

There have a predetermined amount of training pairs $O$ , $t_0^{(c)}$ and $t_b^{(c)}$ where $c$ represents the number of training pairs. Finally, the following Eq. (8) determines the neural network's error:

$$E=\frac{1}{O}\sum_{c=1}^{O}L_2^{(c)}=\frac{1}{O}\sum_{c=1}^{O}\left(t_b^{(c)}-M_b^{(c)}\right)^2 \text{(8)}$$

where $M_b^{(c)}$ is gained by $M_0^{(c)}=t_0^{(c)}$ network propagation. Our goal while training the network is to reduce the $E$ by selecting the proper network topology (layer count, size, and types of activation functions), and then determining the best weights and biases.

### *D. Solution encoding*

Take into consideration the inputs that the server for edge has a specific count of virtual machines (VMs) and tasks while scheduling the job using PDPOA. Every job will be given to a specific VM to process. $S=(S_1,S_2,\ldots S_N)$ For instance, the quantity of tasks is $S_N$ . Moreover, the responsibilities are distributed across 3 different VM numbers. The first task $S_1$ is allocated to the first VM depending on objective function in Eq. (1) (Risk probability, Execution time, Makespan, and Execution cost), the second task is allocated to the third VM, & so on up to the tenth job.

Fig. 2 indicates the solution encoding of task scheduling method in PDPOA model.

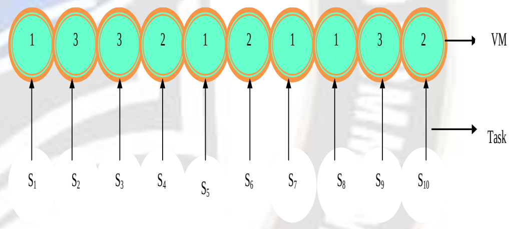

Figure 2. Solution encoding

### *E. Description of Pareto distribution based pelican optimization algorithm for optimal task scheduling process*

Although the existing POA [19] approach shows the correct answers, it is unreliable. In this article, the issues with traditional POA are addressed by the application of the CMPOA approach. It has been demonstrated that self-improvement with existing optimization methods is possible. One portion of the POA's population consists of pelicans. In population-based algorithms, every person in the population indicates a capable results. For every variable in the optimization issue, each participant provides a value on the accessible search area. In Eq. (9), the population's members are first initialized at random.

$$g_{u,v}=C_v+ran(p_v-q_v),\ u=1,2,\ldots H,\ v=1,2\ldots J \text{ (9)}$$

In Eq. (9), $ran$ indicates arbitrary number, $H$ signifies number of population member, $g_{u,v}$ is $v^{th}$ variable achieved through candidate of $u^{th}$ solution, $J$ is amount of problem variable, $p_v$ and $q_v$ refers to $v^{th}$ upper & lower bounds of variables.

The population matrix is given in Eq. (10).

$$G = \begin{bmatrix} G_1 \\ \cdot \\ \cdot \\ G_u \\ \cdot \\ \cdot \\ G_H \end{bmatrix}_{H\times J} = \begin{bmatrix} g_{1,1} & \cdots & g_{1,v} & \cdots & g_{1,J} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ g_{u,1} & \cdots & g_{u,v} & \cdots & g_{u,J} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ g_{H,1} & \cdots & g_{H,v} & \cdots & g_{H,J} \end{bmatrix} \quad (10)$$

where $G \rightarrow$ population matrix of pelicans and $G_u \rightarrow u^{th}$ pelican.

The objective function in Eq. (11) is used to calculate the values.

$$f = \begin{bmatrix} f_1 \\ \cdot \\ \cdot \\ f_u \\ \cdot \\ \cdot \\ f_H \end{bmatrix}_{H\times 1} = \begin{bmatrix} f(G_1) \\ \cdot \\ \cdot \\ f(G_u) \\ \cdot \\ \cdot \\ f(G_H) \end{bmatrix}_{H\times 1} \quad (11)$$

where $f \rightarrow$ vector of objective function and $f_u \rightarrow$ objective function number of $u^{th}$ candidate solution.

This hunting method involves two steps:

(i) Exploration stage (Approaching prey).

(ii) Exploitation stage (Winging over the water surface).

**Exploration Stage (Approaching Prey):**

Before heading that way, the pelicans look for their supper. Because the suggested POA mimics the tactical techniques of the pelican, its search space scanning and exploration power are realistic. Equation (12) makes use of both the pelican's technique and the early mentioned concepts.

$$g_{u,v}^{h_1} = \begin{cases} g_{u,v} + ran(h_v - D.g_{u,v}), & I_h < I_u; \\ g_{u,v} + ran(g_{u,v} - h_v), & else \end{cases} \quad (12)$$

In Eq. (16), $g_{u,v}^{h_1} \rightarrow$new location of $u^{th}$pelican in the $v^{th}$ size, $D \rightarrow$random value that was equivalent to 1 or 2, $I_h \rightarrow$ value of objective function, and $h_v \rightarrow$prey location at $v^{th}$ dimension.

As per proposed PDPOA concept, the pelican location is updated by relying on the best member as per Eq. (13).

$$g_{u,v}^{h_1} = \begin{cases} \begin{bmatrix} g_{u,v} + ran_1(h_v - D.g_{u,v}) + \\ ran_1 * ([2\times ran_1 \times g_{best}] - g_{u,v}) \end{bmatrix}, & I_h < I_u; \\ g_{u,v} + ran(g_{u,v} - h_v), & else \end{cases} \quad (13)$$

Where, $g_{best}$ is the best position of pelican,and $ran_1$ is the random value calculated using sinusoidal map [20]. The sinusoidal iterator is formally defined by the following equation

$$ran_1 = 2.3 * d_1^2.\sin(\pi \times d_1) \quad (14)$$

The novel position update of pelican is specified in Eq. (15) as per the proposed PDPOA concept. Here, we used the pareto distribution as per Eq. (16).

$$g_{u,v}^{h2} = g_{u,v} + \delta \times Pareto(\hat{x}_{\hat{m}}, \tilde{h}) \times (g_{u,v} - g_{best}) \quad (15)$$

Where, $g_{best}$ is the best solution of $\hat{t}$ iteration, and $\delta = 0.0.1$is a constant.

$$\tilde{F}(\hat{x}) = \begin{cases} \left(\dfrac{\hat{x}_{\hat{m}}}{\hat{x}}\right)^{\tilde{h}}, & \hat{x} \geq \hat{x}_{\hat{m}} \\ 0, & \hat{x} < \hat{x}_{\hat{m}} \end{cases} \quad (16)$$

Where, $\tilde{h}$ and $\hat{x}_{\hat{m}}$ are the shape parameter and scale parameter, and $\hat{x}_{\hat{m}}$ has the minimum possible value of $\hat{x}$.

If the pelican's goal is achieved, it may move. Equation (17) offers the POA process paradigm.

$$g_u = \begin{cases} g_u^{h_1}, & I_u^{h_1} < I_u; \\ g_u, & else \end{cases} \quad (17)$$

In Eq. (17), $g_u^{h_1} \rightarrow$novel location of $u^{th}$pelican & $I_u^{h_1} \rightarrow$ objective value in 1st phase.

**Proposed Exploitation stage:**

Conventionally, Eq. (18) uses mathematics to replicate the behavior of pelicans when they are hunting.

$$g_{u,v}^{h2} = g_{u,v} + Q\left(1 - \frac{\hat{t}}{T}\right).(2.ran - 1)g_{u,v} \quad (18)$$

As per proposed PDPOA concept, the pelican location is updated as per Eq. (19). We have applied random dimension of randomly selected neighbor $g_U$

$$g_{u,v}^{h2} = g_{u,v} + Q\left(1 - \frac{\hat{t}}{T}\right).(2.ran - 1).(g_{u,v} - g_U) \quad (19)$$

In Eq. (19), $g \rightarrow$ constant $R = 0.2$ , $g_{u,v}^{h2} \rightarrow$ novel location of $u^{th}$ pelican in the $v^{th}$ dimension in 2nd stage, $Q\left(1 - \frac{\hat{t}}{T}\right) \rightarrow$ neighbour radii of $g_{u,v}$ , $T \rightarrow$ maximum iterations number, $\hat{t} \rightarrow$ current iteration, & $Q \rightarrow$ constant number as 0.2.

The position of new pelican is provided in Eq. (20) then either rejected or accepted by updation, whereas $g_u^{h2} \rightarrow$new pelican position& $I_u^{h_2} \rightarrow$objective number.

$$g_u = \begin{cases} g_u^{h2}, & I_u^{h_2} < I_u; \\ g_u, & else \end{cases} \qquad (20)$$

The pseudo-code of PDPOAscheme is represented in algorithm 1.

```
ALGORITHM 1: PDPOA model
Input the optimization problem.
Determine the iterations count ( $it$ )&population size ( $PS$ ).
Population Initialization
For $\hat{t} = 1:T$
    Generate the position of prey randomly.
    For $D = 1:H$
    Phase 1: Exploration phase
        For $v = 1:J$
            The update as per proposed PDPOA conceptof $v^{th}$ dimensionin Eq. (13).
            The novel position update of pelican is specified in Eq. (15) as per the proposed PDPOA concept.
        End.
        The update of $u^{th}$ population member in Eq. (17).
    Phase 2: Exploitation phase
        For $v = 1:J$
            Determine new status of the $u^{th}$ dimension as per proposed PDPOA concept using Eq. (19).
        End
        The update of $u^{th}$ population member in Eq. (20).
    End.
    Best candidate solution
End.
Return
```

## IV. RESULTS & DISCUSSIONS

### *A. Simulation setup*

The PDPOA paradigm model for task scheduling in edge computing, was executed using Python 3.6, and the results were confirmed. The following are some features of the device: AMD Ryzen 5 3450U CPU, Radeon Vega mobile Gfx 210 GHz, 64-bit operating system, x-64-based processor, and 16GB of installed RAM. The minimum system requirements are Windows 11 Home Single Language Edition, Version: 21H2, and OS Build: 22000.1098.

### *B. Comparitive Analysis*

The suggested PDPOA model comprises the comparison of conventional algorithms such as BOA, BES, HHO, SSA, HBA, and POA. The BOA is a contemporary and effective swarm-based metaheuristic optimization technique. Due to its unique characteristics, such as its high balance between exploration and exploitation and its few adaptive parameters to manage, the BOA has caught the interest of academics.

The BES is a novel type of meta-heuristic optimization algorithm that simulates the social intelligence and hunting strategies of bald eagles in their pursuit of fish by taking its cues from the natural world. A ground-breaking population-based, naturally inspired optimization paradigm is the HHO paradigm. The main sources of inspiration for HHO are Harris' hawks' cooperative demeanor and surprise pounce chasing style in nature. Harris hawks are capable of displaying a range of pursuit tactics depending on the dynamic nature of the scenario and the prey's evasive moves.

SSA is a potent optimization technique that imitates group feeding and predator avoidance behavior of sparrows. The act of searching involves carefully or thoroughly examining something in an effort to find or discover it. The simple act of gathering food for immediate consumption or long-term preservation is referred to as "sparrow seek." The HBA metaheuristic algorithm was created lately and borrows its inspiration from the digging and honey-finding techniques of honey badgers. It explores and utilizes the search space in the process. POA is a new approach to stochastic optimization that draws inspiration from nature. The suggested POA design's primary goal is to resemble pelican hunting behavior.

Makespan, Execution Cost, Fitness, Execution Time, and Risk Probability were further investigated by adjusting the VMs to 10, 20, 30, 40, and 50, correspondingly.

### *C. Execution Cost Analysis of PDPOA Vs extant approaches*

Fig. 3 illustrates the execution cost of the PDPOA is assessed using traditional approaches such as BOA, BES, HHO, SSA, HBA, and POA. Additionally, the PDPOA provided a reduced execution cost of 44 at the 50th virtual machine, whereas more traditional approaches like BOA, BES, HHO, SSA, HBA, and POA produced a greater execution cost. When the VM is initialized to position 40, the PDPOA recorded the lowest execution cost of 39.625, whereas position 30 saw the PDPOA's highest execution cost of 46.664. In general, it has been demonstrated that the PDPOA performs better when tasks are scheduled in edge computing at reduced execution costs.

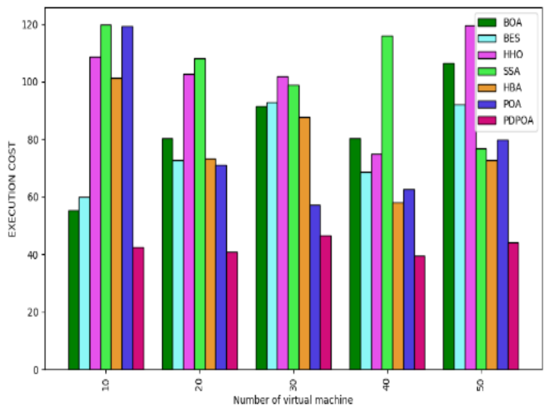

Figure 3. Analysis on Execution cost of PDPOA vs previous models

*D. Analysis on Execution time of PDPOA Vs extant approaches*

An execution time analysis of PDPOA is analyzed with conventional techniques over the standard approaches (BOA, BES, HHO, SSA, HBA, and POA) in the Fig. 4. Likewise, the PDPOA model achieved 1.4358s, 1.7595s, 1.3821s, 1.079s, and 1.147s of execution time while scheduling the tasks in edge computing for 10, 20, 30, 40 and 50 virtual machines. The minimal execution time of PDPOA model is 0.99 obtained in the 40th virtual machine; still, the BOA =2.733, BES =2.659, HHO =2.821, SSA=3.976, HBA =2.592, and POA=1.955, respectively higher execution time. The PDPOA approach has thereby achieved improved task scheduling in edge computing with shorter execution times.

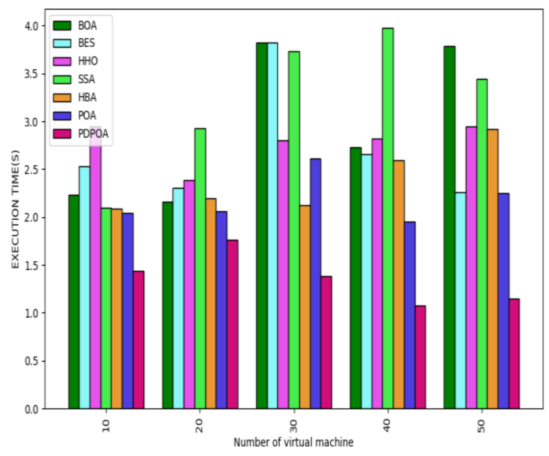

Figure 4. Analysis on Execution time of PDPOA vs previous models

*E. Analysis on fitness of PDPOA versus previous approaches*

The PDPOA model is conflicted with previous methodologies like BOA, BES, HHO, SSA, HBA, and POA model for several VMsbased on Fitness. Fig. 5 presents the results for the fitness measurement. While it experimented with the 50th VM, the PDPOA obtained lowered fitness to 6.4133 while compared to prior models such as BOA is 84.9080, BES is 33.2216, HHO is 22.9076, SSA is 26.7775, HBA is 18.0051, and POA is 16.0240. PDPOA's fitness score for the 30th virtual machine was 7.1982, which was lower than that of more recent approaches such as SMA, AO, CHIMP, DNN, KTCS, TSA, AROA, and BES. This demonstrates that the least fitness obtained by the proposed PDPOA technique assures a higher level of convergence on task scheduling. As a result, it shows how PDPOA may be enhanced for scheduling the task in EC with comparably small Fitness.

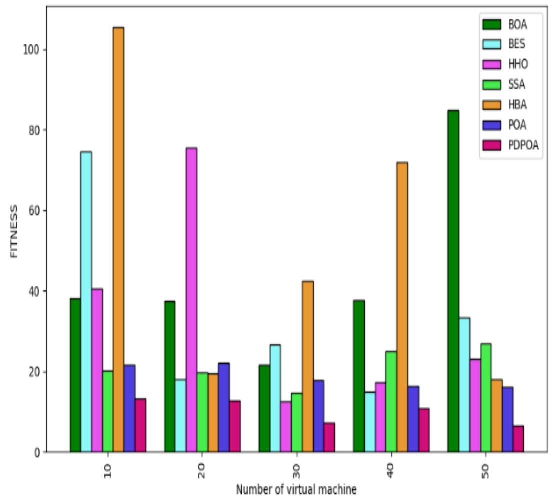

Figure 5. Analysis on fitness of PDPOA vs previous models

*F. Analysis on makespan of PDPOA versus previous approachesAnalysis on Execution Cost of PDPOA Vs extant approaches*

Fig.6 compares the examination of the PDPOA's makespan to well-known techniques such as the BOA, BES, HHO, SSA, HBA, and POA model for unique virtual machines. Additionally, the PDPOA obtained the smallest makespan of 39.7404, outpacing BOA (144.232), BES (64.381), HHO (121.0481), SSA (96.238), HHA (141.176), and POA (56.866) in that order. When evaluated over conventional methods such as the BOA, BES, HHO, SSA, HBA, and POA model, respectively, the PDPOA in the 10th virtual machine achieves a minimum makespan of 44.881. The results suggest that, for task scheduling in EC, the PDPOA has outperformed other alternative schemes with a shorter makespan.

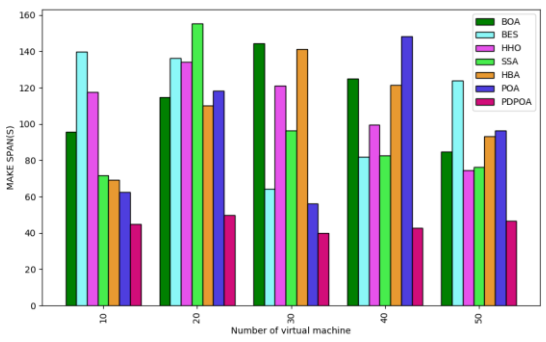

Figure 6. Analysis on makespan of PDPOA versus previous approaches

### *G. Analysis on risk probability of PDPOA versus previous approachesAnalysis on Execution Cost of PDPOA Vs extant approaches*

Figure 7 compares the likelihood of the PDPOA approach to Task Scheduling in EC to that of other existing methods like the BOA, BES, HHO, SSA, HBA, and POA model. Despite the fact that the BOA is 0.252, BES is 0.270, HHO is 0.293, SSA is 0.282, HBA is 0.243, and POA is 0.232, and for the thirty-first virtual machine, the PDPOA model's risk probability is 0.181. The 10, 20, 30, and 50 virtual machines' respective risk probabilities for the PDPOA are 0.1950, 0.2881, 0.1810, 0.2093, and 0.2189. Because the PDPOA performs more consistently than other conventional algorithms, it offers outstanding outcomes with lesser risks.

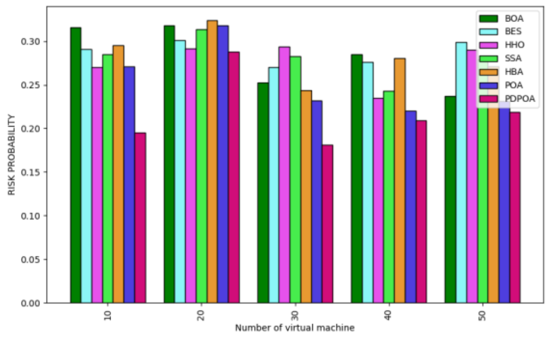

Figure 7. Analysis on risk probability of PDPOA versus previous approaches

### *H. Convergence Analysis*

The chosen PDPOA model's convergence to the traditional schemes is assessed in Fig. 8 by altering the iteration number from 0, 5, 10, 1, 20, and 25, respectively. As the number of iterations rises, the PDPOA method contracts in the convergence analysis of the suggested method. The suggested PDPOA method's cost function likewise minimized from the seventh to the ninth iteration and remained constant. Comparing the chosen PDPOA scheme to rival models like BOA, BES, HHO, SSA, HBA, and POA, A lower constant value (2.3) is obtained from the chosen scheme's cost function from the ninth to the twenty-fifth iteration. The proposed PDPOA approach accomplishes the least cost function in accordance with its objectives as stated in Eq. (1). It is evident that the PDPOA technique had produced better results at lower costs.

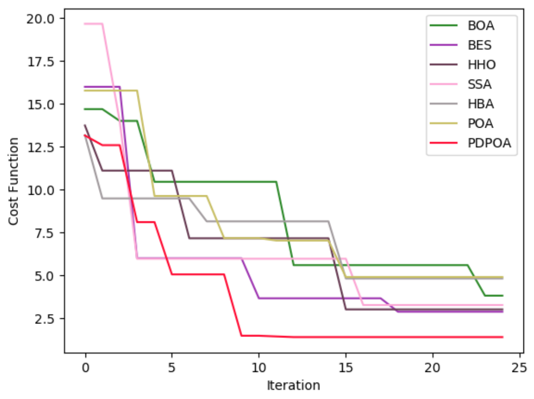

Figure 8. Convergence analysis of suggested model over traditional schmes

## V. CONCLUSION

This research has provided a new task scheduling model that considers Execution cost, Makespan, Risk probability, and Execution time into account. The NN specifically estimated risk probabilities while taking task security and virtual machine security into consideration. This work suggested a new PDPOA model for optimum scheduling of tasks. Results from the proposed system are examined and compared to those from earlier methods using a variety of metrics, including Execution cost, Makespan, Risk probability, Execution time, etc. The minimal execution time of PDPOA model is 0.99 obtained in the 40th virtual machine; still, the BOA =2.733, BES =2.659, HHO =2.821, SSA=3.976, HBA=2.592, and POA=1.955, respectively higher execution time. When compared to other conventional techniques like the BOA, BES, HHO, SSA, HBA, and POA model, the recommended PDPOA model obtains (44.782) with higher accurate findings in Makespan in the mean case scenario.

## REFERENCES

[1] Y. Zhang, J. Fu, Energy-efficient computation offloading strategy with tasks scheduling in edge computing. Wireless Netw, vol. 27, pp. 609–620, 2021.

[2] Z. Chen, J. Hu, X. Chen, J. Hu, X. Zheng and G. Min, "Computation Offloading and Task Scheduling for DNN-Based Applications in Cloud-Edge Computing," in IEEE Access, vol. 8, pp. 115537-115547, 2020.

[3] X. Huang, P. Li and R. Yu, "Social Welfare Maximization in Container-Based Task Scheduling for Parked Vehicle Edge Computing," in IEEE Communications Letters, vol. 23, no. 8, pp. 1347-1351, Aug. 2019.

[4] W. K. Lai, C. -S. Shieh and Y. -P. Chen, "Task Scheduling With Multicore Edge Computing in Dense Small Cell Networks," in IEEE Access, vol. 9, pp. 141223-141232, 2021.

[5] C. Li, J. Tang, Y. Luo, "Collaborative cache allocation and task scheduling for data-intensive applications in edge computing environment", Future Generation Computer Systems. 2019

[6] X. Wang, Z. Ning, S. Guo and L. Wang, "Imitation Learning Enabled Task Scheduling for Online Vehicular Edge Computing," in IEEE Transactions on Mobile Computing, vol. 21, no. 2, pp. 598-611, 2022.

[7] Y. Liu "Dependency-Aware Task Scheduling in Vehicular Edge Computing," in IEEE Internet of Things Journal, vol. 7, no. 6, pp. 4961-4971, 2020.

[8] W. Zhang, Z. Zhang, S. Zeadally and H. -C. Chao, "Efficient Task Scheduling With Stochastic Delay Cost in Mobile Edge Computing," in IEEE Communications Letters, vol. 23, no. 1, pp. 4-7, Jan. 2019.

[9] J. Huang, S. Li, Y. Chen, 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, 2020.

[10] Y. Kim, C. Song, H. Han, H. Jung and S. Kang, "Collaborative Task Scheduling for IoT-Assisted Edge Computing," in IEEE Access, vol. 8, pp. 216593-216606, 2020.

[11] I. Ullah, H.Y. Youn, Task Classification and Scheduling Based on K-Means Clustering for Edge Computing. Wireless Pers Commun, vol. 113, 2611–2624, 2020.

[12] Z. Zhu, A Hardware and Software Task-Scheduling Framework Based on CPU+FPGA Heterogeneous Architecture in Edge Computing," in IEEE Access, vol. 7, pp. 148975-148988, 2019.

[13] L.feng, C. Xianglin, W. Xiulei Wang, "Failure-resilient DAG task scheduling in edge computing", Computer Networks, 2021

[14] X. Zhao, G. Huang, Q. Gao, "Low load DIDS task scheduling based on Q-learning in edge computing environment", Journal of Network and Computer Applications. 2021

[15] J. Zhang, X. Zhou, T. Ge, X. Wang and T. Hwang, "Joint Task Scheduling and Containerizing for Efficient Edge Computing," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 8, pp. 2086-2100, 2021.

[16] V. Garima, “Secure VM Migration in Cloud: Multi-Criteria Perspective with Improved Optimization Model”, Wireless Personal Communications, 2022.

[17] Z. Li, W. Xu, H. Shi, Y. Zhang and Y. Yan, “Security and Privacy Risk Assessment of Energy Big Data in Cloud Environment”, Computational Intelligence and Neuroscience, vol.2021, pp.11, 2021

[18] J. Ravnik, J. Jovanovac, M. Hriberšek, "A sigmoid regression and artificial neural network models for day-ahead natural gas usage forecasting",, Cleaner and Responsible Consumption, 2021

[19] T. Pavel, D.Mohammad. Pelican Optimization Algorithm: A Novel Nature-Inspired Algorithm for Engineering Applications. Sensors. 22. 855. 2022.

[20] X. Zhao, F. Yang, Y. Han and Y. Cui, "An Opposition-Based Chaotic Salp Swarm Algorithm for Global Optimization," in IEEE Access, vol. 8, pp. 36485-36501, 2020.