Journal of Ambient Intelligence and Humanized Computing (2023) 14:2981–2992
https://doi.org/10.1007/s12652-023-04544-6

ORIGINAL RESEARCH

# An efficient meta-heuristic resource allocation with load balancing in IoT-Fog-cloud computing environment

**Ismail Zahraddeen Yakubu**[1] · **M. Murali**[1]

Received: 28 May 2022 / Accepted: 19 January 2023 / Published online: 2 February 2023


**Abstract**
Fog computing is considered a derivative of cloud computing that aims to reduce the huge transmission latency and CPU time, as well as the overall cost of resource usage in the cloud. The deployment of Internet-of-Things (IoT) enabled smart systems, which frequently demand real-time processing, is rapidly expanding. Following that, the volume of generated data and computation workload dramatically increased. Fog resources are limited and typically resource constrained. Therefore, it is impossible to execute all tasks at the edge network. To support the increasing amounts of data and computation, cloud computing, associated with significant delays in transmission and processing of workload, is used. The distribution of tasks between the cloud and fog layer and the allocation of layer resources to satisfy the users' demands prevents layer oversaturation, service degradation, and resource failure due to excessive workload is challenging. This paper proposes a layer fit algorithm that evenly distributes tasks between the fog and cloud, based on priority levels. Also, a Modified Harris-Hawks Optimization (MHHO) based meta-heuristic approach is proposed to assign the best available resource to a task within a layer. The key intention of this paper is to reduce the makespan time, task execution cost, and power consumption and enhance resource usage in both the fog and cloud layer. The simulations are performed using the iFogSim simulation toolkit. The proposed layer fit algorithm and the Modified Harris-Hawks Optimization (MHHO) are compared with the traditional Harris-Hawks Optimization (HHO), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and the Firefly Algorithm (FA). Based on the experimental results, the MHHO has improved the performance of the system in terms of makespan time, execution cost, and energy consumption. The ability of the MHHO to balance the load across resources yields a significant improvement when the number of tasks increases as compared to the traditional HHO and other optimization algorithms.

**Keywords** Fog computing · Cloud computing · Layer fit algorithm · Task allocation · Execution time · Harris-Hawks Optimization (HHO) · Internet-of-Things (IoT) · Resource utilization

## 1 Introduction

Fog computing is a distributed computing paradigm that utilises the components between cloud datacenters and IoT gadgets, providing storage and processing services proximate to the edge devices (Mahmud et al. 2017). In fog computing, devices with processing and storage capabilities, like smartphones, base stations, switches, and routers, perform similar tasks as the cloud resources (Abbasi et al. 2020). Fog computing was first introduced by Cisco (Sarkar and Misra 2016), as an additional virtualized layer between the cloud and IoT devices, to address the limitations of the cloud (Agarwal et al. 2016; Verma et al. 2016; Xu et al. 2017). However, fog computing doesn't replace the cloud; rather, it provides real-time services with mobility support and low latency that are not available in the cloud.

The Internet-of-Things (IoT) technology integrates physical objects with sensing abilities, mobile objects, electronic devices, and home appliances with the internet (Ngu et al. 2016). These devices generate an enormous volume of data and work load that needs to be processed in real-time. To satisfy the QoS requirements of these time-sensitive applications, fog computing must come to play.

In spite of the benefits of fog over cloud, there are some flaws in fog computing that can't be overlooked. For

✉ Ismail Zahraddeen Yakubu
iy1242@srmist.edu.in

[1] Department of Computing Technologies, SRM Institute of Science and Technology, Kattankulathur, India

instance, fog resources are limited and resource constrained. Therefore, it is impossible to execute the enormous volume of data and workload at the fog layer without engaging the cloud. The combination of fog and cloud computing models enables the harnessing of both fog resources and cloud datacenter resources to meet the QoS requirements of these time-sensitive applications (Mahmud et al. 2020).

In fog computing, resource scheduling and assignment is the logical method to map available resources to users over the internet (Choudhari et al. 2018). Allocation of resources over a time period in some logical order is extremely important due to its stringent delay demand. To get the most out of fog computing, it is very important to allocate resources well (Rafique et al. 2019).Otherwise, ineffective resource allotment could lead to higher delays and suboptimal use of resources.

With the limited number of resources in fog computing and the increasing number of requests, it is desirable to process user tasks based on their respective priority levels (Dakshayini and Guruprasad 2011; Pawar and Wagh 2012). Given the importance of response time, makespan, and latency, the workload must be distributed between the fog and the cloud layer so that the optimum value can be achieved.

According to the literature, a number of resource allocation techniques have been proposed to exploit the real benefits of both fog and cloud computing. Naha et al. (2020) consider response time, processing capacity, and bandwidth of a resource when allocating a task for processing. If there are no available resources in the fog, then fog servers or cloud resources will be used to complete the task. The method allocates fog resources to a task based on performance and availability. However, time-sensitive applications like health care applications, augmented reality, gaming e.t.c. (Dastjerdi and Buyya 2016; Puliafito et al. 2019; Khattak et al. 2019) can propagate to the cloud due to a lack of resources in the fog that may be occupied by delay-tolerant tasks. In this case, the aim of the fog is defeated.

In Mani and Meenakshisundaram (2020), tasks are processed by a proximate fog device server, with complete or part of the processing needed by a task. In the absence of resources in the fog, tasks are forwarded to the cloud for processing. However, this method may not satisfy the QoS demands of time-sensitive applications. The paper by Rafique et al. (2019) try to minimise average response time and optimise resource usage but fail to address the issue of priority on usage of fog layer resources.

This paper proposes a priority-based workload allocation in IoT-Fog-Cloud computing. The model is suited for time-sensitive applications. A layer fit algorithm is proposed to distribute tasks between fog and cloud based on their priority levels. The priority level of an incoming task is determined by the base station nearest to the user, and then it forwards a task to the fog layer if it's a priority task and to the cloud layer if it's a non-priority task. Also, a modified Harris-Hawk algorithm based meta-heuristic approach is proposed to map the best available resource to a task within a layer. The MHHO employs an exponential energy update strategy to prevent the population from falling into local optima. To ensure load balance, the MHHO employs a local search strategy to avoid resource overload.

The key contributions of this paper are as follows:

- To develop a layer-fit algorithm that distributes tasks between fog and cloud based on their priority levels.
- Propose a Modified Harris-Hawk Optimization based meta-heuristic approach that selects the best available resource within a layer to meet the QoS demands of users' tasks.
- To reduce the oversaturation in the fog layer due to increasing demand for resources in the fog layer.

The proposed resource allocation approach is implemented using the iFogSim toolkit and is bench marked with the traditional HHO, Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Firefly Algorithm (FA) in the iFogSim environment. The results illuminate the efficacy of the allocation technique, in terms of task execution cost, completion (makespan) time, and overall power consumption of the system.

The rest of this paper is structured as follows: Section II presents work related to resource allocation. Section III presents the working principle of the standard meta-heuristic HHO optimization algorithm. Section IV explains the proposed system architecture and resource allocation approach. Section V illustrates the simulation steps, presents the details of the experiment, and discusses the results obtained. Lastly, section VI concludes the work and presents the future research directions.

## 2 Related work

This section presents the various methods proposed in literature to address the task scheduling problems in a fog-cloud environment.

In Bitam et al. (2017), a bio-inspired optimization approach entitled "Bees Life Algorithm" was proposed to address the task allocation problem in a fog computing environment. The main objective of this work is to determine the optimum tradeoff between task completion time and storage needs for fog services generated by mobile users. However, this work focuses on the fog layer without considering the cloud layer.

In Potu et al. (2021), an extended Particle Swarm Optimization with additional gradient methods was

proposed to address the task scheduling problem in fog computing. The main objective of this work is to enhance the performance of fog resources and reduce task execution time. Considering the drawbacks of fog computing as mentioned in Sect. 1, using the fog layer alone to execute all IoT applications is impossible without involving the cloud.

In Akintoye and Bagula (2019), the authors formulate tasks and resource allocation problems in a single fog-cloud environment. A Hungarian Algorithm Based Binding Policy (HABBP) was used to execute a new allocation policy in the widely used CloudSim simulator. Also, a Genetic Algorithm Based Virtual Machine Placement (GABVMP) was proposed to address and optimise the virtual machine placement problem in a cloud environment.

In Iyapparaja et al. (2019), authors proposed a model to improve quality of service through efficient resource allocation using Queuing Theory based Cuckoo Search (QTCS) model. The tasks received from users are graded and allotted resources based on their relative weights. Authors used the queuing theory to prioritize the tasks and the cuckoo search to optimize resource allocation. Despite the priority given to tasks, the model only considers the fog layer, which requires the cloud layer to satisfy the clients' QoS demands.

In Abouaomar et al. (2019), Lyapunov optimization is used to formulate a resource allocation problem in fog computing. Authors optimise resource allocation through entity supervision by using a resource representation scheme that exposes the resources of individual devices in the fog through the Mobile Edge Computing Application Programming Interface. This work also focuses on resource allocation and optimization in fog computing, leaving out the cloud.

In Naha and Garg (2021), the dynamic changes in the behaviour of fog users are addressed using a multi-criteria-based resource allocation. The approach takes into consideration both fog computing characteristics like heterogeneity, mobility, resource constraints and dynamic variation in user requirements to make resource reservations. Allocation of resources to a task is done through a multi-objective function. The results of this work prove promising in terms of low response time and minimal SLA violations. However, authors focus on resource allocation from fog perspectives leaving out the cloud.

In Mohammady Talvar (2021), the authors improve resource allocation in fog computing through the Nash equilibrium and auction algorithm. In this approach, each player is assigned a matrix that represents fog nodes, data service subscribers, and data service operators. Individual players generate the optimal strategy based on other players' strategies. This work also focuses on the fog layer, leaving out the cloud layer.

In Sarma et al. (2021), an optimised fuzzy clustering-based resource scheduling and dynamic load balancing algorithm is proposed to address fog computing resource scheduling and load balancing problems. An enhanced fast fuzzy c-mean with crow search optimization is used to allocate resources to tasks, and finally, load balancing is performed using a scalability decision technique. In this work, both resource optimization and load balancing among resources are performed, but only focused on allocating resources to tasks at the fog layer without using the cloud layer.

In Abedi et al. (2022), authors proposed an enhanced firefly algorithm based on load balancing to address the Dynamic Resource Allocation (DSA) problem in a cloud environment. The method ensures load balance between available resources and minimises completion time by choosing appropriate objectives in the fitness function. The authors employ a heuristic approach, rather than the conventional random approach, to create the initial population of the firefly algorithm, which is based on the tasks' priority.

In Salem et al. (2022), the authors review the various meta-heuristic algorithms used to address resource allocation problems in a fog environment. Based on the findings by the authors, it was observed that meta-heuristic algorithms achieve better performance in terms of cost, time, energy usage, and resource utilization.

In Shakarami et al. (2021), the authors proposed an autonomous offloading framework to address the issues faced by time-intensive and resource-intensive applications. In this work, numerous simulations, including Deep Neural Networks, multiple linear regression, hybrid models, and Hidden Markov Models as the planning module of the aforementioned autonomous technique, were carried out to deal with the size of the offloading decision-making problem.

In Guerrero et al. (2022), authors examine and analyse various resource optimization approaches in fog computing with a focus on genetic-based solutions, their characteristics, and their respective design options. In this article, the authors presented a comprehensive, exhaustive, and systematic review of the state-of-the-art techniques.

In Abohamama et al. (2022), the authors proposed a semi-dynamic real-time scheduling approach for task applications in a hybrid cloud-fog computing environment. A modified Genetic Algorithm (GA) is used to generate schedules of permuted tasks and assign tasks to the virtual resources based on the order of the best permutation.

In Singh et al. (2022), authors proposed a cluster-based load balancing approach that ensures load balance among fog-cloud environment resources. The approach considers three different resource cluster states: busy, working, and free. Tasks are allocated to resource clusters based on their

state. The proposed algorithm keeps track of the number of clusters and resources in each cluster to ensure efficient allocation.

## 3 Harris Hawk Optimization (HHO) algorithm

The HHO is a cutting-edge meta-heuristic algorithm that imitates hawks' hunting and prey-capture behaviours. Heidari et al. (2019) proposed the HHO approach to tackle problems with global optimization. As shown in Fig. 1 below, HHO conducts the seeking process in two steps (exploration and exploitation) using a variety of tactics.

Figure 1 shows the exploration and exploitation phase of the Harris Hawks based on various strategies used. The blue border shows the exploitation phase with the energy and strategies used by the hawks. The orange border indicates the exploration phase with the various patching strategies based on the random locations and relative positions of other hawks.

a. *Exploration phase*

In this phase, the hawks initiate the prey search process by perching on various locations in the hunting territory. In an attempt to spot the potential prey item, the hawks perch based on the relative positions of their neighbors. In some cases, the hawks perch based on random locations within their home range. When no prey item is spotted, the hawks advance the hunting area and the prey search process continues until a potential prey item is spotted. The relative position and random position perching have an equal probability of success, which is modelled in Eq. 1 below.

$$P(t+1) = \begin{cases} P_r(t) - r_1\left|P_r(t) - 2r_2P(t)\right|, q \geq 0.5 \\ (P_{rabbit}(t) - P_{avg}(t)) - \theta, q < 0.5 \end{cases} \quad (1)$$

where $P(t+1)$ represents the position vector of hawks in the subsequent iteration t. $P(t)$ represents the current position vector of hawks, $P_{rabbit}(t)$ represent the position of the prey (rabbit), $r_1$ through $r_4$ and $q$ are random value inside (0, 1), $\theta = r_3(LB + r_4(UB - LB))$ and $P_{avg}$ is the average position of the current population and is formulated as:

$$P_{avg}(t) = \frac{1}{N}\sum_{i=1}^{N} P_i(t) \quad (2)$$

where $P_i(t)$ represents the location of individual hawk in iteration t and N is the population size.

b. *Transition phase*

In this phase, the hawks transfer from exploration to exploitation. The transition takes place immediately after a prey item is spotted by the perched hawks. The hawks exploit the prey, and the prey applies different escape behaviours, which considerably decreases its energy. The exploitation strategy used by the hawks to intercept the prey varies with the variation in the energy of the prey. Equation 3 below denotes the energy model of the prey.

$$E_p = 2E_0\left(1 - \frac{t}{T_m}\right) \quad (3)$$

where $E_p$ represents the escaping energy of the prey, $E_0$ represents the initial energy of the prey, $T_m$ represents the total number of iterations and t represents the current iteration. $E_0$ varies between -1 and 1. $E_0$ decreases from 0 to -1, if the prey is weak and increases from 0 to 1 if the prey strengthen.

c. *Exploitation Phase*

In this phase, the spotted prey is attacked by the perched hawks using various attacking strategies. On the other hand, the prey attempts to escape the attack using various escape strategies. Based on the escaping and attacking behaviour exhibited by both the prey and the hawks, four (4) attacking strategies of HHO are modelled as follows:

I. *Soft Besiege*

In this attack strategy, the prey is encircled softly by the hawks to exhaust the energy of the prey and finally

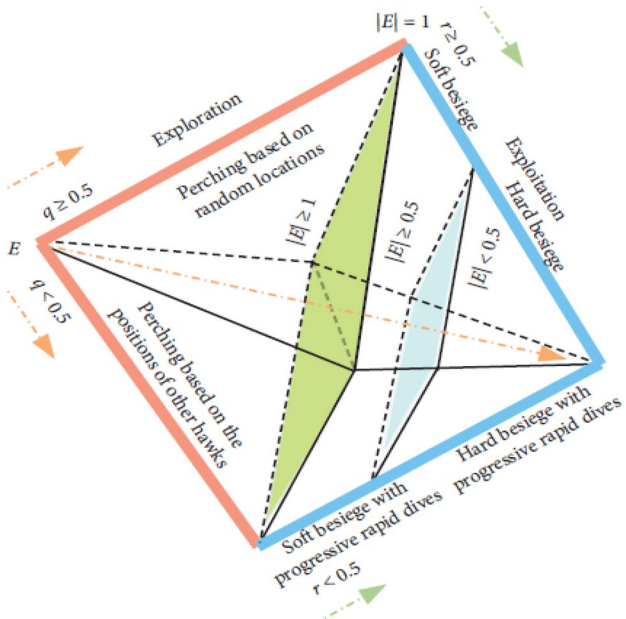


**Fig. 1** Phases of the HHO (Heidari et al. 2019)

perform a surprise pounce to intercept the prey. This attacking strategy is employed when the prey has enough energy to escape the attack. The soft besiege attacking behaviour of the hawks is modelled in Eq. 4.

$$P(t+1) = \Delta P(t) - E_p \left| JP_{rabbit}(t) - P(t) \right| \tag{4}$$

$$\Delta P(t) = P_{rabbit}(t) - P(t) \tag{5}$$

J represents the random pounce and is given as:

$$J = 2\left(1 - r_5\right) \tag{6}$$

where $\Delta P(t)$ represents the difference between the position vector and current position of the prey in iteration t, $r_5$ is a random value inside (0, 1).

This behavior is applied when the chances of escaping the attack is greater than 0.5 (i.e. $r \geq 0.5$) and the energy of the prey is greater than 0.5 (i.e. $\left|E_p\right| \geq 0.5$).

II. *Hard Besiege*

In this strategy, the prey is weak and has low escaping energy ($r \geq 0.5 \& \left|E_p\right| \geq 0.5$). The hawks hardly encircle the prey and finally perform a surprise pounce to intercept the prey. This behaviour is formulated as:

$$P(t+1) = P_{rabbit}(t) - E_p \left| \Delta P(t) \right| \tag{7}$$

III. *Soft Besiege with Progressive Rapid Dive*

In this strategy, the prey has enough energy to escape ($r < 0.5 \& \left|E_p\right| \geq 0.5$). Hence, the prey performs a zigzag deceptive motion to escape the attack. The hawks construct a soft besiege before pounce and competitively dive towards the prey. The hawks base their next move on the following rule in Eq. 8

$$Y = P_{rabbit}(t) - E_p \left| JP_{rabbit}(t) - P(t) \right| \tag{8}$$

The result of the movement is compared to the previous dive to detect whether it will be efficient or not. If not efficient, then an irregular, abrupt, and rapid dive towards the prey is performed based on the following rule in Eq. 9

$$Z = Y + S \times LF(D) \tag{9}$$

where D represent the dimensionality of the problem, S represent a random vector of size $1 \times D$, LF represent a levy flight function, which is calculated as follows:

$$LF(x) = 0.01 \times \frac{u + \sigma}{|v|^{\frac{1}{\beta}}},$$

$$\sigma = \left( \frac{T(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{T\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{\frac{1}{\beta}} \tag{10}$$

where u, v are random parameters of LF inside (0, 1) and $\beta$ is a constant with value of 1.5

The HHO algorithm selects the best strategy for updating the position of hawks based on Y and Z as follows:

$$P(t+1) = f(p) = \begin{cases} Y, if F(Z) < F(P(t)), \\ Z, if F(Z) < F(P(t)). \end{cases} \tag{11}$$

IV. Hard Besiege with Rapid Dive

In this strategy, the prey does not have enough energy to escape ($r < 0.5 \& \left|E_p\right| < 0.5$). The hawks construct a hard besiege before pounce and reduce the distance of their average location towards the prey. The following rule in Eq. 12 is performed in a hard besiege condition.

$$P(t+1) = f(p) = \begin{cases} Y, if F(Z) < F(P(t)), \\ Z, if F(Z) < F(P(t)). \end{cases} \tag{12}$$

where Y and Z are obtained using the rule in Eqs. 13 and 14 below:

$$P(t+1) = P_{rabbit}(t) - E_p \left| JP_{rabbit}(t) - P_{avg}(t) \right| \tag{13}$$

$$Z = Y + S \times LF(D) \tag{14}$$

$P_{avg}(t)$ is obtained using Eq. 2

Algorithm 1: Harris Hawk Optimization (HHO)

**Input:** Population size N & total no of iterations $T_m$

**Output:** the various locations of the prey and its fitness value

Randomly generate initial population $P_i$, $i \in (1, N)$

**While** (not termination condition) **do**

Compute the fitness values of Harris-hawks

Set $P_{rabbit}$ to best location

**for (**each Harris-hawk ($P_i$)) **do**

Update $E_p$ & $J$ using eqn. 3 & 6

**if** ($|E_p| \geq 1$ ) **then**

Compute the new position of hawks $P_i$ using eqn. 1

**if** ($|E_p| < 1$ ) **then**

**if** ($r \geq 0.5$ &$|E_p| \geq 0.5$ ) **then**

Determine the new value $P_i$ using eqn. 4

**else if** ($r \geq 0.5$ &$|E_p| < 0.5$ ) **then**

Determine the new value $P_i$ using eqn. 7

**else if** ($r < 0.5$ &$|E_p| \geq 0.5$ ) **then**

Determine the new value $P_i$ using eqn. 11

**else if** ($r < 0.5$ &$|E_p| < 0.5$ ) **then**

Determine the new value $P_i$ using eqn. 12

**Return** $P_{rabbit}$

## 4 Proposed method

To curtail the drawbacks of the techniques presented in Sect. 1, a layer fit algorithm has been developed to improve the allocation of workloads between fog and cloud. Also, a Modified Harris-Hawk Optimization (MHHO) based meta-heuristic approach is proposed to assign the best available resource to a task within a layer. The modification of the HHO is done in two ways. First, we modify the energy update equation of the HHO to avoid the population falling into local optima. Secondly, we employ load balancing strategy to avoid over-loading of resources, which can lead to resource failure or performance degradation.

A. *System Architecture*

The model in this work consists of an edge layer, a base station, a fog layer, and a cloud layer. Devices in the edge layer generate tasks to be processed by either the fog or cloud. The tasks from edge devices are first received by the base station. The base station decides on the number of tasks to be assigned to the fog and the amount that should be assigned to the cloud. The fog layer consists of fog devices that are relatively close to the base station. Devices in the fog layer are limited and resource constrained. Therefore, the base station allocates resources from the fog layer based on priority. The cloud layer consists of resources with high processing and storage capacity but far away from the base station and edge devices. Both fog and cloud layers maintain a resource manager that monitors resources in the layer. The base station asks the layer resource manager for the number and average capacity of the active resources.

Figure 2 shows the architecture used in this paper. The edge devices generate the tasks to be processed and forward these tasks to the fog layer through the base station available in the fog layer. Tasks eligible for processing by the fog layer are allocated resources, and non-eligible tasks are forwarded to the cloud layer for processing.

B. *Layer Selection and Allocation*

The selection and allocation of layers to an incoming task is based on priority. For each incoming task, the base station computes the task priority as follows:

$$Del^T_{max} = D^T_l - t^T_i \tag{15}$$

where $Del^T_{max}$= maximum tolerable time of task $T_i$, $D^T_l$= deadline of task $t_i$ and $t^T_i$ is the current time.

$$Del^T_{trans} = h_t * m \tag{16}$$

where $Del^T_{trans}$ = delay in transmitting task on the network, $h_t$= intensity of the congestion and $m$ is the number of task to be transmitted.

$$Del^T_{proc} = \frac{U^T_c}{n(t).k - U^T_c} \tag{17}$$

where $Del^T_{proc}$ = processing delay, n(t) represent number of active fog resources, k represent the processing capacity of each resource, $U^T_c$ represent the current workload in the fog layer. In this case, M/G/1 queuing model is considered.

$$T_{total} = Del^T_{trans} + Del^T_{proc} \tag{18}$$

$$Priority = \begin{cases} Priority, if Del^T_{max} \leq T_{total} \\ Non - Priority, Otherwise \end{cases} \tag{19}$$

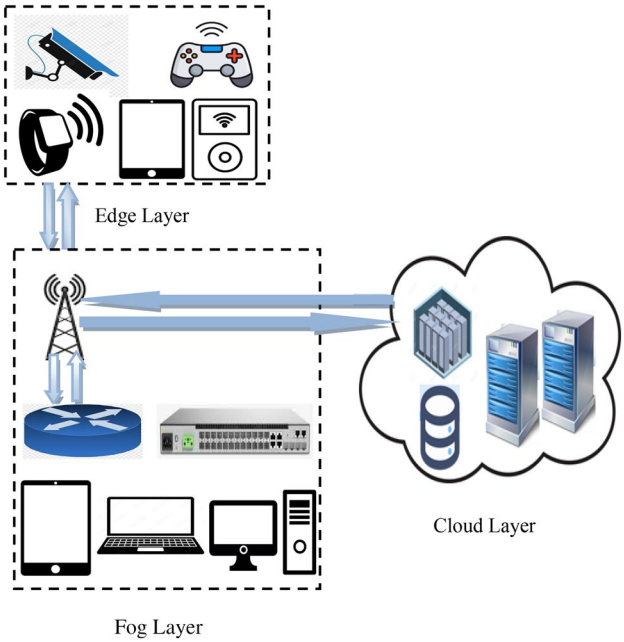


**Fig. 2** The proposed Model for Workload Allocation

If a task is a priority task, then a resource in the fog layer is used to process the task. Otherwise, a resource in the cloud layer is used to process the task.

Algorithm 2: Layer Fit Algorithm

Input: Tasks, Number of active servers in fog.
Output: Task to Layer Mapping
**For each** task $t_i$ **do**
Compute task maximum delay using eqn. 15
Compute task transmission delay using eqn. 16
Compute task processing delay using eqn. 17
Compute total time to be spent in fog using eqn. 18
**If** $Del^{T}_{max} \leq T_{total}$ **then**
Offload task to fog
**Else**
Offload task to cloud
**End if**
**End for**

C. *Fog Layer Resource Allocation*

In this section, the Modified Harris Hawk optimization (HHO) algorithm is used to allocate resources to tasks in the fog layer.

*I. Modified Harris Hawks Optimization (MHHO)*

The energy update strategy in the standard Harris-Hawk Optimization (HHO) is accompanied by a major drawback. The energy can never exceed one (1) when the number of iterations approaches 50% of the total number of iterations (H. Jia et al. 2019). For multi-peak and high-dimensional problems like task scheduling, the population is likely to fall into local optima. To address the drawback of the standard HHO, the exponential decreasing strategy (Y. Zhang et al. 2020) is applied. The exponential energy update strategy is given as:

$$E = e^{-\frac{t_m}{T}} \tag{20}$$

A local search strategy is used to make sure that resources are used evenly so that the standard HHO can do a better job of allocating resources.

The proposed method consists of four stages as given below:

a. *Initial Phase*

In this phase, the modified HHO determines the population size, total number of iterations $t_m$, number of tasks m submitted to the fog for execution, and a random number that represents the candidate solution. Where n is the size of the VM's to be assigned to tasks.

b. *Evaluation Phase*

In this phase, the solution in the initial population is evaluated using three performance measures: makespan, cost, and energy consumption. The solution is accessed using a fitness function derived from the three performance measures as explained below.

a) *Completion (Makespan) Time*

The makespan time is the total time elapsed while executing the entire task using the available resources. When user tasks are submitted to the fog layer, the tasks are transferred to the fog broker, who maintains the tasks' properties and processing demands. The fog broker requests from the fog resource manager the services required to process the tasks from users. The tasks are then mapped to the detected services.

Given the set of m independent tasks $T = \{t_1, t_2, \ldots, t_m\}$ received from users, the characteristics of the tasks are task length measured in millions of instructions (MI), task deadline, and task arrival time. The fog broker is responsible for allocating those tasks to available resources (VM's) to meet the user's demands. Given the set of n heterogeneous virtual

machines $VM = \{VM_1, VM_2, \ldots, VM_n\}$, the time taken to execute task $t_i$ on $VM_j$ can be obtained by Eq. 21 below:

$$E_{i,j}^{t} = \frac{t_i^l}{VM_j^p} \tag{21}$$

where $E_{i,j}^{t}$, is the time to execute task $t_i$ on $VM_j$. $t_i^l$ is the length of task $t_i$ and $VM_j^p$ is the computing power of $VM_j$.

Since VM's are heterogeneous, their CPU capacities will vary from one VM to another. Hence, tasks executed on multiple VM's will encounter multiple costs of execution. Based on the characteristics of the VM's and tasks submitted, the fog broker computes an m by n matrix that represents the execution time of each task on various virtual machines.

$$E_{i,j}^{t} = \begin{bmatrix} E_{1,1}^{t} & E_{1,2}^{t} \ldots .. & E_{1,n}^{t} \\ \ldots .. & \ldots \ldots \ldots .. & \ldots .. \\ E_{m,1}^{t} & E_{m,2}^{t} \ldots .. & E_{m,n}^{t} \end{bmatrix} \tag{22}$$

The makespan of the schedule X is given as follows:

$$M^{t}(X) = max_{j=1,2,\ldots n} \sum_{i=1}^{n} E_{i,j}^{t} \tag{23}$$

b) *Execution Cost*

Cost refers to the total amount to be paid by the user to the service provider for the services rendered. The cost of each resource (VM) in the fog is dynamically affected by the capacity of the resource. This means that a more powerful resource is always more expensive. The cost of resource usage is charged based on the execution time of a task and the cost of VM per unit of time. Therefore, the execution cost $E_{i,j}^{C}$ of task $t_i$ on resource $VM_j$ can be determined using Eq. 24.

$$E_{i,j}^{C} = P_j^{vm} * \frac{E_{i,j}^{t}}{3600} \tag{24}$$

where $P_j^{vm}$ represents the price of $VM_j$, $E_{i,j}^{t}$ is the execution time of task $t_i$ on $VM_j$.

c) *Energy Consumption*

The amount of energy consumed by a virtual machine depends on its state. Virtual machines consume less energy in their idle state as compared to their busy state. The energy consumed by a virtual machine is calculated as follows:

$$Energy_j^{vm} = \left(exec_j^{vm} x\alpha + idle_j^{vm} x\beta\right) * PS_j \tag{25}$$

where $\alpha and \beta$ represents the energy consumed in joules per Millions of Instructions in busy state and the energy consumed at idle state. $PS_j$ represents the processing speed of the *VM*.

d) *Fitness Function*

The key objective of resource allocation is to determine an efficient mapping of a user's tasks to computing resources that optimises some objectives. The key objectives of this work are to reduce the makespan time, cost of execution, and overall energy consumption of the system. The objective function in this work is given as follows:

$$f(x) = \min\left(\left(max_{j=1,2,\ldots n} \sum_{i=1}^{n} E_{i,j}^{t} * 0.5\right) + \sum_{i=1}^{n} E_i^{C} * o.5\right) \tag{26}$$

c. *Update Phase*

In this phase, the initial solution is updated based on the output of the evaluation phase (fitness value). The process of updating the solution is repeated until the termination criteria is reached.

d. *Balance Phase*

In this phase, a local search is applied to determine the resources with the highest and least number of allocated tasks. A random task allocated to the resource with the highest number of tasks is removed. This task will be allocated to a resource with the least number of tasks if it lowers the fitness value. The process is repeated until optimal load balance is achieved.

Algorithm 3: Modified Harris-Hawks Optimization

**Input:** number of individual hawk N, number of task m, and number of virtual machines n, total no of iterations$T_m$.
Generate a random population of hawks.
Compute the fitness values of each hawk using eqn. 26
Set $P_{rabbit}$ to best value
$t = 1$
**repeat**
  Update $E_p$ & $J$ using eqn. 20 & 6
  **if** ($|E_p| \geq 1$) **then**
    Compute the new position of hawks $P_i$ using eqn. 1
  **end if**
  **if** ($|E_p| < 1$) **then**
    **if** ($r \geq 0.5$ &$|E_p| \geq 0.5$ ) **then**
      Determine the new value of $P_i$ using eqn. 4
    **else if** ($r \geq 0.5$ &$|E_p| < 0.5$) **then**
      Determine the new of $P_i$ using eqn. 7
    **else if** ($r < 0.5$ &$|E_p| \geq 0.5$) **then**
      Determine the new of $P_i$ using eqn. 11
    **else if** ($r < 0.5$ &$|E_p| < 0.5$) **then**
      Determine the new of $P_i$ using eqn. 12
    **end if**
  **end if**
  $\boldsymbol{P_{bal\_load}} = P_{rabbit}$
  Sort $\boldsymbol{P_{bal\_load}}$ in order of allocated task
  **for** i = $\frac{n}{2} + 1$ *to* $n$ **do**
    randomly remove a task $t_{ran}$ from $vm_i$
    randomly select $vmj \in (1, \frac{n}{2})$
    assign $t_{ran}$ to $vmj$
    **if** ($fitness(P_{bal_{load}}) \leq fitness(P_{rabbit})$)**then**
      $P_{rabbit} = \boldsymbol{P_{bal\_load}}$
    **end if**
  **end for**
  t = t + 1
**until** $\boldsymbol{t > T_m}$
**Return** $P_{rabbit}$

## 5 Experimental results and discussion

In this section, the details of the experiment and the results obtained are presented.

a. *Experimental Setup*

Extensive simulations have been carried out using the iFogSim (H. Gupta et al. 2017) simulator to assess the efficacy of the proposed approach. The proposed algorithm is encoded in Java with the Eclipse IDE using iFogSim and ran on a machine with the specifications in Table 1. The fog environment consists of 20 heterogeneous virtual machines with randomly generated processing and memory capacity. The specifications of the virtual machines are presented in Table 2. The cost of using the VMs is determined by their respective computation power. To obtain the utilisation cost of an individual virtual machine, its capacity is divided by $10. For example, a VM with a processing speed of 1000 will cost $100 per unit time. The experiment was conducted using 100 randomly generated tasks with varying characteristics, as shown in Table 3. The parameter settings for the proposed method (MHHO) and the benchmarked techniques (PSO, FA, HHO) are presented in Table 4.

The experiment is conducted in two ways: firstly, by increasing the number of VM's and maintaining the number of tasks. Secondly, by increasing the number of tasks and maintaining the number of VM's, the performance of the approach is measured based on completion time (makespan), cost, and energy consumption (Table 4).

b. *Results and Discussion*

For performance investigation, the experiments conducted include the proposed Modified Harris-Hawk

**Table 1** System Specification

| Properties | Specification |
|---|---|
| System | Intel (R) Core i5 5th Gen @ 1.7 GHz |
| RAM | 4 GB |
| OS | Windows 10 64-bit Operating System |

**Table 2** VM specifications

| Properties | Range |
|---|---|
| No of VM's | 5–20 |
| CPU (MIPS) | 1000–3000 |
| RAM (BM) | 1000–2000 |
| Cost ($) | 100–300 |

**Table 3** Task specifications

| Properties | Range |
|---|---|
| No of Instructions (MIPs) | 100—500 |
| Memory Required (MB) | 100–500 |
| Deadline (ms) | 5–50 |

**Table 4** Parameter settings for PSO, FA, HHO, and MHHO

| Algorithm | Parameters | Values |
|---|---|---|
| ACO | Population size | 50 |
| | β | 2 |
| | ρ | 6 |
| PSO | Population size | 50 |
| | $C_1$ | 1.45 |
| | $C_2$ | 1.45 |
| FA | Population size | 50 |
| | α | 0.5 |
| | β | 0.2 |
| | γ | 1 |
| HHO | Population size | 50 |
| | $E_o$ | [ 1, 1] |
| MHHO | Population size | 50 |
| | $E_o$ | [ 1, 1] |
| | β | 0.85 |

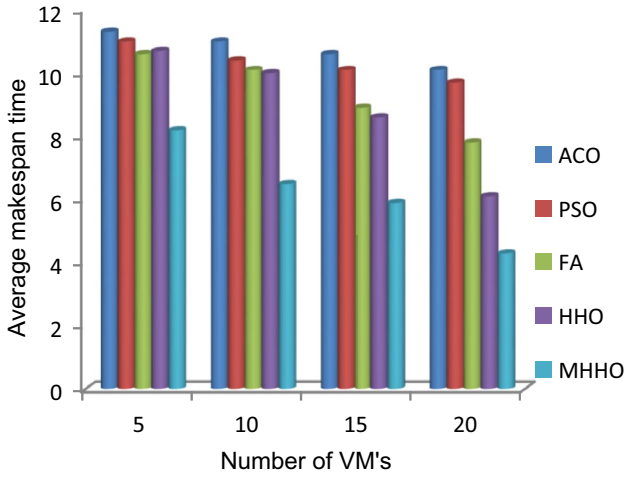


**Fig. 3** Average Makespan for 100 task and different VM size

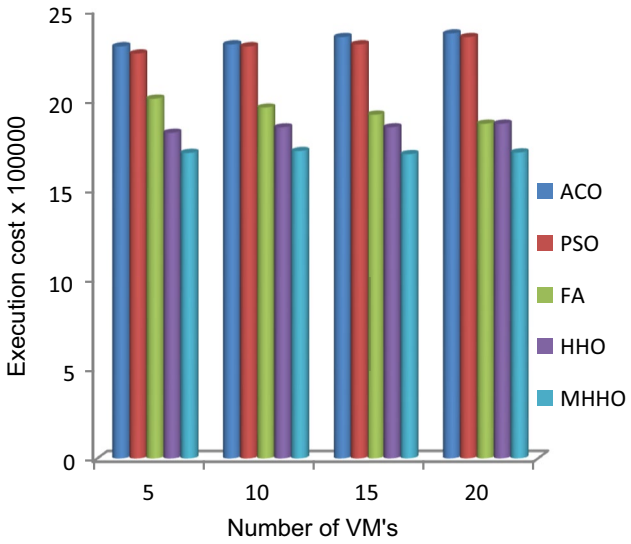


**Fig. 4** Comparison of Average execution cost for 100 task and different VM size

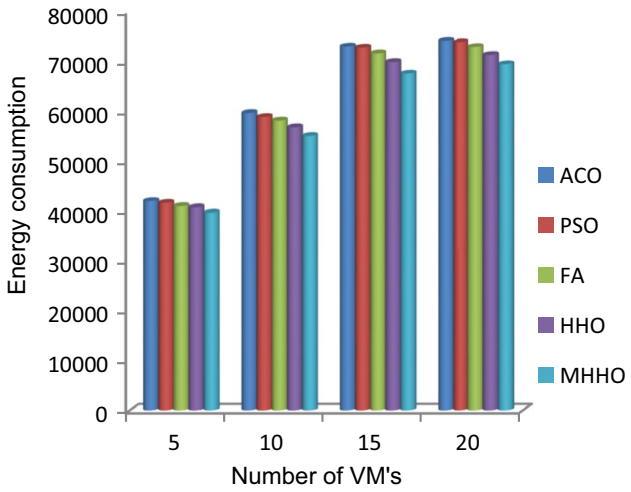


**Fig. 5** Comparison of Energy Consumption with different VM size

Optimization (MHHO), the traditional Harris-Hawks Optimization (HHO), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Firefly Algorithm in the iFogSim simulator.

The simulation was run fifty (50) times for each number of VM's, and the average of the makespan time, cost, and energy consumed was calculated. Figure 3 shows the average makespan time of the five algorithms on different VM sizes. It was discovered that increasing the number of VMs reduces the average makespan time for both algorithms. The proposed Modified Harris-Hawks Optimization (MHHO) algorithm records the least average makespan time. This proves that the load balancing and the exponential energy update strategy in the proposed method have an impact on the completion time of the tasks.

Figure 4 shows the average execution cost of 100 tasks on different VM sizes for the five algorithms. It was observed that there were no changes in the execution cost between the traditional HHO and the MHHO. This shows that the fitness function is a key part of how both algorithms control how much it costs to run. Based on the results in the figure, the MHHO recorded the least execution cost for the 100 tasks. This proves that the load balancing feature in the proposed method can improve the performance of the HHO when employed in resource scheduling where there is limited processing capacity.

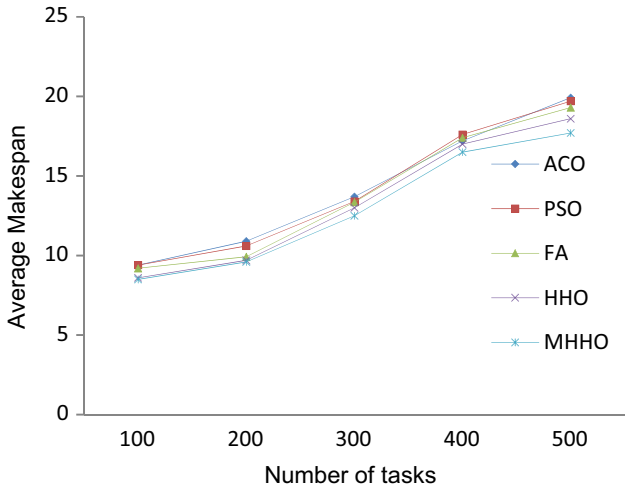


**Fig. 6** Average Completion time (Makespan) for different task size with 20 Virtual Machines

Figure 5 shows the average amount of energy consumed with an increase in the size of resources in the environment. It can be inferred that the energy consumption increases with the rise in the size of resources (VM's), thus inflating the total energy consumption of the system.

Figure 6 shows the comparison of the average makespan time for 500 tasks with 20 virtual machines. It is observed that the average makespan time for the five algorithms increases with an increase in the number of tasks. It was inferred that the increase was as a result of the increase in the workload on the existing virtual machines in the environment. It was also observed that the proposed MHHO records the least average makespan time from tasks 300 to 500 as compared to the traditional HHO.

## 6 Conclusion

Fog computing is a model that aims to overcome the drawbacks of cloud computing by performing computation at the edge of the network. Fog resources are limited and typically resource constrained. To exploit the real benefits of the fog, there is a need for a robust approach for allocating resources to tasks, especially time-sensitive ones. To prevent the fog layer from oversaturation by delay tolerant tasks, the proposed layer fit algorithm filters the tasks to be executed in the fog and propagates the remaining tasks to the cloud layer. The proposed Modified Harris-Hawks Optimization (MHHO) algorithm allocates resources to the filtered tasks, taking into accounts the completion time, execution cost, and energy consumption. To prevent the population of the HHO from falling into local optima and avoid overloading of resources in the environment, a new energy update function and load balance strategy are used in the MHHO. The simulation results illustrate that our system recorded the least makespan time, execution cost, and resource energy consumption when equated to the traditional HHO, ACO, PSO, and FA. In our subsequent work, a joint task schedule and application module placement on fog devices will be considered to further improve the performance of the fog system.

**Data availability** The data used to evaluate the performance of the method in this study is synthetic and are available from the corresponding author upon request.

## References

Abbasi M, Yaghoobikia M, Rafiee M, Jolfaei A, Khosravi MR (2020) Efficient Resource Management and workload allocation in fog–cloud computing paradigm in IOT using learning classifier systems. Comput Commun 153:217–228

Abedi S, Ghobaei-Arani M, Khorami E, Mojarad M (2022) Dynamic resource allocation using improved firefly optimization algorithm in cloud environment. Appl Artif Intell 36(1):1–27

Abohamama AS, El-Ghamry A, Hamouda E (2022) Real-time task scheduling algorithm for IOT-based applications in the cloud–fog environment. J Netw Syst Manage 30(4):1–35

Abouaomar A, Cherkaoui S, Kobbane A, Dambri OA (2019) A resources representation for resource allocation in fog computing networks. IEEE Global Commun Conf (GLOBECOM) 2019:1–6. https://doi.org/10.1109/GLOBECOM38437.2019.9014146

Agarwal S, Yadav S, Yadav AK (2016) An efficient architecture and algorithm for resource provisioning in fog computing. Int J Inform Eng Electron Business 8(1):48–61

Akintoye S, Bagula A (2019) Improving quality-of-service in cloud/fog computing through Efficient Resource Allocation. Sensors 19(6):1267

Bitam S, Zeadally S, Mellouk A (2017) Fog computing job scheduling optimization based on bees swarm. Enterprise Inform Syst 12(4):373–397

Choudhari T, Moh T, Moh T-S Prioritized task scheduling in fog computing. In: Proceedings of the ACMSE 2018 Conference, 2018.

Dakshayini M, Guruprasad HS (2011) An optimal algorithm for priority based service scheduling policy for cloud computing environment. Int J Comput Appl 32:0975–8887

Dastjerdi AV, Buyya R (2016) Fog computing: helping the internet of things realize its potential. Computer 49(8):112–116

Guerrero C, Lera I, Juiz C (2022) Genetic-based optimization in Fog computing: Current trends and research opportunities. Swarm Evol Comput 72:1–22

Gupta H, VahidDastjerdi A, Ghosh SK, Buyya R (2017) IFogSim: a toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments, Software: Practice and Experience, 47(9):1275–1296.

Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris Hawks optimization: algorithm and applications. Futur Gener Comput Syst 97:849–872

Iyapparaja M, Khalaf Alshammari N, Sathish Kumar M, Siva Rama Krishnan S, Lal Chowdhary C Efficient Resource Allocation in fog computing using QTCS model, Comput Materials Continua 70(2):2225–2239, 2022.

Jia H, Lang C, Oliva D, Song W, Peng X (2019) Dynamic harris hawks optimization with mutation mechanism for satellite image segmentation. Remote Sens 11(12):1421

Khattak HA, Arshad H, ul Islam S, Ahmed G, Jabbar S, Sharif AM, Khalid S Utilization and load balancing in fog servers for Health Applications, EURASIP Journal on Wireless Communications and Networking, vol. 2019, no. 1, 2019.

Mahmud R, Srirama SN, Ramamohanarao K, Buyya R (2020) Profit-aware application placement for integrated fog–cloud computing environments. J Parallel Distributed Comput 135:177–190

Mahmud R, Kotagiri R, Buyya R (2017) Fog computing: A taxonomy, survey and Future Directions, Internet of Things, 103–130.

Mani SK, Meenakshisundaram I (2020) Improving quality-of-service in fog computing through efficient resource allocation. Comput Intell 36(4):1527–1547

Mohammady Talvar H, Haj Seyyed Javadi H, Navidi H, Rezakhani A (2021) A new resource allocation method in fog computing via non-cooperative game theory, J Intell Fuzzy Syst 41(2):3921–3932.

Naha RK, Garg S (2021) Multi-criteria–based dynamic user behaviour–aware resource allocation in fog computing. ACM Trans Internet Things 2(1):1–31

Naha RK, Garg S, Chan A, Battula SK (2020) Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud environment. Futur Gener Comput Syst 104:131–141

Ngu AH, Gutierrez AH, Metsis V, Nepal S, Sheng MZ IOT middleware: A survey on issues and Enabling Technologies. IEEE Internet Things J 1–1, 2016.

Pawar CS, Wagh RB Priority based dynamic resource allocation in cloud computing, In: 2012 International Symposium on Cloud and Services Computing, 2012.

Potu N, Jatoth C, Parvataneni P, Optimizing resource scheduling based on extended particle swarm optimization in fog computing environments, Concurrency Comput 33(23), 2021.

Puliafito C, Mingozzi E, Longo F, Puliafito A, Rana O (2019) Fog computing for the internet of things. ACM Trans Internet Technol 19(2):1–41

Rafique H, Shah MA, Islam SU, Maqsood T, Khan S, Maple C (2019) A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in Fog Computing. IEEE Access 7:115760–115773

Salem AH, Ghaleb Al-Gaphari, Meta-heuristic algorithms for resource allocation in fog computing, Int J Modern Trends SciTechnol 08(02):134–143, February, 2022

Sarkar S, Misra S (2016) Theoretical modelling of Fog computing: a green computing paradigm to support IOT applications. IET Networks 5(2):23–29

Sarma B, Kumar R, Tuithung T (2021) Optimised fuzzy clustering-based resource scheduling and dynamic load balancing algorithm for Fog computing environment. Int J Comput Sci Eng 24(4):343

Shakarami A, Shahidinejad A, Ghobaei-Arani M (2021) An autonomous computation offloading strategy in mobile edge computing: a deep learning-based hybrid approach. J Netw Comput Appl 178:1–19

Singh P, Kaur R, Rashid J, Juneja S, Dhiman G, Kim J, Ouaissa M (2022) A fog-cluster based load-balancing technique. Sustainability 14(13):1–14

Verma M, Bhardwaj N, Yadav AK (2016) Real time efficient scheduling algorithm for load balancing in fog computing environment. Int J Inform Technol Comput Sci 8(4):1–10

Xu J, Palanisamy B, Ludwig B, Wang Q, Zenith: Utility-aware resource allocation for edge computing. In: 2017 IEEE International Conference on Edge Computing (EDGE), 2017.

Zhang Y, Zhou X, Shih P-C (2020) Modified harris hawks optimization algorithm for Global Optimization Problems. Arab J Sci Eng 45(12):10949–10974