Journal of Theoretical and Applied Information Technology
31st July 2023. Vol.101. No 14
© 2023 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

INVESTIGATION OF OPTIMAL TASK OFFLOADING AND RESOURCE ALLOCATION USING HYBRID GREY WOLF LION OPTIMIZATION (HGWLO) IN CLOUD–EDGE COMPUTING

K.VINOTHKUMAR[1], DR. D. MARUTHANAYAGAM[2]
[1]Research Scholar, Periyar University, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India
[2] Head/Professor, PG and Research Department of Computer Science, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India.
E-mail:[1]vinothkumarphd2022@gmail.com,[2]kesimaruthu@gmail.com

ABSTRACT

The two main components of edge computing are task offloading and resource allocation. System energy consumption can be reduced and task processing times increased with a sensible job offloading and resource allocation plan. The vast majority of existing research on the task migration of edge computing only takes the resource distribution between terminals and edge servers into account, completely excluding the enormous computing resources in the cloud centre. Under cloud edge computing, Hybrid Grey Wolf Lion Optimization (HGWLO) using job offloading and resource matching method was presented in order to adequately utilise cloud and edge server resources. This research study establishes the job offloading decision of many end-users as a task scheduling in cloud edge computing with the experimental findings showing the that the suggested algorithm outperforms other pre-existing algorithms of dragonfly, grey wolf, and lion optimization with regard to Makespan, Energy Latency and Energy Consumption, System Utility, Task Completion Time, Execution Delay, and Convergence Rate.

Keywords: *Edge Cloud Computing, Resource Allocation, Task Scheduling, Lion Optimization Algorithm, Grey Wolf Optimization.*

I. INTRODUCTION

One of the key components of 5G networks, edge computing has drawn a lot of attention in recent years. In order to sink cloud computing capabilities to the edge of the network, share the computing load of nearby mobile users, and increase the network's overall computing capacity, edge computing deploys servers at the edge of the wireless access network (such as small base stations, macro base stations, cellular base stations, and WiFi access points)[1]. Although the edge cloud does not have the powerful computing capability of the central cloud, it is closer to the terminal device and does not need to be transmitted through the backbone network, reducing transmission distance significantly. The computing capabilities of smart devices, however, are typically constrained [2] by their physical size and the limitations of their underlying hardware and cannot fulfill the demands of these applications. As a result, edge cloud computing technology [3][4][5] is regarded as an effective and promising method of addressing the challenges associated with smart devices with limited resources and mobile applications with high demand. As opposed to traditional cloud computing [6][7], For handling delicate tasks with minimal latency, edge cloud is more appropriate. It can distribute service management and calculation in a smart home, significantly improving operational efficiency. Computers use the concepts of memory and cache. Users' frequently used data is placed closer to the edge of networks to reduce latency while reducing the load on the core network. To address the issues of insufficient processing capacity and limited resources of smart devices, the industry has introduced the concept of offloading in a cloud edge computing environment. The current **edge computing problem's main research direction is how to offload the tasks performed by the devices to the edge server and make efficient and reasonable offloading decisions.** Another critical aspect of scientific task scheduling is the data placement strategy.

It can only consider resource allocation between terminals and edge servers to mitigate data of edge computing, ignoring the massive computing resources in the cloud centre. In cloud edge environments, transmission and data storage costs are incurred. There are many algorithms on how to obtain task offloading decision-making in order to sufficiently utilize cloud and edge server resources in the computational offloading of multiple smart devices and multiple edge processors [8][9]. This research paper proposed an algorithm for hybrid grey wolf lion optimization (HGWLO) using task offloading and resource matching algorithm under cloud edge computing. Many research studies have been proposed to investigate the problems associated with task offloading. Through **the optimization of offloading decisions and the associated resource allocation, such as the allocation of transmit power and computation resource, system performance gains, such as reduced delay or energy consumption**, can be obtained.

Task offload: When using task offloading methods in edge-cloud computing, these methods do not take into account the task offloading decision because they assume that all tasks are offloaded to the edges or clouds or that the user chooses which tasks to offload. These methods are appropriate for task offloading in the scene of tasks requested by devices with limited processing power, such as sampling sensors [10], or for experts.

Resource allocation challenges: The resource allocation mechanisms that determine how and where the offloaded tasks will be executed in a remote platform have a significant impact on task offloading. As a result, **task offloading and resource allocation decisions are inextricably linked and must be addressed together**. The following are the main issues this issue raises.

Partitioning Decision: The resource allocation mechanisms that determine how and where offloaded tasks will be executed in a remote platform have a significant impact on task offloading. As a result, task offloading and resource allocation decisions are inextricably linked and should be addressed together. The following are the main issues this issue raises. ***A poor partitioning decision may result in application execution performance bottlenecks.*** As a result, a balance between when and which tasks should be offloaded to the Cloud/Edge must be sought, taking into account any potential transmission costs in terms of energy, delay, and resources..

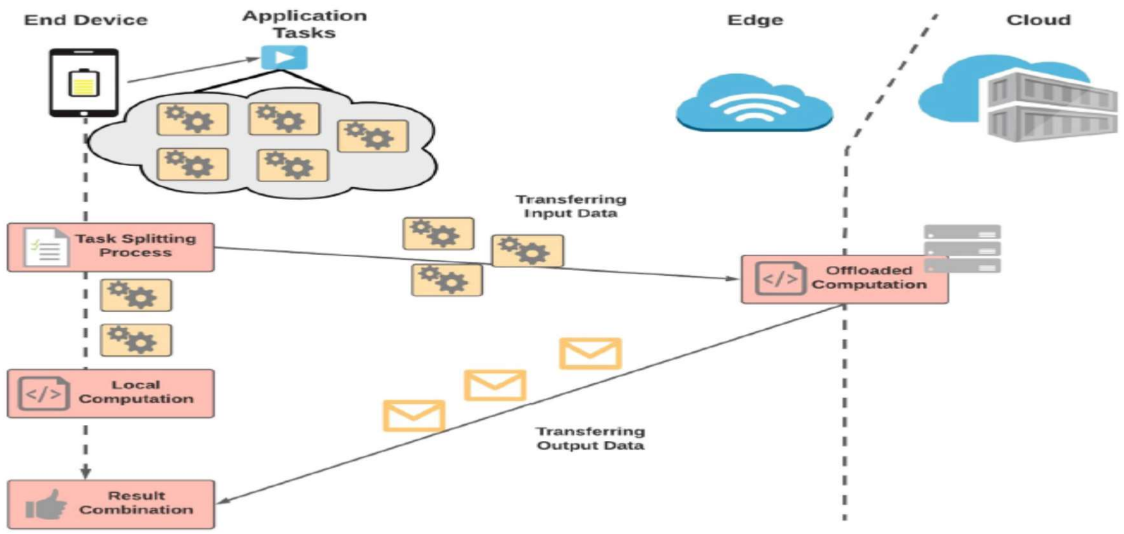


Figure 1: Task Offloading Process

Local computation and task offloading are performed by edge users. Local computing is the act of directly calculating and processing data on-site. The process of offloading is computing task to an edge server. The calculation operation is performed by the edge server, and the result is returned to the edge user. The computing task is offloaded to the edge server for execution, which

Journal of Theoretical and Applied Information Technology
31st July 2023. Vol.101. No 14
© 2023 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

can achieve the goal of relieving the local device's calculation and storage pressure, thereby extending the battery's service life. **All task programmes are assumed to be partition able in this research**. Assume that the kth edge user's time to unload the task is k, W is the system bandwidth, and hk is the channel gain between the edge user and the edge server, pk is the kth edge user's transmitting power, and 2 is the noise power. The total number of bits unloaded by all users to the edge server and the number of bits unloaded from the kth edge user to the server are as shown in,

$$R_k^o = \tau_k w \quad \log_2(1 + \frac{p_k h_k}{\sigma^2})$$

$$R_0 = \sum_{k=1}^{k} R_k^o = \sum_{k=1}^{k} \tau_k w \quad \log_2(1 + \frac{p_k h_k}{\sigma^2})$$

Task offloading refers to the user equipment processing some computationally intensive applications and uploading the data processing these applications to the edge server through wireless transmission under the condition of weighing continuous or other indicators. Resource allocation refers to the edge server for these uploads the processing application allocates certain computing resources, in this way to obtain continuous or gradual replacement, providing a better user experience. The issue to be resolved by task offloading is for each user, and determines whether the tasks generated by it need to be offloaded, while also taking into account the dense deployment of edge servers, the constrained computational load of distributed servers. Therefore, it must be resolved for each offload processing the issue of how much computing resources are allocated by the application. ***To produce better results, these two issues must be addressed concurrently.***

2. METAHEURISTIC OPTIMIZATION ALGORITHMS

As previously stated, the meta-heuristic optimization algorithms Greywolf, dragonfly, and lion optimization algorithms were used as the foundation for adapting the scenario of task scheduling and offloading in a cloud edge successful transformation..

2.1. Dragonfly Algorithm

The Dragonfly Algorithm, a relatively new algorithm, is based on the swarming behaviors of dragonflies, which include both static and dynamic swarming. Dragonflies in the former form small groups that hunt in small areas; in the latter, many dragonflies move over a long distance. These features serve as the foundation for exploitation and exploration. Separation to avoid a collision, alignment for velocity matching, cohesion for moving towards the centre of the neighbourhood, attraction to food, and distraction away from the predator are some of the corrective patterns of dragonflies in a swarm [11]. Each fly's position is updated using a step vector, which is obtained by combining each of the above patterns with the inertial factor as follows [12]:

- Separation refers to the avoidance of static collisions between individuals in the neighbourhood.
- Alignment, that is the matching of an user's velocity to that of others in the neighbourhood.
- Cohesion, that also describes a community's tendency for its residents to gravitate toward the centre.

The separation factor is calculated to avoid a static collision between one dragonfly and other dragonflies in the vicinity, and it is as follows:

$$s_i = -\sum_{j=1}^{N} x_i - x_j \qquad (1)$$

Where Xi represents the current dragonfly's position, Xj represents the position of the j-th neighbour, and N represents the number of neighbouring dragonflies. Following are the calculations for the alignment factor, which is used to compare one dragonfly's velocity to those of nearby dragonflies:

$$A_i = \frac{\sum_{j=1}^{N} v_j}{N} \qquad (2)$$

In which Vj is the jth neighbor's velocity and N is the number of dragonflies in the neighbourhood. The following is how cohesion is calculated:

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X \qquad (3)$$

In which X represents the present user's position, N represents the number of neighbourhoods, and Xj represents the position of the j-th neighbouring individual. The following formula is used to calculate attraction to a food source:

$$F_i = X^+ - X \qquad (4)$$

Where X represents the current individual's position and X+ represents the location of the food source. The following formula is used to calculate enemy distraction:

$$E_i = X^- + X \quad (5)$$

In which X represents the current individual's position and X-denotes the enemy's position.

Exploration and exploitation can be accomplished using the five parameters S, A, C, F, and E given in eq.(1) to (5). Proper tuning of these parameters aids in the discovery of the best solution. Thus, eqs. (6) and (7) are used to compute the step vector and position of dragon flies, respectively.

$$\Delta x_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) \quad (6)$$

$$X_{t+1} = X_t + \Delta x_{t+1} \quad (7)$$

In the preceding equation, t represents the number of iterations, and I represents the ith fly. If a dragon fly has no neighbours, the levy flight is used to update its position as shown in equation (8).

$$X_{t+1} = X_t + Levy(d) \times x_t \quad (8)$$

Where t is the current iteration number and d is the dimension of the position vectors. In each iteration, the step vector and position vectors of each dragonfly are updated until the end criterion is met. Algorithm 1 contains the pseudo code for the dragonfly algorithm.

Algorithm1:Dragonfly algorithm

```
Initialize the populations position randomly;
Initialize the step vectors;
While end condition do
        Calculate the objective values of all drangonflies;
        Update the food source and enemy;
        Update the weights;
        Calculate the factors using(1)-(5);
        Update radius of neighborhoods;
If dragonfly has one or more neighbors then
        Update step vector using (6);
        Update position vector using (7);
        Else
        Update position vector using (8);
        End
        Check and correct new position based on upper and lower bounds;
End.
```

2.2. Greywolf Optimization

There is a clear social hierarchy in the grey wolf pack. This four-level hierarchy is led by Alpha, who is regarded as the wolf at the top of the pyramid. It is in charge of making group hunting decisions. At the second level, we have the Beta, who is the wolf who assists the leader in making pack decisions. The Delta is responsible for the safety of the pack and is subordinate to the wolves above in the hierarchy. The pack's other wolves are known as Omegas, and they are at the bottom of the pyramid [13].

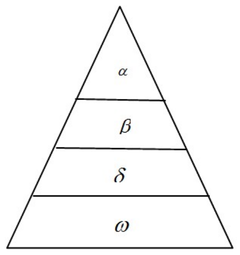


Figure 2: Structure Of Grey Wolf

Hunting: After completing the encirclement of the prey, the Grey wolves concentrated on hunting the prey, the wolf alpha (α) usually leads the hunt, and the wolf Beta (β) and delta (δ) may share in the hunting process in the limited search area. It is not possible to know the best location (prey). When simulating the hunting behavior of Grey wolves, we assume that alpha (δ) is the best initial solution and that elements or wolves beta (β) and delta (δ) have a better knowledge of the potential location of the prey. The behavior of Alpha (α), beta (β) and delta (δ) can be simulated by the following equations.

Upon having completed the encirclement of the prey, the Grey wolves concentrated on hunting the prey; the wolf alpha (α) usually leads the hunt, and the wolves Beta (β) and delta (δ) may participate in the hunt in the limited search area. The best location cannot be determined (prey). When simulating Grey wolf hunting behaviour, we assume that alpha (α) is the best initial solution and that elements or wolves beta (β) and delta (δ) have a better understanding of the potential prey location. The following equations can be used to simulate the behaviour of Alpha (α), Beta (β), and Delta (δ).

$$\vec{D}_\alpha = |\vec{c}_1 * \vec{x}_\alpha - \vec{x}| \quad (9)$$

$$\vec{D}_\beta = |\vec{c}_2 * \vec{x}_\beta - \vec{x}| \quad (10)$$

$$\vec{D}_\delta = |\vec{c}_3 * \vec{x}_\delta - \vec{x}| \quad (11)$$

The location vector can be calculated from prey in relation to wolves Alpha (α), beta (β) and delta (δ) using the following mathematical formulas:

$$\vec{x}_1 = \vec{x}_\alpha * \vec{A}_1 - \vec{D}_\alpha \qquad (12)$$

$$\vec{x}_2 = \vec{x}_\beta * \vec{A}_1 - \vec{D}_\beta \qquad (13)$$

$$\vec{x}_3 = \vec{x}_\delta * \vec{A}_1 - \vec{D}_\delta \qquad (14)$$

The conceptual formulation for optimization is presented while taking into account the grey wolf pack's following steps [14]:

- Choosing and following the target;
- Surrounding the target;
- Moving closer to the target;

As a result, the first step is to determine which three best ECs will guide the pack in its search for the best Edge to store the resource. The Alpha (α) is therefore thought to be the best EC. Then it is determined which one is the Beta (β) and which one is the Delta (δ). The final EC will be designated as Omega. **To implement the GWO optimization algorithm model, hunting is guided by solutions *α*, *β* and, *δ*.** The equations 15 and 16 were proposed by [15] to mold mathematically the behavior of wolves while chasing their Prey.

$$ED = |CV\,2 * Pr(x) - P(x)| \qquad (15)$$

$$P(r+1) = Pr(x) - CV\,1 * ED \qquad (16)$$

Where x denotes the current iteration, Pr denotes the resource's value variable, P denotes the Edge Computing values, and CV 1 and CV 2 are the coefficients used in the equation for movement. As a result, the distance between the prey (resource) and the wolf (Edge) is calculated, with each iteration decreasing. CV 1 and CV 2 are calculated using the equations 17 and 18, respectively:

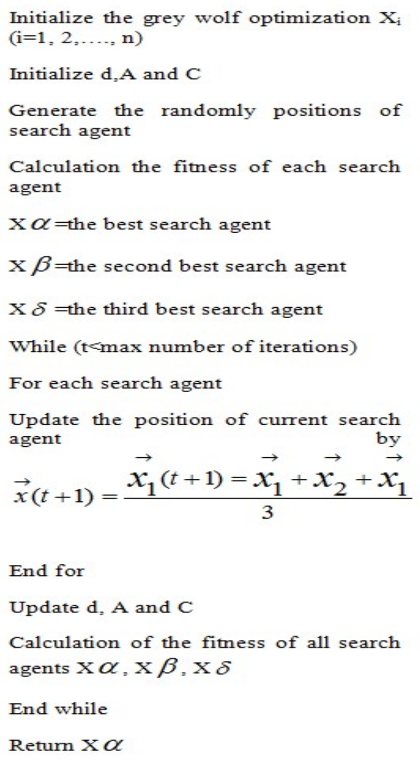
$$CV\,1 = 2a * t1 - a \qquad (17)$$

$$CV\,2 = 2 * t2 \qquad (18)$$

During the iteration, an is decremented from 2 to 0, and t1 and t2 are random variables with values ranging from 0 to 1. In each iteration, something like an is in charge of getting the wolf to approach the prey. According to equation 19, the best edge computing is updated with the sum of the three best positions.

$$\vec{x}(t+1) = \frac{\vec{x}_1 + \vec{x}_2 + \vec{x}_3}{3} \qquad (19)$$

GWO Algorithm:



```
Initialize the grey wolf optimization $X_i$ (i=1, 2,...., n)
Initialize d,A and C
Generate the randomly positions of search agent
Calculation the fitness of each search agent
X$\alpha$=the best search agent
X$\beta$=the second best search agent
X$\delta$ =the third best search agent
While (t<max number of iterations)
For each search agent
Update the position of current search agent by
$\vec{x}(t+1) = \frac{\vec{x}_1(t+1) = \vec{x}_1 + \vec{x}_2 + \vec{x}_1}{3}$
End for
Update d, A and C
Calculation of the fitness of all search agents X$\alpha$, X$\beta$, X$\delta$
End while
Return X$\alpha$
```

2.3. Lion Optimization Algorithm

The inspiration for the proposed meta-heuristic is first discussed in this section. Following that, the Lion Optimization Algorithm (LOA) is introduced.

Initialization: The LOA is a population-based meta-heuristic algorithm, with the first step being to randomly generate the population over the solution space. Initially, the population is generated arbitrarily across the solution space. Every outcome is referred to as a lion (edge devices). In a d-dimensional optimization dilemma, a lion (edge devices) is denoted as d set of key differentiators, as:

Lion (edge devices) = $[l_{1,,,,,,}l_d]$ (20)

Every edge device (lion fitness)'s range is determined by evaluating the objective function given in (16) as:

f(edge device) = $f(l_{1,,,,,,,,,,}l_d)$ (21)

In the first stage, dpop solutions are generated arbitrarily in exploring space, and a percentage d of completed results is chosen as migrant edge devices arbitrarily. The remaining population is randomly divided into the prides. Each solution had a distinct gender and remained consistent throughout the optimization task. While searching, each lion looks at its most excellent entered site. Each pride's region is built around such observed locations. As a result, observed sites (the best entered sites) generate that pride's region through its representatives for each pride[15].

Hunting: A certain number of females in each P seek prey in a group to feed the members of P. These hunter lions use specific tactics to encircle and capture their prey. During hunting, the lions generally follow a similar pattern. Stander (1992) classified lions into seven distinct groups based on their roles. While hunting, each lioness adjusts its location based on the location of its own or other members. As a result, some of the hunter lions encircle the prey and attack it from the opposite direction; the hunters are divided into three "wings." The centre wing has the greatest cumulative fitness, while the left and right wings are fixed at random. When a hunter improves its fitness, the prey flees to a new location, as shown in Equation (22).

$$PY' = PY + rand(0,1) \times py1 \times (py - Hunter) \quad (22)$$

Where PY denotes the current location of the prey, Hunter denotes the new location of the hunter, which attacks the prey, and PYI denotes the percentage of fitness improvement in the hunter. The following is the new location of the hunters from the left and right wings.

$$Hunter' = \{rand((2\times py - hunter), py), (2\times py - hunter) < py$$
$$= \{rand(p, (2\times py - hunter)), (2\times py - hunter) > py$$

(23)

The new locations of the center hunters are created as defined in the Equation (24).

$$Hunter' = \begin{cases} rand(hunter, py), hunter < py \\ rand(py, hunter), hunter > py \end{cases}$$

(24)

R and (a, b) generates a number between a and b at random. This hunting behaviour has some advantages in terms of finding better solutions. This mechanism creates a circle-shaped neighbourhood around the prey, causing the hunters to approach it from different directions. Furthermore, because some of the hunters occupy opposing locations, this process allows solutions to escape from local optima.

Roaming and mating: The system's ability to roam improves local search capability and aids in the discovery of workable solutions (task scheduling solution). Within the pride's territory, each resident male lion can be found. If the male lion discovers a better location than the current one while roaming, it must update the new location as the best frequented location [16]. Furthermore, Ma% of the female lions in the pride P mate with one or more resident male lions, resulting in offspring. Using equations (20) and (21), the mating procedure yields two offspring on average (2).

Offspringj 1 =β × female Lionj +Σ $(1-\beta)$ $\Sigma Si\ NR$ i=1 × male $Lionji$ × Si (1)

Offspring j2 = (1-β) × female Lionj ×female Lionj + Σ $(1-\beta)$ $\Sigma Si\ NR$ i=1 × male $Lionji$ × Si (2).

The number NR, which represents the number of resident males in a pride, is a completely random number with a standard error of 1 and a mean value of 0.1. Si=1 selects a male lion for mating; otherwise, Si=0. The mutation is then performed on the two additional offspring, one of whom is randomly identified as female and the other as male.

Movements towards Safety: Only a few female lions hunt for prey, while the rest stay in safe territory. The best positions for each territory are calculated and saved. A high victory rate indicates that the lions have deviated from their optimal point. Lower values indicate that lions are roaming for improvement, and thus competition evaluation indicates success.

Mating: Mating is an important procedure that ensures the survival of the lions and allows for data exchange among members. Female lions mate with one of the many resident males in each pride. To produce offspring, these males are arbitrarily selected from the same pride as the female. Nomad lions, on the other hand, only mate with one male who is chosen at random. The mating operator is a linear combination of parents for the purpose of producing two new offspring.

Defense: Lions value this type of behaviour. Mature male lions engage in combat with other lions. Losers either become nomads or flee the territory. When nomadic lions win a battle, they take over the territory of the loser. Thus, LOA defends lions in two ways: against newly matured resident males and nomadic males. LOA thus finds the strongest lion in the group.

Equilibrium: The number of live lions must be kept under control at the end of each iteration. As a result, female nomad lions are classified based on their fitness levels. The best females are chosen and distributed to prides to fill the vacancies left by the migrated females. In relation to the maximum number of female nomads, the weakest females are removed (1-S). Male nomads are also sorted based on their fitness values, and the lions with the least fitness will be removed with respect to the maximum number of male nomads%S.

3. PROBLEM STATEMENT

In edge computing, resource allocation is a critical challenge. The efficient allocation of constrained resources to competing services with various features and requirements, such that the edge system obtains maximum resource utilisation while also satisfying the services, is referred to as resource allocation.

- For starters, devices differ in terms of computing capabilities and the characteristics of the computing tasks they generate. The tasks have varying delay requirements, input data amounts, and computational complexities.
- Second, it's likely that devices in edge computing systems will function independently [17, 18].
- Third, the edge The Swedish Research Council funded part of the research through project 621-2014-6. Multiple heterogeneous wireless access points and edge clouds may be used in computing systems. As a result, device offloading decisions should be coordinated so that resources are efficiently utilized while considering the interests of individual devices, the heterogeneity of their tasks, and interactions with the resource allocation policies of the edge cloud providers. This makes joint management of wireless and computing resources for edge computing inherently difficult.

Edge cloud computing has undoubtedly provided numerous benefits; however, it still faces significant challenges that must be addressed. The first challenge in such a complex environment with a variety of resources is task scheduling. In this complex environment, this challenge typically considers real-time execution, as well as a massive amount of flowing data. As a result, a scheduling approach that is effective in completing all tasks on time, resulting in real-time execution, is required. The second challenge focuses on processing business workflows with the goal of completing task scheduling while taking QoS (Quality of Service) requirements into account (for example cost and deadline).

However, the task offloading time is in the tens to hundreds of milliseconds range. The task offloading time for some delay-tolerant services can reach a few seconds. Without considering time-varying fading, offloading strategies will be inaccurate, resource utilisation will be reduced, and task delay requirements cannot be guaranteed. Furthermore, when time-varying channels are considered, the task transmission time is related to the vehicle's location and the allocated bandwidth. As a result, allocating bandwidth for time-varying channels is a critical and difficult issue.

3.1. Problem solution: Based on the aforementioned issues, this paper investigates the Cloud-Edge system, which is motivated by minimizing delay, energy consumption, and computing cost, and it constitutes the task offloading and task scheduling problem for multiple end users in order to maximize system utility. This paper's contributions are summarized below*:*

1. Task scheduling is a traditional method for transferring tasks to external platforms due to device limitations in computing power, storage, and energy. It can boost computing efficiency, shorten task completion times, and make better use of other devices' resources. Because the edge computing network has the characteristics of ultra-dense deployment and simultaneous access by a large number of users, the selection of user computing task mode is critical and directly determines the system's computing time and cost. This paper proposes a task scheduling and task offloading scheme based on Cloud-Edge, including computational task scheduling and task offloading strategy optimization, to address the utility maximisation problem.
2. The mixed integer nonlinear programming problem is used to describe the joint task offloading and task scheduling problem, which combines task offloading decisions and resource allocation for offloading users to maximise system utility.

3. For the task offloading and task scheduling problem, a Hybrid Greywolf Lion Optimization Algorithm based on cloud edge computing system was proposed to obtain the task offloading strategy that achieves the best results in terms of Makespan, Energy latency, and CPU utilisation. Energy consumption, system utility, task completion time, execution delay, and convergence rate are all factors to consider.
4. Different resource allocation and task offloading schemes are used as comparison schemes for the hybrid greywolf lion optimization algorithm, and simulation experiments are run with different parameters. The results show that the offloading scheme proposed in this paper significantly improves users' offloading utility**.**
5. All computing tasks are offloaded and processed at the Edge. Offloading is typically translated into a simple resource allocation problem, with **tasks executed on virtual machines or containers at the Edge. Energy savings at the end device can be maximized**, but other sources of energy dissipation, such as the device's transmission power, must be considered..

4. PROPOSED ALGORITHM OF HYBRID GREYWOLF LION OPTIMIZATION (HGWLO):

Figure 3 depicts an emerging technology that performs data analytics and storage close to the data source (i.e., virtual machine) to reduce network latency. Edge computing provides computing, storage, and network services from a cloud data centre. When compared to cloud computing, edge computing enables decentralised computation and storage. Devices for real-time communication can be supported by resources nearby the user. The primary goal of edge computing is to process data, reduce latency, and provide real-time response. Task scheduling is used to connect tasks to computation resources and plan their execution while satisfying task dependency and meeting resource management goals.

The optimizer attempts to group task scheduling in order to reduce overall flow task execution time while using the fewest number of cores. It is worth noting, however, that a core represents a VM, with a single core primarily used for application benchmarking. The next step in the edge cloud computing environment is to allocate resources and schedule scheduling tasks. ***There are five steps in resource management scheduling: a resource optimizer, a task scheduling plan generator, a task queue builder, a task execution time generator, and a cost estimator.*** The amount of resources and time required to run the task scheduling are estimated during the resource estimation stage based on task execution constraints such as deadline, throughput, and waiting time. As a result, tasks for scheduling are organised into execution groups. Next, the required task offloads for the group-based technique can simplify task scheduling in an edge cloud environment, using task offloading, it becomes clear that the network's Edge infrastructure creates an additional resource layer between the virtual and the external platform. This layer is capable of reducing energy consumption, transport, and cloud networks, thereby reducing communication delays, improving energy efficiency, and, as a result, extending the battery-powered device's lifetime. Given this, it is critical to schedule these data and large applications in these systems. Scheduling is an important aspect of task control in the cloud. The Scheduling process estimates the amount of resources required to complete the task and decides which tasks should be assigned to which computing component. Before processing the subtasks in parallel, they can be broken down into smaller subtasks. The overall advantage of the implementation is increased by breaking a computation into smaller subtasks and implementing these subtasks on different processors. Furthermore, the goal of the task scheduling algorithm is not to schedule the entire task into the available processor in order to increase profit (profit here refers to the combination of low cost, low memory utilisation, and energy conservation) without affecting the primary requirements. It makes task scheduling difficult. The next step in task offloading is to optimize the offloading of computation-intensive tasks from the end user device to a remote site while keeping various computational, communication, and mobility constraints in mind. This procedure entails application partitioning, offloading decision making, and distributed task execution**.**

The powerful processing power of the edge server can accelerate task computing, reduce task completion time, and save energy for devices.

Devices that use task offloading do not need to have a lot of computing power or storage space. As a result, even in the presence of limited computing power and other hardware constraints, edge users can complete computing tasks.

Because the cloud provider must maintain a large number of users, the scheduling technique alleviates the cloud provider's burden. This study proposed a new algorithm based on greywolf and Lion Optimizer to schedule tasks and offload in cloud edge computing. ***The proposed HGWLO algorithm assists in locating the best VM for task allocation.*** The HGWLO algorithm generates the evaluation of fitness values for each wolf. HGWLO-based approaches assume the random deployment of a number of wolves (search agents) with a random initial position and the ability to change their positions. The three best candidate solutions are considered as alpha, beta, and delta wolves in each iteration based on the predefined number of iterations. To find better solutions, the omega wolves had to encircle the alpha, beta, and delta wolves.

Begin populating our goal in this investigation is to determine the assignment planning for distributed computing and to limit the make span of the arrangement, which is the longest consummation time for all projects. In this manner, we should outline a basic answer for a lion. A lion communicates with a project planning arrangement and schedules the task in cloud edge by mapping cloud tasks (cloudlets) to cloud assets (virtual machines (VMs)). The object that provides the requested services and is closest to the best search optimal position is the final solution.

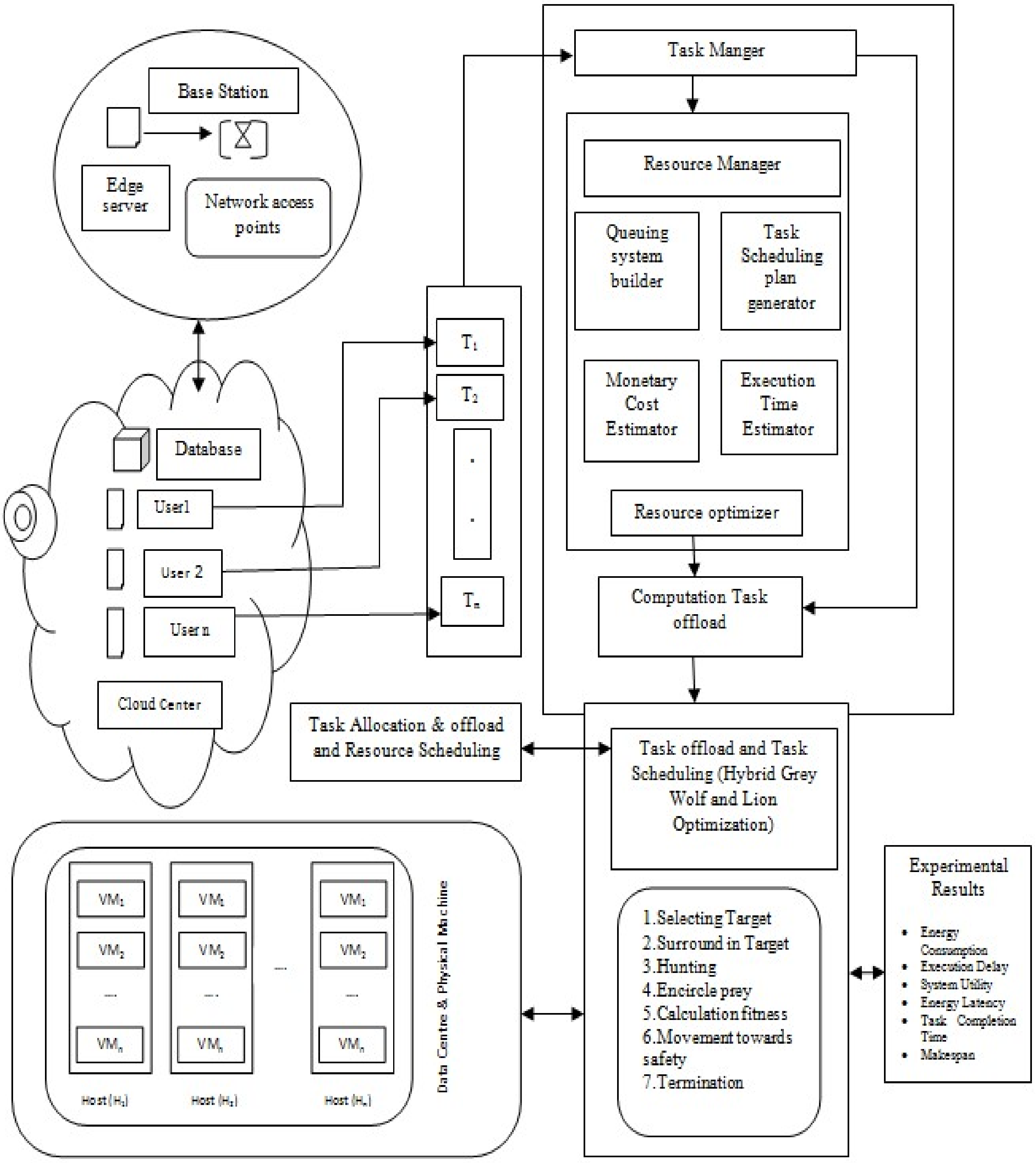


Figure 3: Proposed Model Hybrid Grey Wolf Lion Optimizer (HGWLO)

The proposed algorithm's details are provided below. Figure 3 depicts the general framework for this proposed model. This is a process that occurs when using a constrained task based on the operation that the virtual machines are to perform. The scheduler collects data from the Cloud User and edge server and then computes it to make a decision that assigns each task to its respective virtual machine. We use makespan, system utility, energy consumption, execution delay, energy latency, and task completion time among VMs as performance metrics in this research proposed method to optimise task and resource, using a Hybrid grey wolf lion optimizer (HGWLO) algorithm based on the proposed model is scheduling the task and offload in cloud edge computing.

Proposed algorithm

To schedule task offloading environments, the proposed task offloading framework employs a hybrid Grey Wolf Optimizer (GWO) and Lion Optimization algorithm. The current system addresses competing goals, namely make span. As a result, the task offloading model is intended to schedule tasks in the cloudedge computing environment. The manufacturer claims that the makespan improves user satisfaction by maximising resource utilisation via the minimization function.

Input:
Requirement of HGWLO algorithm
Specification of Task Scheduling & task allocation.
Tasks T1, T2,……..TN and Resources R1, R2,…….RM and Max_{itr}
Output: Set of tasks allocate to $\text{vm}_{\text{s,}}$ decrease the makespan of total scheduled tasks

```
Procedure: HGWLO
Initialize the number of task , number of resources , Set the initial values and the maximum number of iterations Maxitr
Step1:Set t = 0 {counter initialization}. //population initialization
Step2: n= 1 . Number of task
Step 3: **while** n< Max **do** . Executes within the maximum number of task
 Step4: Bring about an initial population randomly.

Step 5: Calculate the each wolf agent I and initialize the lion optimization pheromone matrix
Step 6 : **for** i = 1, 2, . . . ,m **do**
Step 7 : **for** j = 1, 2, . . . ,m **do**
**Step 8:Estimate objective function(makespan)**
```

Makespan Consider p(*i*,*j*) as the offloading task execution time on *i*th VM of *j*th cloudlet task and the completion time of offloading task is represented as *L*(*i*) = *Σp*(*i*,*j*) for *i*th VM. The objective function for scheduling is regarded as *Lmax*= max(*Li*), which is the cloud-VM makespan. The proposed method schedules the *n* tasks on *m* cloudlets VMs such that *n* > *m* and the makespan minimization problem is considered. The upper limit for the minimization function is defined as *m* × *Lmax*. Consider *G* as the scheduling makespan, where *m* × *Lmax*> *G*, where execution time of assigned task takes a time of *Lmax* to each cloudlet VMs. The relation *m* × *Lmax*> *G* holds true for all scheduling makespan and thus optimal makespan (*O*) is formulated by modifying the relation based on *m* × *O* > *G*. The task offloading execution time *t* *T* is considered as the longest time required for input data reception, which is represented as *Tw* in Eq. (1) with the processing time of *t* represented by *Te* in Eq. (2). max*i*(*t*) expressed in Eq. (3) is regarded as the largest data received at task offloading *t* from its subtasks and *B* is regarded as the execution delay of cloudlet VMs, which is calculated in Ms.
The *MI* in Eq. (2)
The cost is also computed based on cost per process, storage and transfer based on MIPS processing, per second hosting time and mbps data transfer between the cloudlet VMs is the task size expressed in MIPS and it is the cloudlet processing power. The offloading task completion time is the sum of *Tw* + *Te* and the cloudlet VMs completion time is given as *VMi* (*i* = 1.*m* with *m* as the total cloudlet VM availability), which is expressed in Eq. (3) and it is the sum of offloading task completion time. The maximum completion time for a offloading task in cloudlet VMs is regarded as the makespan, which is expressed in Eq. (4).offloading calculation selection
problems in the edge cloud server calculation offloading decision.

$$T_w = \frac{\max_i(i)}{B} \tag{1}$$

$$T_e = \frac{MI(t)}{MIPS(vm(i))} \tag{2}$$

$$T_t = \sum_{t \in T}(T_w + T_e) \tag{3}$$

$$makespan = \max(T_{t-1t}, T_{t-2t}, \ldots, T_{t-mt}) \quad (4)$$

Step 9: Calculate the probability of selecting the remaining points when the lion optimization departs from point i. Select the departure alternative with the highest probability ki, and update the next city selection for lion i as ki

Step 10:Move Towards Safer Place
position for female lion (FL) is expressed as

$$FL = FL + 2D \times random(0,1)\{R1\} + U(-1,1) \times \tan(\theta) \times D \times$$
$$random(0,10\{R2\}$$
$$\{R1\}.\{R2\} = 0, \| R2 \| = 1$$

where FL is the current place of FL, D provides the distance among the FL's position as well as chosen point selected by the tournament chosen in the pride region. {R1} is a vector that indicates the primary point is the earlier place of the FL, and it is focused on the elected position. {R2} is perpendicular to {R1}.

Step 11:For each Nomad lion
Both male and female move randomly in the search space Identify their new position as,

$Lion=\{Lion, if \quad rand>prRAND, otherwise$

$pr=0.1+m(0.5Nomad-BestNomadBestNomad)$

Where, rand is a random number between 0 and 1, pr is a probability, Nomad is the fitness value of the current nomad, and BestNomad is the best fitness value of the nomad lions. %M of females mate with only one male Nomad males attack prides

Step 12: For each pride,
Nomads are %I of the pride that is immigrated.
Do

(i)Every male and female lions under nomad category are sorted according to their fitness score.

(ii) Female lions faring above fitness score are selected and disseminated to prides, filling out the empty positions.

(iii) Lions faring below fitness score are taken out in accordance to maximum permissible count under each gender

Step 13: Set t=t+1 (iteration counter increasing)

Step 14: Termination criteria

Step 15: until ($t<Max_{itr}$). (Termination criteria satisfied)

Step 16: Otherwise go to step2

Step 17: Produce the optimum solution

5. EXPERIMENTAL RESULTS

The algorithm is simulated in this experiment using CloudEdgeSim. Assume that device generation follows the Poisson distribution. H= 50 M is the expected number of CPU cycles required for each task. The expected task transmission delay for a typical 100 Mb Fast Ethernet LAN is 100 ms. In this paper, we evaluate four Dragonfly algorithm algorithms. Grey wolf optimization, lion optimization, and our proposed HGWLO are all options. It is assumed in the simulation experiment that both devices and cloud edge servers can handle tasks. Additionally, it is assumed that no task queues will exist to cause time delays and that each task will be completed immediately upon arrival.

Table1: Simulation Parameters

Parameters	Values
Number of Hosts	3
Number of cloudlets	10-15
Number of cloud edge server	1-10
Number of vm_s	5-15
Number of Users	10-90
Number of Nodes	10,20,30,40,50,60,70
Storage	1 TB
RAM	2 GB
Processing Speed	50-300 MIPS (Millions of Instruction Per Second)
The input data size of the tasks	[600, 1200] KB
The number of CPU cycles	*Cm* = [500, 1000] Megacycles
The Bandwidth resources	[180, 230] KB

A).Makespan: Makespan calculates the maximum completion time by indicating the last task's completion time. The most popular optimization criterion for task scheduling in cloud edge computing is to minimise the makespan. It can be calculated using the equation below:

$$\textbf{Makespan}=\textbf{max}_{\textbf{task i}}(\textbf{Fn}_{\textbf{time}})$$

In which case, Fn Time indicates the completion time of task i. The HGWLO algorithm is compared to Greywolf, Dragonfly, and Lion Optimization algorithms for computing and minimizing make span in this section. The total time elapsed from beginning to end is known as the make span. In the context of scheduling, the

term is frequently used. A large project has been broken down into several sub-tasks.
To begin, the proposed algorithm assigns tasks to cloudlets that are only one hop away from the user, resulting in a shorter makespan than when tasks are assigned to VMs on cloud servers. In comparison to existing methods Greywolf, Dragonfly, and Lion Optimization, the proposed algorithm assigns tasks to the VM with a lower value of load and execution time, resulting in a shorter makespan. The make span is the total amount of time required by the tasks to complete the execution. Makespan is assessed and analyzed by varying the task and resource values. The proposed HGWLO algorithm produces the lowest makespan values when compared to Greywolf, Dragonfly, and Lion Optimization. **In Figure 4, the HGWLO method achieves the shortest make span time.**

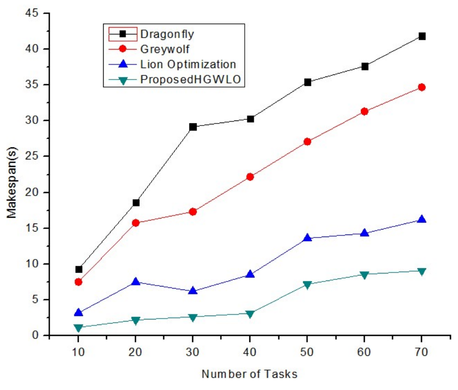


Figure 4: Make span

B) Comparison of System Utility and Number of Users: The overall utility of the system decreases as the total number of user equipment increases, because as user equipment increases, more and more tasks must be uninstalled. On the one hand, as user count rises, ***each user uses fewer and fewer communication resources; as a result, offload rate falls and offload delay rises, reducing system utility.*** However, as the number of users grows, so do the computing resources that each user can allocate, resulting in increased latency and, as a result, a decrease in overall system utility. Figure 5 shows that the overall system utility of the resource scheduling scheme based on is greater when compared to ***pre-existing Greywolf, Dragonfly, and Lion Optimization algorithms with our proposed HGWLO, indicating the algorithm's superiority.***

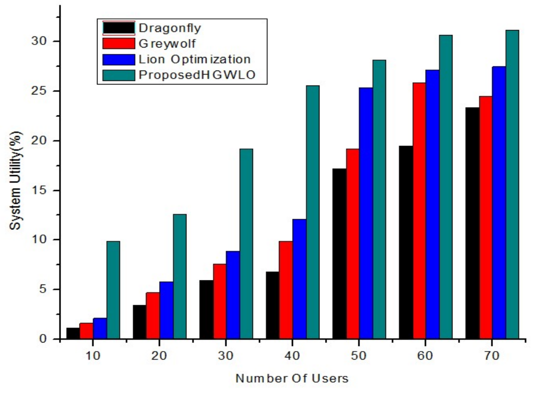


Figure 5: System utility

C) Convergence Analysis: Figure 6 depicts how the number of iterations decreased the make span. It demonstrates that the quality of task scheduling improved with each iteration. However, previous schemes dragonfly, greywolf, and lion optimization were found to have premature convergence, whereas HGWLO provided faster convergence as well as a reduction in makespan than dragonfly, greywolf, and lion optimization. The preceding demonstrates that the proposed HGWLO is appropriate for handling task scheduling and offloading in a cloud edge environment.

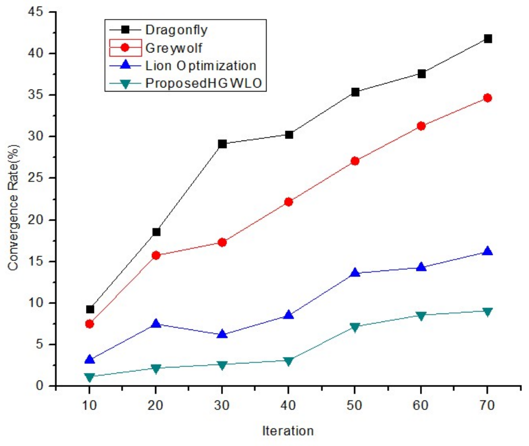


Figure 6: Convergence Analysis

D) Energy Consumption: Figure 7 depicts the results of comparing the number of tasks under various energy consumption constraints. M is the number of devices, and N is the number of cloud edge servers. The total system cost of offloading decreases as energy consumption constraints increase. Dragonfly performs significantly worse than the other greywolf, lion optimization, and proposed HGWLO when the energy consumption constraint is 10J/ms. When the energy consumption constraint is set to 40J/ms, the cost of dragonfly optimization begins to be lower than that of lion optimization, owing to the increase in energy consumption constraints and the model's decreased sensitivity to energy consumption constraints. As a result, suitable energy consumption constraints must be chosen. Under all energy consumption constraints, the proposed HGWLO algorithm outperforms the existing greywolf, lion optimization, and dragonfly algorithms. When the energy consumption constraint is near 70J/ms, the effects of dragonfly, greywolf, and lion optimization differ significantly. In contrast to the HGWLO. When compared to other algorithms, the proposed HGWLO algorithm significantly reduces network energy consumption, according to the findings. Only a subset of tasks is responsible for achieving balanced energy consumption.

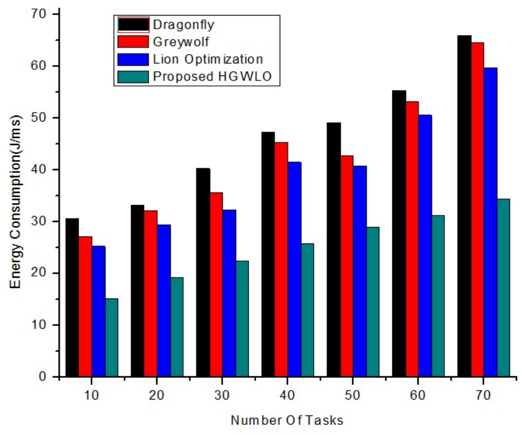


Figure 7: Energy consumption

E) Execution delay: Figure 8 compares the execution delay in the number of devices when the number of devices is different. When the number of devices is 10, there is no discernible difference in the execution delays of HGWLO and Lion Optimization. Because the number of devices increases and the amount of data to be processed increases, so does the execution delay. The extension will grow in length. When the number of devices is less than 50, the algorithm execution delays are comparable. HGWLO, on the other hand, has a slightly lower execution delay than the other two algorithms. When the number of devices exceeds 60, the execution delay of dragonfly begins to increase sharply, which is much greater than the execution delay of greywolf and lion optimization, HGWLO. When the number of devices is 70, the proposed HGWLO outperforms previous methods greywolf, lion optimization, and Dragon fly algorithms in terms of execution delay. Furthermore, testing to see if the proposed HGWLO algorithm can solve the scheduling and offloading problem in the cloud edge environment.

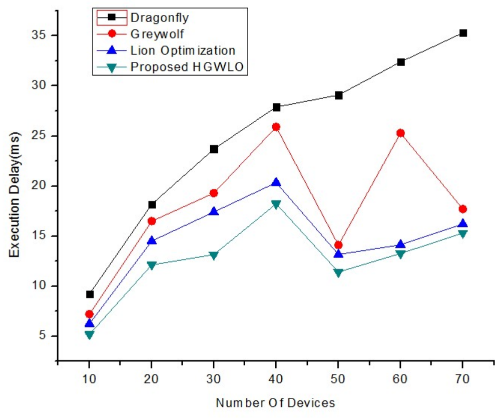


Figure 8: Execution Delay

F) Energy Latency: When all subtasks are processed at edge servers, latency is significantly reduced when compared to sending them to cloud servers for execution. However, sending all subtasks to the cloud can cause both computing resources and network bandwidth to become congested, resulting in increased queuing latency and, eventually, total latency. To reduce total latency, offload workloads that do not need to be executed locally on devices to edge servers with sufficient computing power, which are closer to the local device than the remote cloud. Figure 9 depicts the energy latency of the proposed HGWLO algorithm in comparison to other pre-existing greywolf, lion optimization, and dragonfly algorithms. According to the results, as the number of tasks increases, so does the scheduling latency in both the proposed and other algorithms. The proposed algorithm's dynamic task allocation helps to minimise latency more than other algorithms. The results demonstrate that the proposed HGWLO algorithm has lower latency than other algorithms. For various offloading tasks, say70, the latency is considered to be the highest in all existing algorithms, and it is higher in greywolf, lion optimization, and dragonfly algorithms than in the proposed

HGWLO. The same can be said for other offloading tasks ranging from 10 to 70.

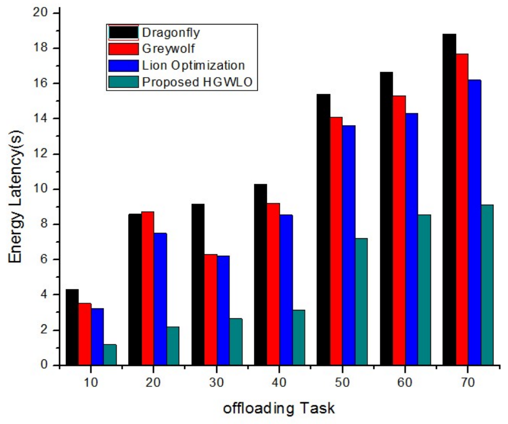


Figure 9: Energy Latency

G) Task Completion Time: The completion time is the time elapsed between the beginning and ending times. Figure 10 depicts the iterative process of combining Dragonfly, grey wolf, and lion optimization, as well as the proposed HGWLO, to achieve the best total time and make span. The proposed HGWLO algorithm outperforms the Dragonfly, grey wolf, and lion optimizations in terms of completion time. Each algorithm's completion time increased as the task volume increased. The completion time for each algorithm varies between 10 and 70 tasks. Based on this observation, the proposed HGWLO algorithm completes the process in less time. Furthermore, they precisely allocated resources on the cloud edge server to offload a better scheduling operation.

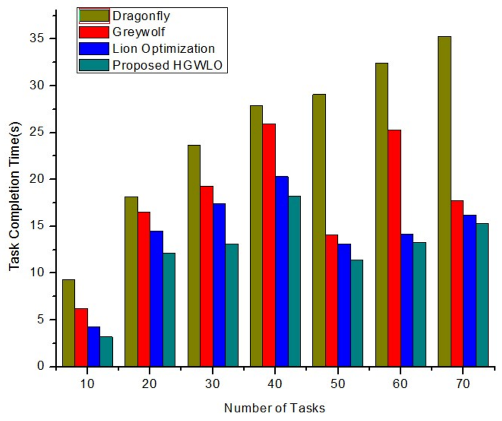


Figure 10: Task Completion time

6. CONCLUSION

The goal of task allocation problems in a multi-user network under cloud edge computing is to optimize makepan and system utility. Because the optimal solution to these problems is difficult to obtain. However, according to the experimental results, our proposed scheme Hybrid Grey Wolf Lion Optimization (HGWLO) effectively solves computational resource and task offloading and computational resource allocation problems. When compared to previous algorithms such as dragonfly, grey wolf, and lion optimization, the proposed method reduces makespan, energy latency, energy consumption, system utility, task completion time, execution delay, and convergence rate while ensuring better performance.

REFERENCES

[1] . Suzhi Bi and Ying Jun Zhang. 2018. Computation Rate Maximization for Wireless Powered Mobile-Edge Computing With Binary Computation Offloading. IEEE Transactions on Wireless Communications 17, 6 (2018), 4177–4190. https://doi. org/10.1109/TWC.2018.2821664

[2] . Shi, W.; Zhang, X. Edge Computing: State-of-the-Art and Future Directions. IEEE J. Comput. Res. Dev. 2019, 56, 69–89.

[3] . Cheng, N.; Xu, W.; Shi, W.; Zhou, Y.; Lu, N.; Zhou, H.; Shen, X. Air-ground integrated mobile edge networks: architecture, challenges and opportunities. IEEE Commun. Mag. 2018, 56, 26–32.

[4] . Bagchi, S.; Siddiqui, M.B.;Wood, P. Dependability in edge computing. Commun. ACM 2020, 63, 58–66.

[5] . Bokhari, M.U.; Shallal, Q.; Tamandani, Y.K. Cloud computing service models: A comparative study. In Proceedings of the IEEE International Conference on Computing for Sustainable Global Development, New Delhi, India, 16–18 March 2016.

[6] . Al-Dhuraibi, Y.; Paraiso, F.; Djarallah, N.; Merle, P. Elasticity in cloud computing: state of the art and research challenges. IEEE Trans. Serv. Comput. 2018, 11, 430–447.

[7] . M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based

cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[8] . Zhang, M.; Zhou, Y.; Quan,W.; Zhu, J.; Zheng, R.;Wu, Q. Online Learning for IoT Optimization: A Frank-Wolfe Adam-Based Algorithm. IEEE Internet Things J. 2020, 7, 8228–8237. [CrossRef]

[9] . Cui, X.; Shan, N.; Li, Y. A Multilevel Optimization Framework for Computation Offloading in Mobile Edge Computing. IEEE/ACMTrans. Netw. 2020, 2020, 4124791.

[10] . M. Z. A. Bhuiyan, J. Wu, G. Wang, T. Wang, and M. M. Hassan, ``E-sampling: Event sensitive autonomous adaptive sensing and low-cost monitoring in networked sensing systems," *ACM Trans. Auton. Adapt. Syst.*, vol. 12, no. 1, pp. 1_29, Mar. 2017.

[11] . Wikelski M, Moskowitz D, Adelman JS, Cochran J, Wilcove DS, May ML (2006) Simple rules guide dragonfly migration. Biol Lett 2:325–329

[12] . Russell RW, May ML, Soltesz KL, Fitzpatrick JW (1998) Massive swarm migrations of dragonflies (Odonata) in eastern North America. Am Midl Nat 140:325–342

[13] . S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46 – 61, 2014. [Online]. Available:http://www.sciencedirect.com/science/article/pii/S0965997813001853

[14] . Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Advances in engineering software 69, 46–61 (2014)

[15] . S. Saranya and 2N. Sabiyath Fatima ,"Efficient Handling of Medical Data Classification in Cloud-Edge Network using Optimization Algorithm", Journal of Computer Science, 2021, 17 (11): 1116.1127

[16] . A.Tamilarasi, A.Abarna, K.Chitra, K.Nagendhiran,R.Aarthi, " Effective Data Clustering Using K Means Along with Lion Optimization Algorithm" ISSN: 2005-4238 IJAST, Copyright, 2020 SERSC

[17] . L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," ACM SIGCOMM CCR, vol. 44, no. 5, pp. 27–32, 2014.

[18] . P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," ACM SIGCOMM CCR, vol. 45, no. 5, pp. 37–42, 2015.