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

Optimizing Multi-Objective Computation Offloading In Heterogenous Environments Using Adaptive Offloading Cat Hunt Optimization Algorithm

M.Jyothirmai[1*], Dr. Kesavan Gopal[2], Dr.M.Sailaja[3]

[1*]Research Scholar, JNTUK Kakinada
[2]Professor LPU Punjab
[3]Professor JNTUK Kakinada

Abstract
The advent of smart mobile devices has ushered in a new era of computing, but their performance is inherently constrained by factors like processing power and battery capacity. To optimize task execution, it is critical to strike a balance between tasks executed on the devices and those offloaded for remote processing, as proper offloading greatly enhances the quality of service. Prevailing techniques often emphasize on single objectives and make computationally complex, lacking a universal approach that balances targets and complexity effectively. To tackle these issues, this research proposes an Adaptive Offloading Cat Hunt Optimization (AOCHO) algorithm, which is designed to optimize computation offloading in mobile edge computing, with a primary focus on minimizing time, energy consumption and resource utilization. Primarily, this research starts by formulating the problem using Directed Acyclic Graphs (DAGs) in heterogeneous environments, aiming to reduce energy consumption for mobile users. Subsequently, the AOCHO-based offloading algorithm tackles multi-objective problems. The experiments conducted in the MATLAB environment, yield superior results. The simulation demonstrates a substantial reduction in delay by 0.0172 sec, a decrease in energy consumption by 0.251 (10^{-3} J), and a cost reduction of 0.387. These results clearly reveal that the proposed algorithm surpasses other benchmark algorithms in various situations. This underscores the algorithm's effectiveness in enhancing offloading efficiency for mobile devices in the realm of mobile edge computing.

Keywords: Offloading mechanism, Multi-objective, Edge computing, Direct acrylic graph, Heterogenous environment

1.Introduction

The expansion of wireless network and the Internet of Things (IoT) has piloted in a new era for smart Mobile Devices (MDs), making them a prominent platform for various purposes, such as face recognition, video surveillance and natural linguistic processing [1]. These applications demand significant computational power, creating a dilemma as MDs are often resource-constrained. This incongruity between the capabilities of MDs and the demands of computation-intensive applications poses a significant hurdle in ensuring a satisfactory Quality of Experience (QoE) [2][3]. Computation offloading plays a pivotal role in addressing this challenge, initially gaining traction in cloud computing and subsequently finding applications in edge computing [4]. It enables MDs to offload computational tasks to remote servers for execution. However, due to network limitations, not all tasks need to be offloaded [5]. This necessitates rapid decisions regarding which tasks are executed on servers and which ones are handled locally. Only when this decision is made wisely on the Quality of Service (QoS) is maintained or improved, ultimately enhancing the overall QoE [6][7]. While Mobile Edge Computing (MEC) has the probable to enhance MD application performance, the concurrent transmissions of multiple MDs are to compromise channel quality, leading to degraded transmission performance and increased response latency [8]. Employing multi-channel communication may mitigate this issue by allocating

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

different channels to MDs, yet it is important to note that band resources still proves insufficient to adapt all MDs effectively [9].

Cloud computing systems often transfer computation tasks, either partially or entirely, to cloud servers to improve the computational load on MDs. However, a significant challenge with this cloud-based offloading method is the often-unacceptable transmission delay, mainly due to the substantial detachment between clients and cloud servers [10][11]. In contrast, the concept of MEC resolves this concern by deploying servers or micro-servers in the immediate vicinity of the MDs [12]. This setup dramatically reduces transmission delays, making MEC an attractive computing model for several mobile applications. The challenges associated with mobile computing have accelerated the expansion of MEC, particularly within the framework of 5G architecture. Unlike traditional cloud computing, MEC covers cloud services from central cloud data centers to the networks. It permits User Equipment (UEs) to directly offload tasks to adjacent MEC servers, utilizing Base Stations (BS) for this purpose [13][14]. This approach not only accommodates the increasing demand for computational skills but also enhances the QoS of mobile applications by significantly reducing Energy Consumption (EC) and latency conditions [15].

The proposed AOCHO algorithm is to optimize the task execution in a heterogeneous environment by jointly considering task latency and EC. The algorithm aims to achieve this objective by effectively scheduling the execution of subtasks, taking into account their dependencies and execution constraints. By optimizing the scheduling sequence and decision of subtasks, the algorithm improves the parallelism of task execution and reduces the overall delay and EC and resource utilization. The ultimate goal is to enhance the effectiveness and competence of computation offloading in a heterogeneous environment, leading to improved user experience and resource utilization. The key contributions of this research are described as follows,

- ✓ The proposed AOCHO algorithm provides an advanced framework for optimizing mobile computation offloading. By combining the strengths of the Cat Hunting Optimization (CHO) with adaptive optimization techniques, this research offers a robust and efficient solution for addressing key objectives such as EC, delay reduction and cost optimization simultaneously.
- ✓ This research explores the difficulty of task offloading involving subtask dependencies and formalizes task dependency as DAGs in heterogenous environment. This take into account the diverse characteristics of MEC and the limited resources of UE and MEC servers.
- ✓ The AOCHO algorithm is designed to support real-time decision-making, a crucial capability for applications that demand low latency and immediate responses. This holistic approach ensures a balanced and well-rounded offloading strategy that aligns with the diverse requirements of mobile applications. It contributes to the overall efficiency, cost-effectiveness and sustainability of mobile computation offloading systems.

The research work is organized as follows; Section 2 explains some recent literatures based on computation offloading. Section 3 explains the system model and proposed algorithm. Section 4 covers the outcomes and discussions of the proposed algorithm. Finally, Section 5 ended up with conclusion and future scope.

2.Related works

In recent years, there are numerous researches undergoes to efficiently optimize the task offloading challenges in MEC. Among those few of them is listed as follows,

Linbo Liao et al. [16] presented a Double Reinforcement Learning Computation Offloading (DRLCO) model that aimed to diminish EC in the MEC platform. The scheduling-based algorithm was performed to resolve the issue of delay. Additionally, an adaptive prioritized experience replay algorithm was employed to enhance the performance of the system. The experimental results showed a reduction in both delay and EC when related to previous methods. ***Huan Zhou et al. [17]*** developed a technique constructed on Double Deep Q Networks (DDQN) to define the dual policy for resource allocation and offloading mechanism. This approach effectively approximated the Q-learning value function. Simulation results indicated that DDQN method significantly improved the performance in various scenarios while compared with other baseline methods. ***MOHAMMED S. ZALAT et al. [18]*** suggested an approach constructed on Niching Genetic Algorithm with a Markov Decision Process (NGA-MDP) to enhance the multisite offloading system. MDP was employed to ascertain the most ideal location for executing individual elements. NGA was utilized to control the optimum shift probabilities for mechanisms functioning across multiple sites. The experimental consequences indicated

that the NGA-MDP method consumed minimal energy, executes quickly compared with other methods. ***Si-feng Zhu et al. [19]*** presented a Multi-Objective Immune cloning Algorithm (MOIA) that efficiently addressed three optimization objectives. These objectives encompassed computing tasks, EC and server load balance. Furthermore, this approach conducted an extensive set of virtual experiments to validate the efficiency of the MOIA system. ***Xiangjun Zhang et al. [20]*** suggested a Deep Deterministic Policy Gradient (DDPG) method to deals with crucial unloading problems. This process aimed to enhance the point modification and amplitude by utilizing Reconfigurable Intelligent Surface (RIS. In the end, DDPG algorithm exhibited substantial performance improvements than non-RIS learning algorithms and other traditional algorithms. ***Mengxing Huang et al. [21]*** developed a Multi-Objective Whale Optimization Algorithm (MOWOA) to address the ideal computation offloading mechanism in MEC. The algorithm aimed to simultaneously reduce time and EC while enhancing QoS. Furthermore, an enhanced version, MOWOA2, utilizing the gravity reference point technique to attain a more diverse solution set. Experimental results demonstrated notable improvements in the quality of the final solutions. ***Sadoon Azizi et al. [22]*** presented a Deadline-aware and Energy-efficient Computation Offloading (DECO) method for arranging and handling of tasks created by IoT systems. This method took into interpretation task priorities and the edge servers during the task-node plotting progression. The outcomes validated the effectiveness of the suggested algorithm, which outperformed than other methods. ***Zheng-yi Chai et al. [23]*** recommended an efficient multi-objective evolutionary algorithm that concentrated on jointly optimizing delay, EC and cost objectives. It utilizing the NSGAIII-TOMEC algorithm in various situations relating numerous MEC servers, MDs and tasks. Simulation results demonstrated a substantial optimization in the offloading revenue of MDs when compared to other techniques. ***Xiao Chu et al. [24]*** developed a dynamic fine-tuning model using a Deep Q Network (DQN) to adjust the offloading proportions for each user, aiming to achieve a cost-effective MEC system. The task offloading model was represented as an MDP and computation offloading was implemented accordingly. Simulation results indicated a reduction in typical delay and average EC compared to alternative techniques. ***Si feng Zhu et al. [25]*** developed a Multi-Objective Immune Algorithm (NMIA) by building upon an enhanced evolutionary algorithm based on the principles of immune algorithms. NMIA was designed to efficiently generate a set of solutions that strike a balance among response time and EC. Experimental results demonstrated the capability of the NMIA approach to meet response time necessities and attain a more energy-efficient strategy in comparison to existing offloading schemes. The limitations of prevailing methods are presented in Table 1.

Table 1 Summary of prevailing methods limitations

Authors	Methods	Limitations
Linbo Liao et al. [16]	DRLCO	✓ high task generation rates ✓ Increased execution delay
Huan Zhou et al. [17]	DDQN	✓ limited resources of the MEC server ✓ Limited scalability
MOHAMMED S. ZALAT et al. [18]	NGA-MDP	✓ Lack of differentiation in network structure handling ✓ Lack of robustness and versatility
Si-feng Zhu et al. [19]	MOIA	✓ Lack of adaptability for real-time scenarios ✓ The objectives of this approach were not fulfilling the relevant aspects of MEC
Xiangjun Zhang et al. [20]	DDPG	✓ Increased transmission delay due to reduced available bandwidth ✓ Inadequate consideration of server computing resources.
Mengxing Huang et al. [21]	MOWOA	✓ Computation complexity was high ✓ Lack of convergence and diversity

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

Sadoon Azizi et al. [22]	DECO	✓ Lack of energy efficiency ✓ Limited mobility support
Zheng-yi Chai et al. [23]	NSGAIII-TOMEC	✓ inherent latency in sending tasks to remote servers for processing. ✓ This approach was not suitable for real time scenarios.
Xiao Chu et al. [24]	DQN	✓ overall offloading is not effective due to high time delay ✓ Lack of fault tolerance
Si feng Zhu et al. [25]	NMIA	✓ Lack of security and privacy concern ✓ High execution time

3. System model

In this research, the complexities of computation offloading within a confined edge network, such as those found in enterprise, campus, or home environments have been considered [26]. The designed Edge MEC system that centres around a single MEC server, offering computational support to MDs. As shown in Fig 1, The whole network involves of single cloud server with N edge servers, which is signified as $\{1,2....N\}$. All edge server covers k stations. Time is allocated with period ϕ and the set of time slot index is defined by $D = \{0,1,.....D\}$.

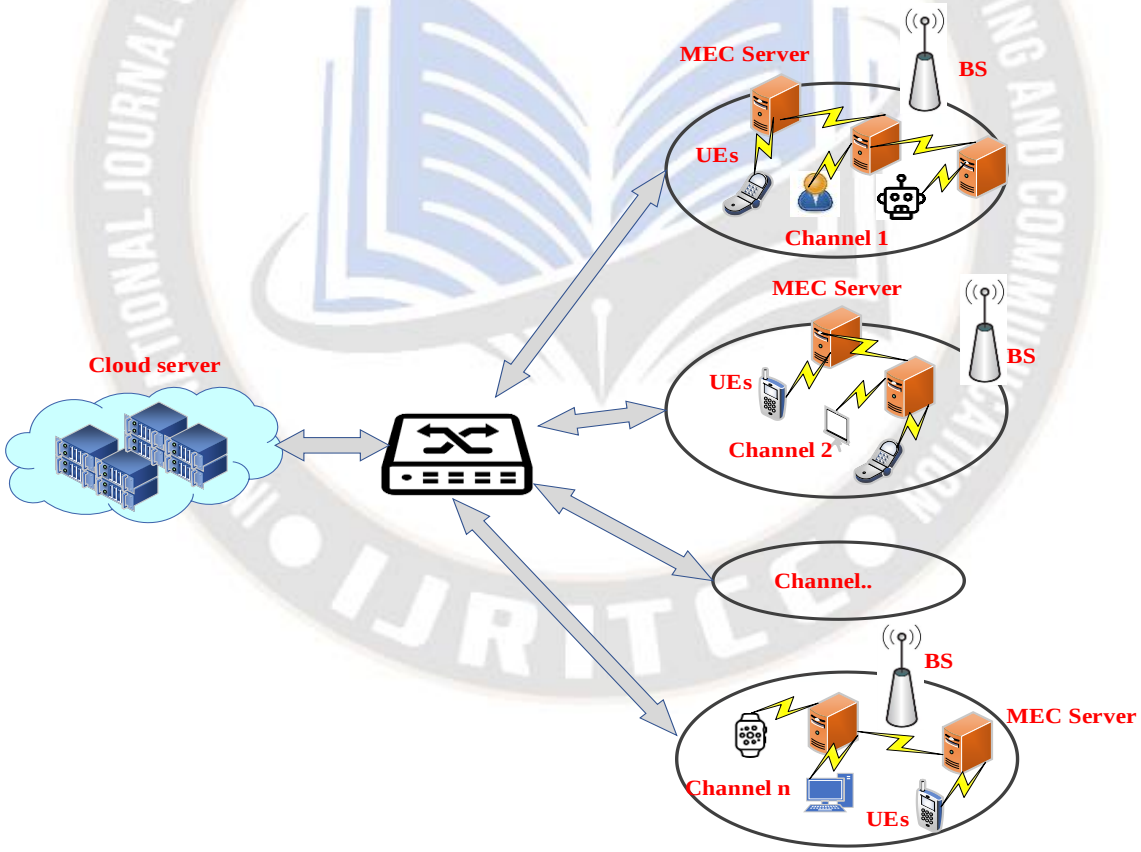


Fig 1 Overall System model

There are three kinds of devices are presents in the heterogenous environment, which is described as follows,

i. ***Cloud Server:*** Cloud servers possess significant computational power, making them well-suited for handling intricate computing tasks. This research primarily centres around task offloading within the context of end-edge association. Therefore, the

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

principal role of the cloud server is to leverage its loading and computational capabilities for aggregating and updating the decision model of the edge server.

ii. ***MEC server:*** The MEC server takes on the responsibilities of making offloading decisions, executing tasks and optimizing the decision model. Both of these functions rely on the deployment of a task scheduling element on the edge server for their execution.

iii. ***Terminal:*** The terminal has the role of communicating with the user, especially in specific scenarios. It generates tasks randomly and these tasks sometimes surpass the terminal's own computing capabilities. To minimize task latency and EC.

3.1 Directed acyclic graph application model

The application created by vH is combined of some tasks with dependence. The primary model utilizes a DAG for achieving fine-grained task offloading scheduling. This involves an analysis of the likelihood of parallel task processing, which enhances execution efficiency and aligns with a more practical approach. [27]. Fig 2 illustrates the application model formed by vH .There are $u_0, u_1 \in EC$ and there is direct connection from u_0 to u_1 is established.

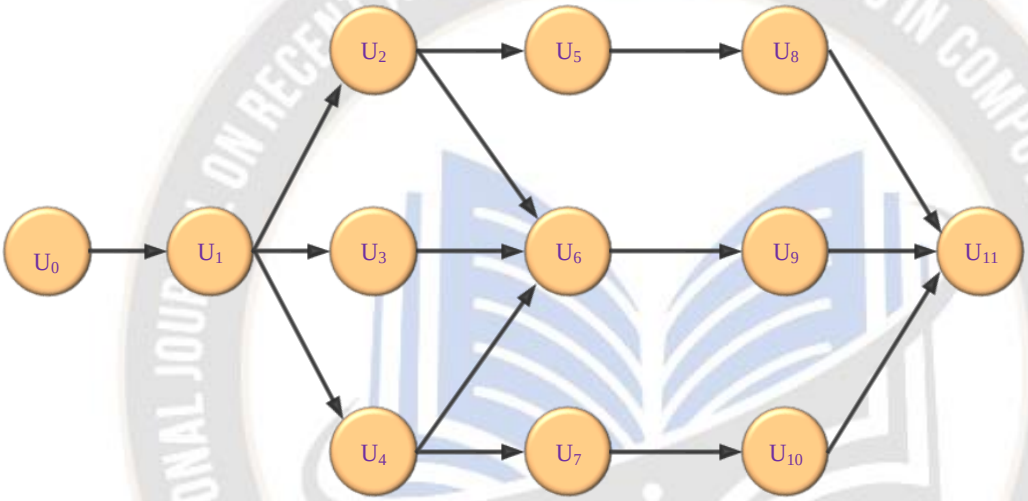


Fig 2 DAG model created by UE

The application model is represented as a DAG graph, where $_{vH_g}$ creates application $_{Z_g}$ .so, similarly $_{vH_g}$ denoted as $Z_g = U_g, H_g$, where $_{U_g}$ represents the generated tasks group and $_{H_g}$ represents the dependence connection among tasks. The task created by VH as $u\eta$, where $u\eta \in U_g$ .Assume, $u_r = \{w_r, d_r, \sigma_r, \mu_r\}$, where w_r represents the total workload of u_r , d_r represents the executing task data size, σ_r represents the relation of output to input data dimensions created by entity $_{vH_g}$. μ_r represents the highest delay a task tolerates amount. The binary variable $Q_{r,g,h} \in [0,1]$ designates, when the task was offloaded to the MEC server. Where, $Q_{r,g,h}$ indicates where the task is performed locally. Meanwhile assuming the single task is only be performed on individual place, consider $\sum_{h=0}^{P} Q_{r,g,h} = 1$. The main key notations expanded in the system model is provided in Table 2.

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

Table 2 List of key notations

Notation	Description
ϕ	Time is slotted with duration
$D = \{0,1,.....D\}$	Time slot index set
vH	User equipment
vH_g	Generates application of Z_g
U_g	Generated tasks group
H_g	Dependence relationship among tasks
$u\eta$	Task created by VH
u	Set of generated tasks
w_r	Total workload of u_r
d_r	Information amount of the task
σ_r	Proportion of output to input information sizes of tasks created by entity vH_g
μ_r	The determined acceptable task latency
$Q_{r,g,h}$	Local executed task
Bs	Base-station
Zi	Mobile device user i
Txi	i user's transmission rate
$\hat{\partial}$	Channel bandwidth
q_i	Transmission power
l_i	Channel gain between BS and user
ω^2	Background noise
N	Edge servers
$\{1,2....N\}$	Group of edge servers
k	Each edge server contains k terminals.
Z_l	Decision vector
g	Priority of the task
u_s	Amount of the task data in bits
d	The required Central processing unit (CPU) runs to execute one bit of a task.
D_p^{off}	Offloading delay
EC_p^{off}	Offloading EC
t^d	Distance among the MEC server and MD
t_0	Antenna far-field test distance
T_{dx}^g	Transmission delay through offloading
s^d	cost of uplink transmission
$S_p^{d,i}$	computation delay per one task
EC_p^d	EC of offloading computing system
M_p^d	size of the $u\eta$ executing task
R_p^d	uplink transmission rate of UE

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

$T^{g}_{r,i,trs}$	required time transmitting from UE to the MEC server
$T^{g}_{r,i,exe}$	required executing time from UE to the MEC server
u^{i}	task delay of user i

3.1.1 Communication Model

Primarily, the MEC server incorporating wireless contact in the communication model, is introduced. The BS of wireless access Bs survives the communications of MD users. The connection of wireless networks that Bs are signified as $N=\{1,2,3...N\}$ .Furthermore, $Z_l=(z1,z2.....zn)$ is a decision vector, where $Zi(i=1,2....x)$ signifies the MD user i offloads $Zi\times100\%$ of its task to be accomplished on edge servers and the remaining of the task which executed locally, that is $(1-Zi)\times100\%$ .In the system, the transmission rate Txi of user i is defined by using Eqn (1).

$$Txi=\partial\log_2\left(1+\frac{q_i l_i}{\omega^2-q_i l_i+\sum_{n=1}^{m}q_n l_n}\right) \tag{1}$$

where, ∂ represents the channel frequency, q_i represents the transmission capacity of user i, l_i is the channel gain among base-station and user i , ω^2 signifies the background noise. $-q_i l_i+\sum_{n=1}^{m}q_n l_n$ signifies the outcomes from other MD users.

The communication model described in Eqn (1) reveals that when a maximum quantity of MD users opts to offload their computations through the same channel simultaneously, it results in significant interference and subsequently lowers data rates, which has a detrimental impact on the performance of MEC.

3.1.2 Computation Model

The lacking loss of the computation task is symbolized as a tuple $h(g,u,d)$, where g denotes the priority of the task, u_s signifies the task information size in bits, d represents the essential CPU series to process a task per one bit. The offloading rate of the task is represented by $\Re\in\{0,1\}$, where $\Re=0$ indicates the local execution approach and $\Re=1$ indicates the offloading approach.

3.1.3 Local Computing

MDs dynamically regulate their CPU frequency according to the task requirements, impacting both task delay and EC. Where the mobile device CPU rate is represented by g_1^d and the local computation latency of the task i is denoted as $h^i(g^i,u^i,d^i)$, which is derived through Eqn(2),

$$S_p^{d,i}=\frac{H^i}{g_1^d} \tag{2}$$

where, $S_p^{d,i}$ signifies the computation delay per one task, H^i represents the essential CPU cycles to execute in one task and g_1^d signified by $g_1^d\le g_{\max}$, $d\in D$, the EC of task performing at the individual MD is represented by Eqn (3),

$$EC_p^{d,i}=\eta(g_1^d)^2 S_p^{d,i} \tag{3}$$

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

where, $EC_p^{d,i}$ represents the EC calculation per one task, η represents the switched capacitance, which determine on the chip construction.

3.1.4 Edge Computing

The primary distinction between edge and local computing lies in the extra costs associated with offloading computations from a MD. These additional costs encompass the offloading delay and the EC required for transmitting information to the MEC server. The delay and EC for the offloading are determined as D_p^{off} and EC_p^{off} .After MD p completes the transmission, the MEC server accomplishes the computation task $u\eta$ effectively. Accordingly, the delay for executing task $u\eta$ is D_p^g.Therefore, the entire executing duration of the task $u\eta$ particularly encompasses dual phases: the primary phase is the duration of communicating the task Txi_n from UE to the MEC server and the next phase is the computing performance duration on the MEC server. The delay for uploading data of the task $u\eta$ is accompanied with uplink transmission rate and input data size of UE i straight achieved by Eqn(4).

$$D_p^g = \frac{M_p^d}{R_p^d} \tag{4}$$

where, M_p^d represents the size of the $u\eta$ executing task, R_p^d represents the uplink transmission rate of UE. Then, the equivalent EC for transmitting the task Txi_n to the MEC server is measured.

3.1.5 Server Computation Model

Once, vH_g selects for offloading, the transmitting delay of task D_p^i from vH_g to MEC_g is achieved by Eqn(5),

$$T_{r,i,trs}^g = \frac{t_k}{R_{i,g,k}} \tag{5}$$

The computational duration of task U_g performed at MEC_g is determined through Eqn (6).

$$T_{r,i,exe}^g = \frac{\phi_k}{E_g} \tag{6}$$

where, $T_{r,i,trs}^g$ represents the required time transmitting from UE to the MEC server, $T_{r,i,exe}^g$ represents the required executing time from UE to the MEC server, t_k is the data size of U_g , $R_{i,g,k}$ is the transmission rate from vH_g to MEC_g and E_g is the computing power of MEC_g .

3.1.6 Task offloading & execution model

During transmission, it aims to strike a balance between offloading as many tasks as possible to edge servers and preserving local execution for tasks that require it. According to the Shannon formula, the transmission frequency at time duration d is expressed through Eqn (7),

$$Tr^d = BW \log_2(1+\frac{A_{Tx}^d g^d}{P_n}) \tag{7}$$

where, BW represents the bandwidth, P_n represents the noise at the receiver, A_{Tx}^d represents the transmitting capacity of the MD, which is inhibited by $h_{\max}$, Subsequently the channel gain g^d among the MD and the MEC server at the duration d is defined through Eqn(8).

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

$$g^{d} = G_0\left(\frac{t_0}{t^{d}}\right)^{\vartheta} \tag{8}$$

where, t^{d} is the space between the MD and the MEC server, t_0 is the antenna distance test distance of the BS to which the MEC server is associated, G_0 represents the channel gain variable, ϑ represents the path-loss variable. The transmission latency of the task $u\eta$ is associated to the data dimension and uplink rate of respective time slot. Thus, the transmission delay is achieved by Eqn (9)

$$T_{dx}^{g} = \frac{u^{i}}{s^{d}} \tag{9}$$

where, T_{dx}^{g} represents the transmission delay through offloading, u^{i} represents the task delay of the user i , s^{d} represents the cost of uplink transmission. The EC is achieved by Eqn (10).

$$EC_{Tx}^{i} = H_{Tx}^{d} T_{Tx}^{i} \tag{10}$$

where, EC_{Tx}^{i} represents the transmission EC of task i , H_{Tx}^{d} represents the transmit power of MD, T_{Tx}^{i} represents the transmission delay of task i . During the execution point, the task is carried out on the MEC server. Where the CPU rate of the MEC server $Freq$ remains constant. Hence the execution delay is achieved by Eqn (11).

$$T_{r}^{i} = \frac{Z^{i}}{Freq} \tag{11}$$

where, T_r^i represents the delay of the task execution phase, Z^{i} represents the execution delay of the transmission queue, $Freq$ represents the CPU frequency of the MEC server.

3.1.7 Energy Consumption Model

For individual time slot d, UE is assumed to formulate any assessment to perform the task i, so the EC of the task execution period is defined through Eqn (12).

$$EC_{p}^{d} = \frac{q_i M_{p}^{d}}{R_{p}^{d}} \tag{12}$$

where, q_i represents the transmission power, EC_{p}^{d} represents EC of offload computing system, When a larger amount of computation resources is assigned to a particular UE, the task performance time decreases, but this results in an increased EC. Accordingly, a total delay and EC for UE *i* to offload the task $u\eta$ are simply expressed by Eqns (13) and (14).

$$D_{p}^{off} = D_{p,Tx}^{d} + D_{p,exe}^{d} = \frac{M_{p}^{d}}{R_{p}^{d}} + \frac{d}{\chi_{p}^{d} MEC} \tag{13}$$

$$EC_{p}^{off} = EC_{p,Tx}^{d} + EC_{p,exe}^{d} + EC_{p}^{d} MEC = \frac{q_i M_{p}^{d}}{R_{p}^{d}} + \frac{h_i^{u} d}{\chi_{p}^{d} MEC} \tag{14}$$

where, D_{p}^{off} and EC_{p}^{off} represents offloading delay and EC respectively, $D_{p,Tx}^{d}$ and $EC_{p,Tx}^{d}$ represents transmitting time and EC of uploading task respectively, $D_{p,exe}^{d}$ and $EC_{p,exe}^{d}$ represents execution time of delay and EC of the particular task, χ_{p}^{d} represents the percentage of computation resources allocated to accomplish the task, h_i^{u} represents the UE

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

standby power state. Thus, the downlink transmission of an offloading task in the model size is typically much less than the input data.

3.2 Problem formulation

In this research, proposing AOAHE algorithm for computation offloading which aims to optimize mobile devices' resource utilization and energy efficiency. The challenge involves addressing the issue of offloading and resource utilization in the proposed MEC mechanism. The primary aim is to reduce the overall EC of the complete MEC system while adhering to definite task delay restraints. To achieve this, the relevant optimization problem is formulated by Eqns (15) and (16).

$$r = \sum_{d=0}^{\eta-1} EC_p^d + D_p^{off} + \chi_p^d \tag{15}$$

$$\min_{y(d),h(d),g(d)} \lim_{\eta\to\infty} \frac{1}{\eta} \sum_{d=0}^{\eta-1} EC_p^d + D_p^{off} + \chi_p^d .$$

$$G1: EC_p^d [y_i^d \in \{0,1\} \ \nabla i \in Q]$$

$$G2: D_p^{off} [h_i^d \le g_{i,\max}^d \ \nabla i \in Q]$$

$$G3: \chi_p^d [g_i^d \in 0 \le \gamma_i^d \le 1 \ \nabla i \in Q] \tag{16}$$

where, $\lim_{\eta\to\infty} \frac{1}{\eta} \sum_{d=0}^{\eta-1} EC_p^d + D_p^{off} + \chi_p^d$ is the time average of the complete MEC model. ∇i represents the decision variable, Q represents the total task set, $G1$ signifies the EC of the edge execution model, $G2$ signifies the execution delay of the task, and $G3$ signifies the allocation of computation resources allocated to UE. To solve the problem stated in Eqn (16), it's essential to determine the finest solutions for the offloading decision vector $y(d) = \{y_i^d \,|\, i \in Q\}$, computation resource allocation vector $g_i^d = \{\gamma_i^d \,|\, i \in Q\}$ and specified delay $h_i^d = \{g_{i,\max}^d \in Q\}$ in each time slot. These factors work together cohesively to reduce the overall computation, and EC within the system while adhering to the specified delay restrictions. Particularly, the offloading assessment variables y_i^d are binary variables, whereas the resource allocation variable γ_i^d and $g_{i,\max}^d$ are vigorously varying. Consequently, the scheme necessitates a significant quantity of network state report to construct the comprehensive decisions concerning offloading and resource allocation built on the current network position. Since some existing methods is not able to adjust to the dynamic properties and is not suitable for intelligent decisions, thus proposing AOAHE algorithm to address the considered problem in this research.

3.3 Computation offloading and resource utilization optimization problem based on adaptive offloading cat hunt Optimization Algorithm

This research proposes an AOCHO algorithm for heterogeneous environments to reduce time, EC, and resource utilization problems and make a reasonable offloading decision [28]. Each individual or problem solution goes through two primary phases: the search or attack phase, and the determination of their current state. During the rest and alert phase, these individuals explore their surroundings to decide on their next course of action. They contemplate various situations, attempting to move toward the most optimal location. In this behavioral model, the parameters define each individual's memory size, dimension range, and vector dimension during the search process. The flow of the AOCHO algorithm is given below,

- **Evaluating fitness function**

Using the fitness values of the copied vectors, calculate

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

the probability for each copy vector, as shown in Eqn (16). This calculation aims to reduce delay, EC, and resource utilization. Subsequently, select the best copy vector based on these probabilities and replace it with the candidate cat vector.

$$fitness = \begin{cases} 1/r & if\ TP_{\max} = TP_{\min} \\ \dfrac{|TP_i - TP_{\min}|}{TP_{\max} - TP_{\min}} & otherwise \end{cases} \quad (16)$$

where, r represents the delay, EC, and resource utilization function derived from Eqn(15), TP_i signifies the rate of each solution vector, $TP_{\max}$ and $TP_{\min}$ represents the maximum and minimum range of the copy vectors, correspondingly. When the minimum and maximum fitness values are identical, it implies that all the copy vectors have the same fitness values, resulting in equal probabilities for all copy vectors.

- **Tracking towards prey**

In this research, each problem solution updates its speed based on the state of the problem area, as described by Eqn (17),

$$u_d(d+1) = u_j(d) + r \times P \times (z_{best} - z_d(d)) \quad (17)$$

where, z_{best} represents the ideal spot where the current solution in the problem range, $z_d(d)$ represents the present location of the solution vector. $u_d(d+1)$ and $u_d(d)$ represents the new and current velocity of a problem solution to reach the optimal solution respectively, r and P represents the random and learning coefficients of the problem respectively.

- **Mutual learning**

In the proposed AOCHO algorithm for each problem solution are expected to learn from each other. For example, if a solution Z_d happens to encounter a solution Z_j, and Z_j is deemed to be more deserving than Z_d, then Z_d adjusts its position in the direction of a solution Z_j. This modeling is mathematically represented through Eqn (18).

$$Z_d^{new} = Z_d + \sin(\frac{\pi}{2} \times \frac{1}{iteration_{\max}}) * r \times (Z_j - Z_d) \quad (18)$$

where, $\sin(\frac{\pi}{2} \times \frac{1}{iteration_{\max}})$ is a component that amplifies the influence of the factor $(Z_j - Z_d)$ as the algorithm iterates more. As the algorithm progresses through its repetitions, there is a stronger inclination for a solution Z_d to transition towards solution Z_j, signifying a shift in the search behaviour from global to local.

- **Utilizing balance factor with adaptive values**

Consuming adaptive values of $r_{1-adaptive}$ and $r_{2-adaptive}$ rather than fixed values to attain higher performance. The fixed values of $r_1 = 0.18, r_2 = 0.82$,while in the AOCHO adjusting the values of both $r_{1-adaptive}$ and $r_{2-adaptive}$ by decreasing $r_{1-adaptive}$ and increasing $r_{2-adaptive}$ to easily achieves the optimal solution. Initially, with a high $r_{1-adaptive}$ and low $r_{2-adaptive}$, the cats have the freedom to move around the search space without being attracted to the best solution found by the population. Equally, by consuming a smaller value of $r_{1-adaptive}$

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

and a larger value of $r_{2-adaptive}$ later on, the cats are directed to the global ideal solution. This move in the values of $r_{1-adaptive}$ and $r_{2-adaptive}$ helps to raid a balance between exploration and exploitation, leading to improved outcomes. This variation in the values of $r_{1-adaptive}$ and $r_{2-adaptive}$ is precisely signified by Eqn (18).

$$p_{iteration+1} = \begin{cases} D_{iteration} \times r_{1-adaptive} & if\ p > 0.18 \\ D_{iteration} \times r_{2-adaptive} & f\ p \le 0.18 \end{cases} \quad (18)$$

where, p represents the updated position of exploration and exploitation, $D_{iteration}$ represents the distance between exploration to exploitation. The maximum iteration taken to minimize the problem is to 1000. These modifications significantly improved both the optimal value and the convergence speed. AOCHO is equipped to handle multiple objectives simultaneously. It addresses the interconnected challenges of delay, EC, and cost, ensuring a balanced approach that considers the interplay of these factors. This holistic optimization benefits the overall efficiency and performance of the offloading process. Also, it leverages adaptive values and balance factors to enhance its decision-making process. This adaptability is crucial in dynamic mobile environments to contribute to extending device battery life and reducing overall EC. The flowchart of AOCHO is presented in Fig 3.

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

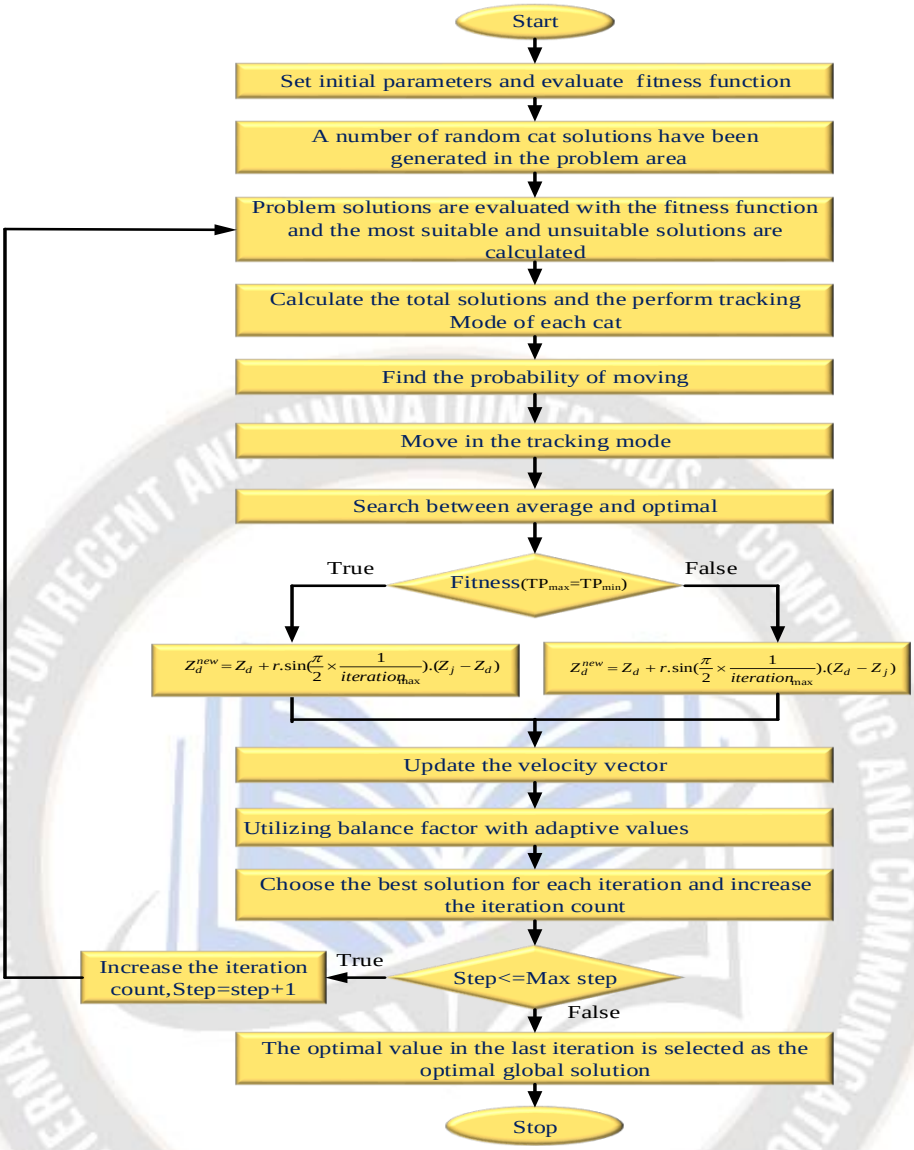


Fig 3 Flowchart of AOCHO algorithm

4. Results and discussions

This section consists of experiment results constructed on the system model and the proposed AOCHO algorithm. An adaptive computation offloading system is simulated under a heterogenous environment and the experimental results are validated and compared by using some existing state-of-art techniques.

4.1 Simulation setup

To assess the proposed AOCHO model efficiency is performed by Matlab with Intel® core™ i5-9300H system. It consists of a 2.4 GHz CPU and 8GB RAM. Moreover, the parameter settings of this experiment are provided in Table 3.

Table 3 Description of network parameters

Simulation parameters	Value
Number of tasks	61,223
Computation size of edge servers	10 GHz
Channel bandwidth	180 KHz
Computation size of local devices	0.5-1 GHz
Data size	300~1000 kb

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

Computation size of MEC server	10 GHz
UE's transmission power	3 W
Distance among MEC server and UE	80~100 m
Density of computation tasks per UE	500~800 cycle/bit
Number of MECs	3-7
Number of MDs	6

4.2 Comparison assessment

To assess the efficiency of the proposed AOCHO method, additionally six baseline algorithms are introduced as follows,

- Local Execution Task (LET): The MD perform their respective tasks locally whenever possible within the higher acceptable delay.
- Server Execution Task (SET): The MD transmit all computation tasks to the MEC server using high transmission power.
- Offloading Scheduling (OS): The offloading mechanisms are optimized to diminish the EC of the whole MEC system, without taking resource allocation optimization into account. The MEC server's resources are evenly allocated among each offloaded UE.
- Random Task (RT): The MDs randomly distribute energy for transmitting information to the server, while the MEC server arbitrarily assigns computing resources to the offloaded tasks.
- DECO [22]: This algorithm primarily focuses on computation offloading decisions in various scenarios. It takes into consideration task priorities and the heterogeneity of ECSs during the task-node mapping progression.
- NSGAIII-TOMEC [23]: This algorithm is based on a multi-objective evolutionary approach and involves multiple MEC servers, MDs, and tasks.

4.3 Performance analysis

Figs 4(a-c) illustrate the performance of delay, EC and cost for each MD following task offloading. Lower index values signify superior performance. The comparison in Figs 4(a-c) reveals that, in contrast to NSGAIII-TOMEC, the AOCHO algorithm outperforms in delay, EC and cost for each device.

Tasks	LET [46] (s)	OS [46] (s)	DECO [46] (s)	SET [46] (s)	RT [46] (s)	NSGAIII-TOMEC [47] (s)	AOAHE (proposed) (s)
1	0.0325	0.032	0.005	0.012	0.023	0.007	0.004
2	0.043	0.007	0.018	0.014	0.021	0.012	0.005
3	0.06	0.016	0.014	0.015	0.012	0.013	0.006
4	0.082	0.023	0.025	0.016	0.015	0.015	0.008
5	0.112	0.027	0.015	0.017	0.016	0.014	0.007
6	0.152	0.051	0.014	0.018	0.025	0.012	0.006

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

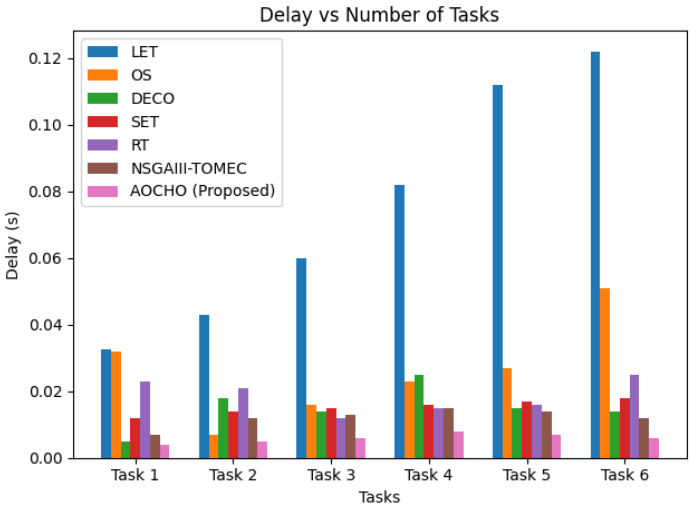


Fig. 4(a)

Tasks	LET [46] (s)	OS [46] (s)	DECO [46] (s)	SET [46] (s)	RT [46] (s)	NSGAIII-TOMEC [47] (s)	AOCHO (proposed) (s)
1	1.77	0.13	0.05	1.23	0.05	0.08	0.05
2	1.45	0.19	0.16	0.46	0.05	0.07	0.03
3	0.9	0.38	0.15	0.69	0.08	0.11	0.09
4	0.56	0.58	0.18	0.96	0.06	0.14	0.05
5	0.32	0.74	0.32	1.32	0.09	0.14	0.09
6	0.12	0.92	0.28	1.56	0.1	0.21	0.1

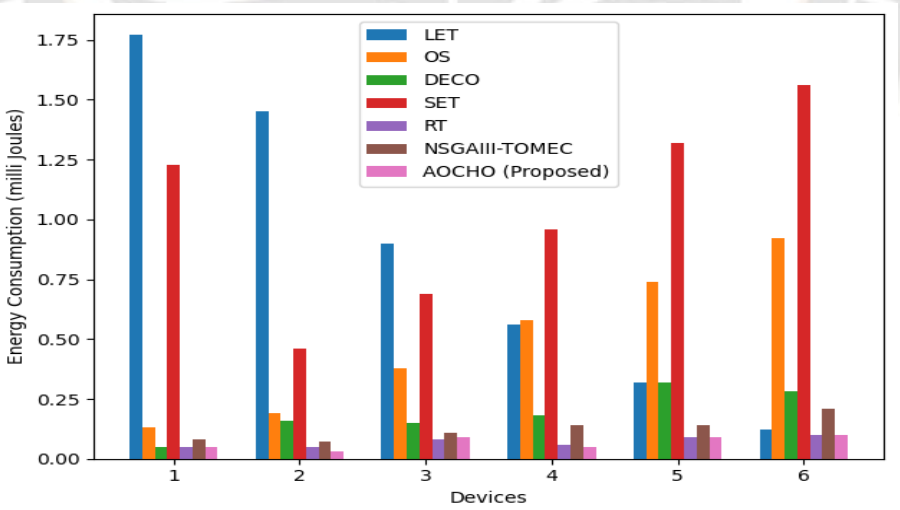


Fig. 4(b)

Devices	LET [46]	OS [46]	DECO [46]	SET [46]	RT [46]		

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

	(s)	(s)	(s)	(s)	(s)	NSGAIII-TOMEC [47] (s)	AOCHO (proposed) (s)
1	0.98	0.48	0.31	0.35	0.37	0.26	0.22
2	1.07	0.43	0.37	0.75	0.49	0.25	0.19
3	0.92	0.58	0.33	0.48	0.45	0.19	0.14
4	0.98	0.36	0.31	0.39	0.24	0.23	0.18
5	0.96	0.83	0.35	0.59	0.47	0.3	0.22
6	0.91	0.66	0.33	0.61	0.42	0.18	0.13

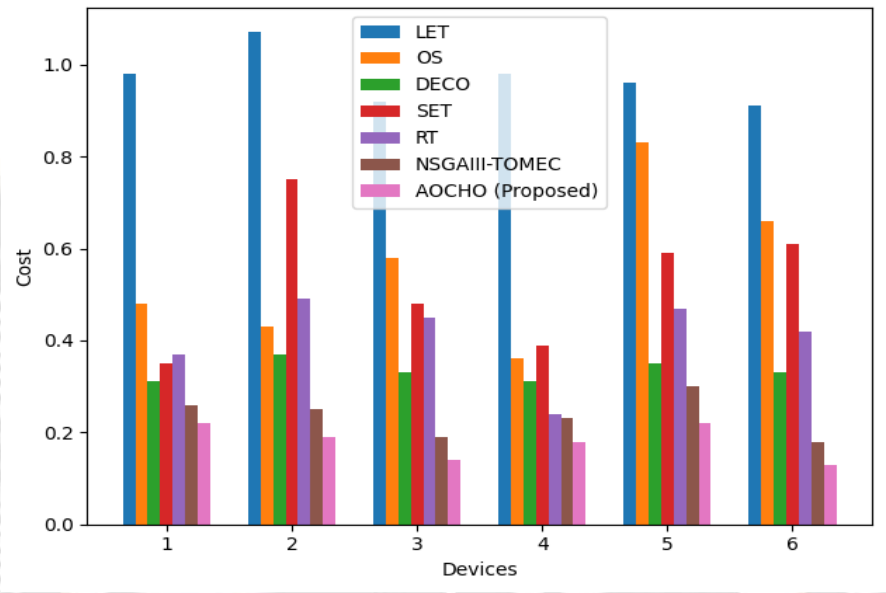


Fig. 4(c)

Fig 4 Effects on mobile devices in related to (a)Delay (b) Energy consumption (c) Cost

Additionally, the delay and cost for each device show superiority over other methods. However, while DECO delivers commendable outcomes in EC and enhances the overall system benefits compared to various benchmark algorithms, but it lags significantly behind AOCHO concerning delay, cost and overall offloading benefits. The test validates the efficacy of the AOCHO algorithm and underscore that the offloading policy derived from the AOCHO algorithm demonstrates a more robust balance.

Fig 5(a) demonstrates the influence of task generation rates for the MD, ranging from 5 to 10 per seconds. When the quantity of tasks multiplies, the execution cost of these tasks progressively rises. In comparison to SET, the proposed AOCHO cuts the execution cost in half. Fig 5(b) illustrates the effect of the computation power of the MEC server on the execution cost. A higher CPU rate of the server results in a reduced execution cost for the tasks. Specifically, when the frequency is 10 GHz, the execution costs of the AOCHO system decrease by 18%, 35%, 38%, 43%, 49% and 87% compared to the other six baseline models.

Tasks	LET [46] (s)	OS [46] (s)	DECO [46] (s)	SET [46] (s)	RT [46] (s)	NSGAIII-TOMEC [47] (s)	AOCHO (proposed) (s)
5	11.46	1.45	1.28	2	3.74	3.22	2.15
6	12.41	2.2	1.73	3.01	3.57	2.62	1.95

7	13.11	2.54	1.96	3.35	2.71	2.16	1.55
8	13.65	3.49	2.42	3.67	2.89	1.96	1.15
9	13.99	3.81	3.05	3.95	2.2	1.33	0.86
10	14.2	4	3.57	4.09	1.39	0.93	0.35

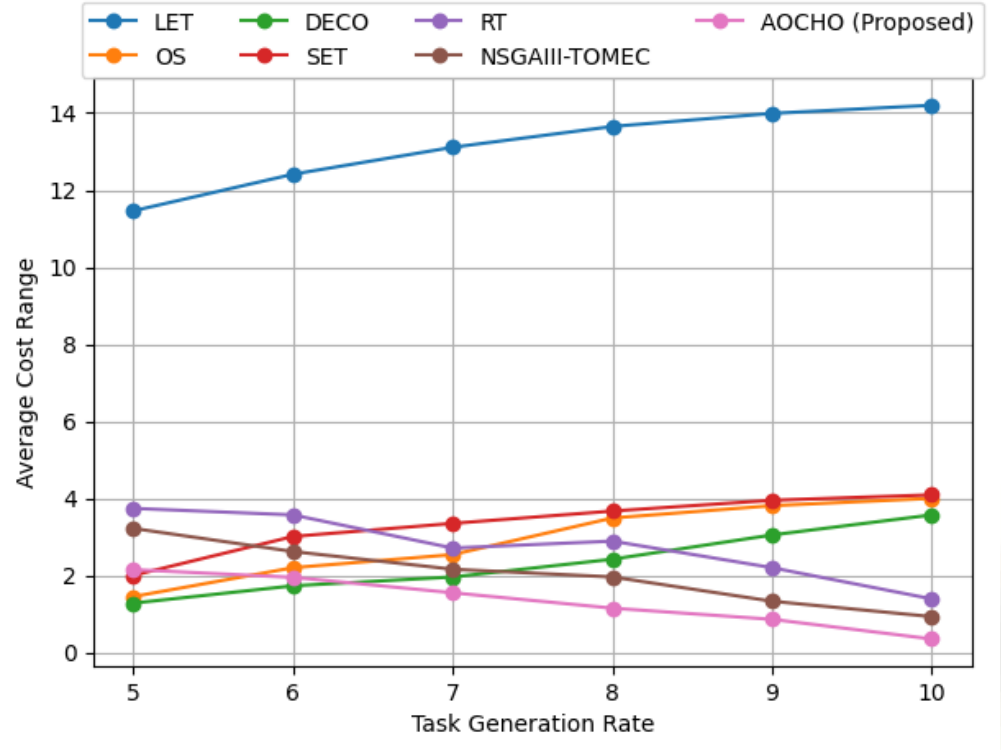


Fig. 5(a)

Tasks	LET [46] (s)	OS [46] (s)	DECO [46] (s)	SET [46] (s)	RT [46] (s)	NSGAIII-TOMEC [47] (s)	AOCHE (proposed) (s)
2	9.77	3.22	2.79	3.57	3.04	2.79	2.52
4	9.73	3.62	3.33	4.03	3.43	2.95	2.64
6	9.77	3.84	3.53	4.26	3.82	3.47	2.91
8	9.83	4.13	3.57	4.77	4.05	3.47	3.16
10	9.75	4.22	3.78	4.92	4.24	3.55	3.45

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

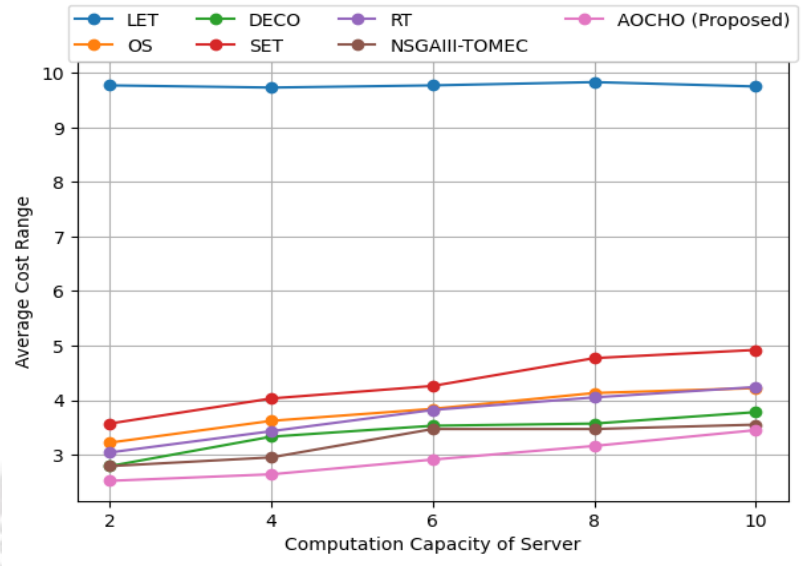


Fig. 5(b)

Mobile Devices	LET [46] (s)	OS [46] (s)	DECO [46] (s)	SET [46] (s)	RT [46] (s)	NSGAIII-TOMEC [47] (s)	AOCHO (proposed) (s)
20	11.5	3.66	1.05	4.6	2.97	2.78	0.53
25	11.45	2.99	1.25	4.14	2.66	2.28	0.89
30	11.52	2.66	1.51	3.66	2.2	1.92	0.96
35	11.52	2.06	1.96	3.14	1.6	1.63	1.22
40	11.57	1.8	2.01	2.8	1.41	1.25	1.44
45	11.45	1.7	2.37	2.4	1.22	0.96	1.63
50	11.43	1.22	2.68	1.92	1.92	0.84	1.92

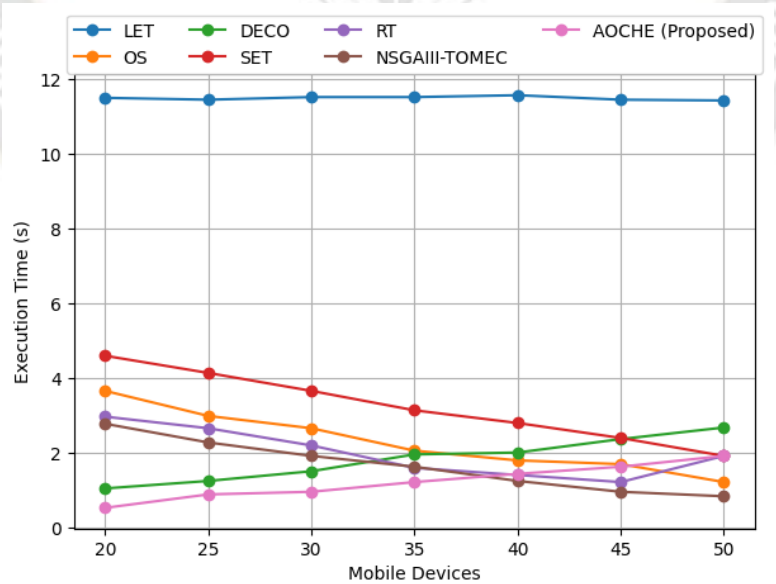


Fig. 5(c)

Fig 5 Task execution performance with (a) Task generation rate (b)Computation capacity of server (c) Different quantity of MD

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

Fig 5(c) illustrates the effect of varying the quantity of MDs on the execution cost. An increased number of MDs leads to higher execution delays for the tasks due to the increased task load. With 20 MDs, the AOCHO algorithm reduces the execution cost by 20%, 28%, 32%, 36%, 54%, and 92% compared to the other six baseline models.

As depicted in Fig 6, the typical delay for all approaches increases with the rising demand for computational resources by tasks. Significantly, the OS method consistently shows higher average delays in comparison to the other six methods, primarily due to the restricted resources of the MEC server. In contrast, the AOCHO method excels, exhibiting the smallest average delay among the baseline algorithms.

Computation Resources required by Tasks	LET [46] (s)	OS [46] (s)	DECO [46] (s)	SET [46] (s)	RT [46] (s)	NSGAIII-TOMEC [47] (s)	AOCHO (proposed) (s)
1500	1.3	1.64	2.36	3.06	2.26	2.53	2.19
2000	1.69	1.96	2.24	2.85	2.13	2.38	2.08
2500	1.99	2.32	2.04	2.57	1.96	2.18	1.9
3000	2.34	2.68	1.79	2.22	1.74	1.9	1.69
3500	2.56	3.04	1.55	1.77	1.5	1.59	1.45
4000	2.76	3.35	1.23	1.33	1.17	1.27	1.11

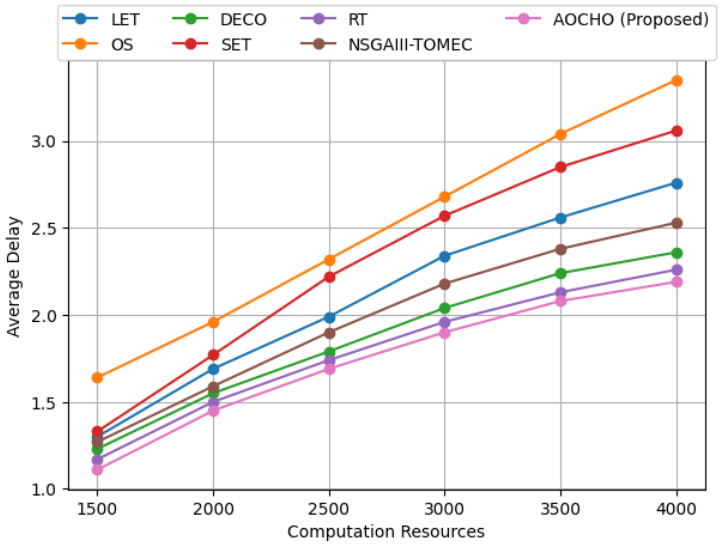
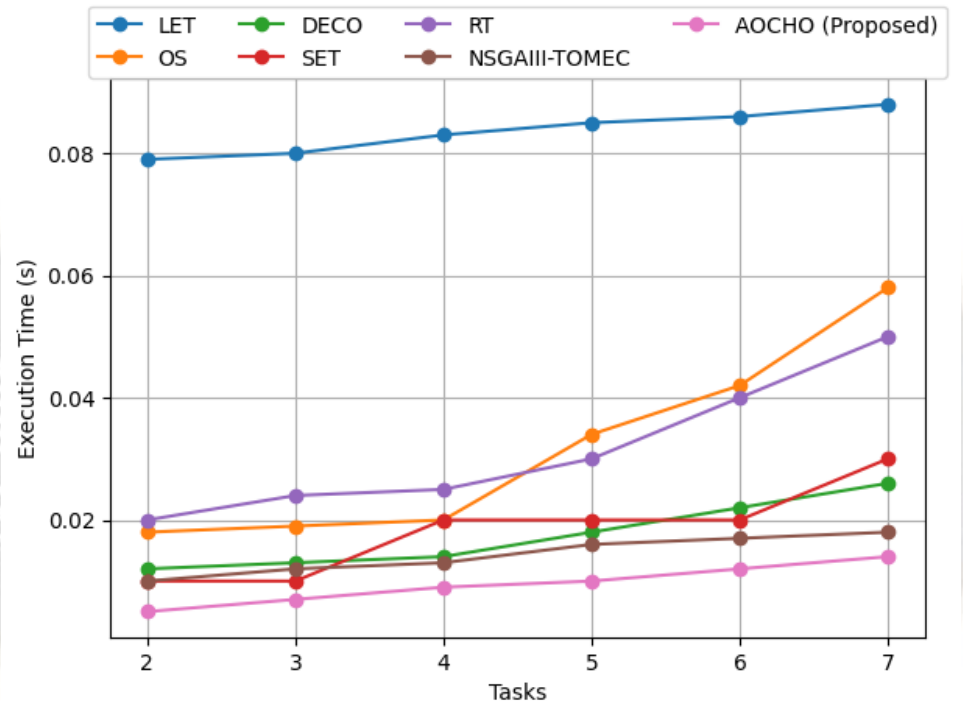


Fig 6 Performance of average delay with respect to computation resources

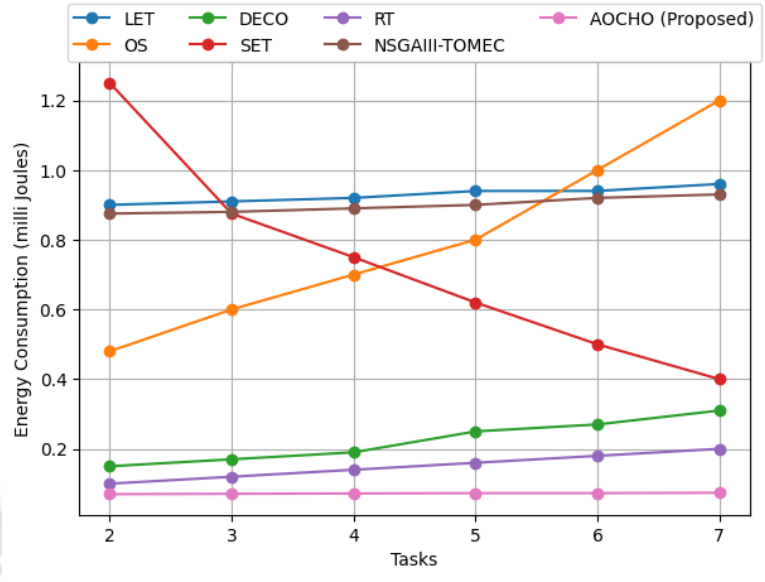
In Fig 7(a), it is noticeable that an increase in the number of tasks on MDs leads to a rise in the average delay experienced by the AOCHO algorithm. This phenomenon primarily occurs due to the constant total computing resources available in the MEC system. In Fig. 7(b), while dealing with a low number of tasks, the AOCHO algorithm primarily offloads most tasks to the MEC server, reducing delay and keeping MDs energy consumption low. However, as the quantity of tasks multiplies, the obtainable resources on the MEC server become inadequate, certainly resulting in increased delay. To manage this situation, certain tasks are allocated for local execution. As illustrated in Fig. 7(c), the average cost progressively diminishes as the amount of tasks on MDs increases, leading to changes in cost efficiency.

Tasks	LET [46] (s)	OS [46] (s)	DECO [46] (s)	SET [46] (s)	RT [46] (s)	NSGAIII-TOMEC [47] (s)	AOCHO (proposed) (s)
2	0.079	0.018	0.012	0.01	0.02	0.01	0.005
3	0.08	0.019	0.013	0.01	0.024	0.012	0.007
4	0.083	0.02	0.014	0.02	0.025	0.013	0.009
5	0.085	0.034	0.018	0.02	0.03	0.016	0.01
6	0.086	0.042	0.022	0.02	0.04	0.017	0.012
7	0.088	0.058	0.026	0.03	0.05	0.018	0.014



Tasks	LET [46] (s)	OS [46] (s)	DECO [46] (s)	SET [46] (s)	RT [46] (s)	NSGAIII-TOMEC [47] (s)	AOCHO (proposed) (s)
2	0.9	0.48	0.15	1.25	0.1	0.875	0.07
3	0.91	0.6	0.17	0.875	0.12	0.88	0.071
4	0.92	0.7	0.19	0.75	0.14	0.89	0.072
5	0.94	0.8	0.25	0.62	0.16	0.9	0.073
6	0.94	1	0.27	0.5	0.18	0.92	0.073
7	0.96	1.2	0.31	0.4	0.2	0.93	0.074

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



Tasks	LET [46] (s)	OS [46] (s)	DECO [46] (s)	SET [46] (s)	RT [46] (s)	NSGAIII-TOMEC [47] (s)	AOCHO (proposed) (s)
1	0.05	0.65	0.5	0.97	0.39	0.48	0.01
2	0.054	0.625	0.45	0.96	0.4	0.39	0.02
3	0.058	0.55	0.41	0.95	0.55	0.37	0.03
4	0.062	0.62	0.4	0.98	0.57	0.36	0.03
5	0.066	0.6	0.4	0.96	0.56	0.35	0.04
6	0.07	0.58	0.39	0.95	0.53	0.34	0.05

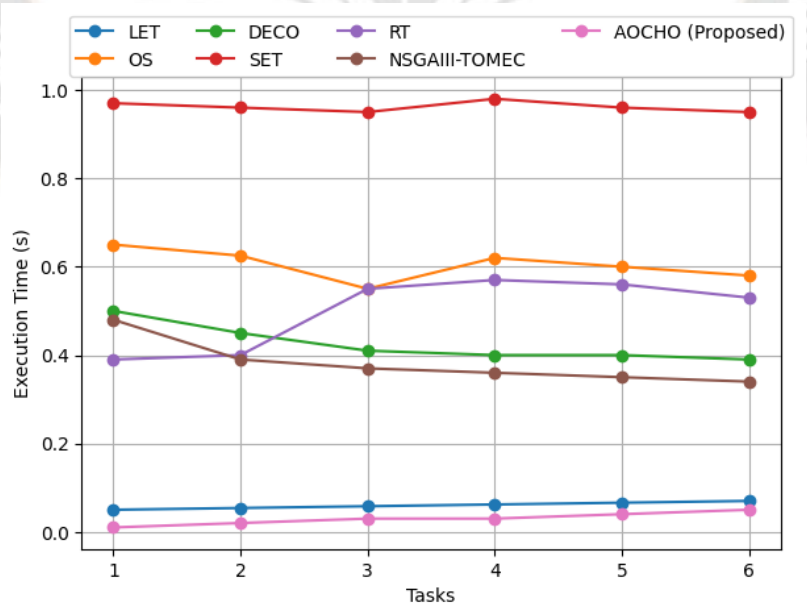
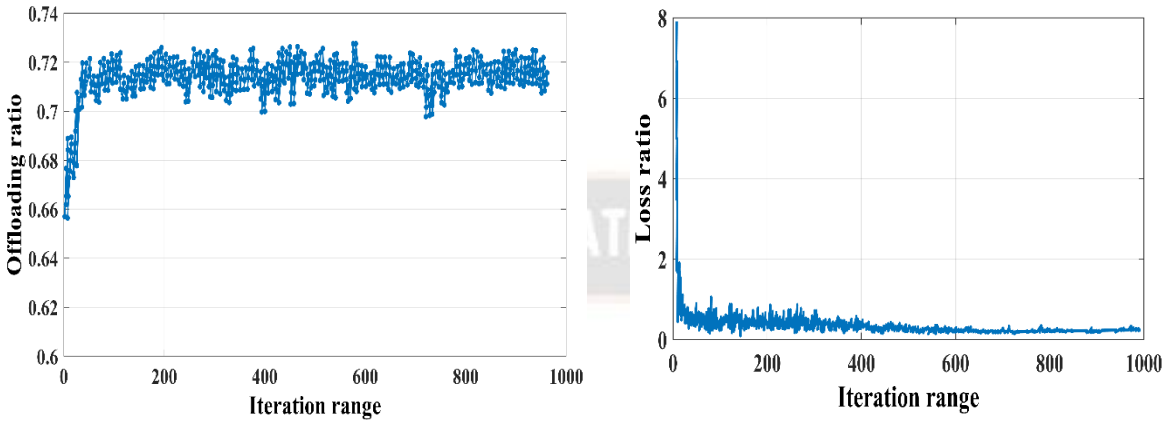


Fig.7(c)

Fig 7 Performance of MEC in terms of (a)Time (b) Energy consumption (c) Cost

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

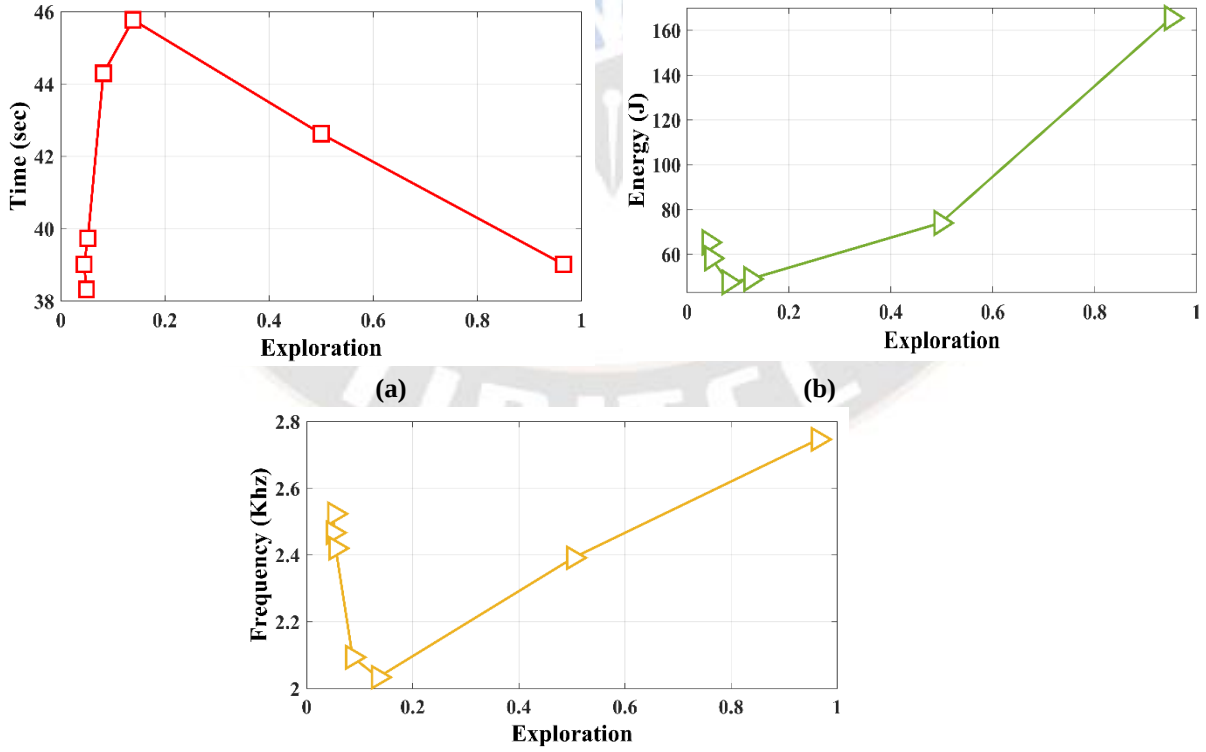
Fig 8(a) represents the evolution of the average offloading ratio in the AOCHO algorithm. This curve indicates the offloading ratio within the model stabilizes over time, reflecting its rapid convergence. Fig 8(b) illustrates the loss curve of the AOCHO model. The outcomes illustrate a gradual reduction in the final loss value, which eventually reaches a stable state.



(b)

Fig 8 Performance of AOCHO in (a) offloading ratio (b) Loss ratio

Figs 9(a-c) display the variations in delay, EC and computing frequency within the AOCHO algorithm, offering insights into its convergence under different exploration rates. The results indicate the exploration rate of the AOCHO algorithm is set to 0.05, it achieves the most favourable overall performance and the changes in computing frequency eventually stabilize.



(a) **(b)**

(c)

Fig 9 Exploration rates for AOCHO algorithm in terms of (a)Time (b) Energy consumption (c) Frequency

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

Fig10(a) displays the average EC of two algorithms. The AOCHO algorithm demonstrates a significant reduction in EC with a minimal relative error compared to the conventional CHO algorithm. Its performance remains superior, especially in terms of EC during task offloading

.

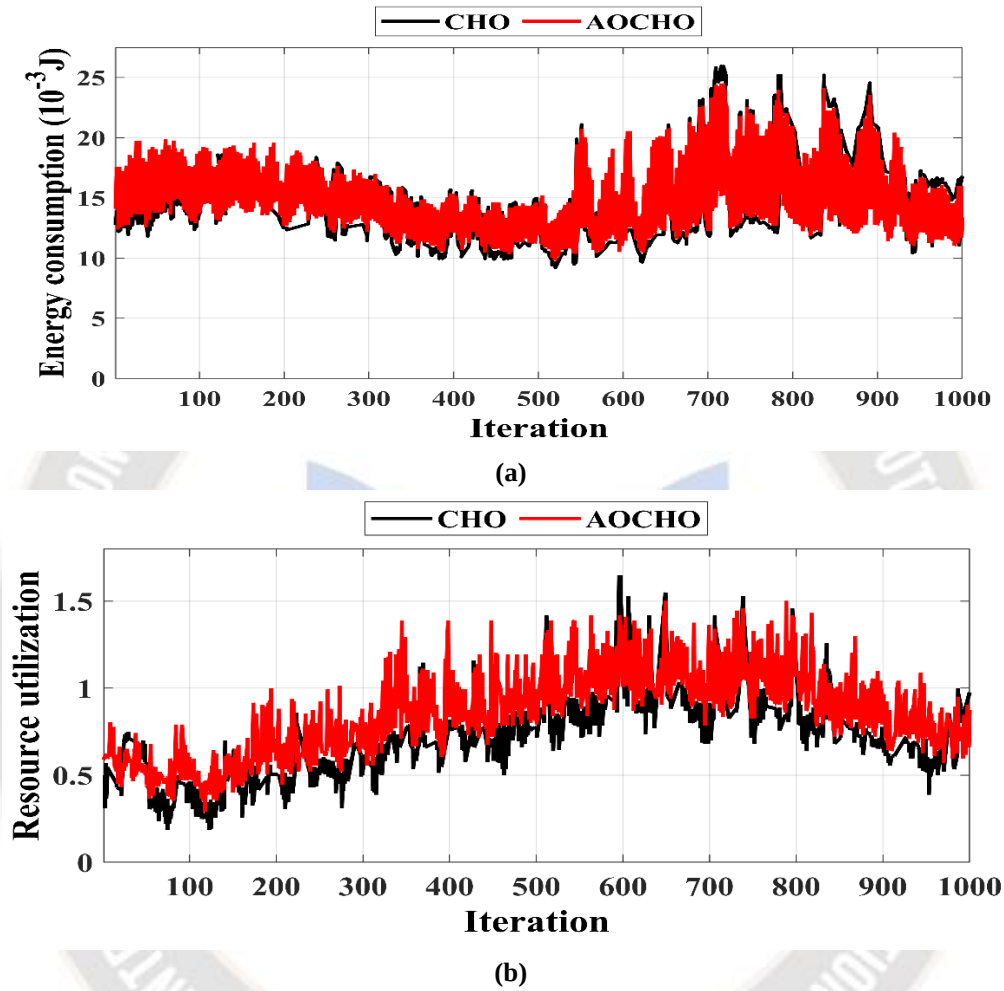


Fig 10 Performance of (a)energy consumption (b) resource utilization impact on CHO Vs proposed AOCHO

Fig 10(b) illustrates the resource utilization of two algorithms. The AOCHO algorithm achieves the highest resource utilization performance with a minimal relative error compared to the CHO algorithm.

The convergence level to local optima is a key metric for evaluating metaheuristic algorithms. In Fig 11, the average convergence amount, represented as a percentage, towards local optima is illustrated for the proposed AOCHO algorithm and various additional algorithms when applied to complex evaluation functions. The experiments reveal the AOCHO algorithm demonstrates a slower convergence rate towards local optima in comparison to the other algorithms. Specifically, AOCHO achieves a local optimal convergence rate of approximately 7.8%, whereas the other algorithms, such as CHO, Firefly Algorithm (FA), Crayfish Optimization Algorithm (COA), Grey Wolf Optimization (GWO), Grasshopper Optimization Algorithm (GOA), Differential Evolution (DE), Spotted Hyena Optimization (SHO) and Particle Swarm Optimization (PSO) have local optimal convergence rates of 8.3%,15%, 14.6%,13%,10.3%, 14.3%, 12.6% and 15.6% respectively [28].

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

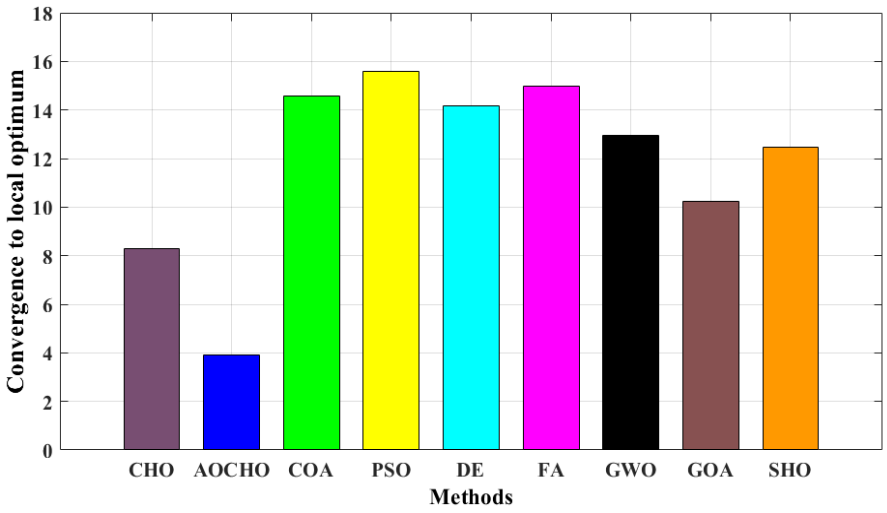


Fig 11 Comparison of AOCHO algorithm convergence with other algorithms

Among those, the PSO algorithm exhibits the highest convergence rate, while the proposed AOCHO algorithm shows the lowest convergence rate to local optima. Specifically, the AOCHO algorithm's convergence to local optima is approximately 1.5% lower than that of the CHO algorithm, making it one of the best-performing algorithms in this regard compared to existing algorithms.

The overall comparison of delay, EC and cost are compared with the existing techniques is presented in Table 4.

Table 4 Overall performance comparison of AOCHO method with existing techniques

Methods	***Delay (Sec)***	***EC (10^{-3}J)***	***Cost***
DRLCO [16]	*1.002*	*0.467*	*1.469*
DDQN [17]	*2.325*	*0.652*	-
MOIA [19]	*1.897*	*0.982*	-
MOWOA [21]	*2.271*	*0.295*	-
MOWOA2[21]	*2.265*	*0.342*	-
DECO (Type 1) [22]	*0.198*	*0.752*	-
DECO (Type 2) [22]	*3.992*	*0.482*	-
NSGAIII-TOMEC [23]	*0.0198*	*0.351*	*0.451*
AOCHO(Proposed)	*0.0172*	*0.251*	*0.387*

From Table 4, the proposed AOCHO method attains the lowest delay of 0.0172 Sec when compared with other methods. Thus, minimizing delay is pivotal in enhancing mobile computation offloading, as it directly impacts user experience, QoS, resource utilization, energy efficiency, and the overall efficiency of mobile networks and applications. Mobile computing systems prioritize low-latency offloading strategies to meet the demands of real-time and resource-efficient mobile applications. When comes to the EC, the proposed AOCHO method consumes lowest energy of 0.251J, which shows the proposed model is very efficient for computation offloading mechanism. Optimizing EC performance in mobile computation offloading is crucial for improving device battery life, ensuring sustainability, accommodating resource-constrained devices, enabling real-time applications, and enhancing the efficiency of networks and remote resources. When comes to the cost performance, the proposed AOCHO method utilizes the lowest cost of 0.387, which shows that efficient offloading decisions allocate computational tasks to the most cost-effective resources. Thus, the proposed AOCHO method effectively reduces the EC and attains favorable average delay and resource utilization for all computation tasks under various scenarios. This demonstrates the considerable stability and superiority of the AOCHO method when compared with other existing approaches in the computation

offloading system.

5. Conclusion

In this research, the AOCHO algorithm has been proposed to address multi-objective optimization by simultaneously focusing on critical objectives such as EC, delay reduction and cost optimization. These factors are of utmost importance in assessing the QoE for mobile users. initially, this research formulates the offloading problem related to DAGs to describe the intricate connection among offloading and resource allocation strategies in heterogeneous environments, considering constraints on delay and the need for EC minimization. Then, the proposed AOCHO method leverages a balance factor with adaptive values to enhance the global search capabilities of partial populations, leading to improved results. In comparative analyses with other baseline schemes and the simulation outcomes demonstrate the effectiveness of the AOCHO algorithm in significantly reducing task delays, optimizing resource utilization and minimizing the EC of MDs. In future, this research will extend to explore more complex MEC systems, wherein offloaded tasks can be segmented into different partitions. Additionally, need to leverage this algorithm to enhance decision-making processes, optimize offloading strategies and adapt to dynamic network conditions to further enhancing system performance.

References

1. Ren T, Niu J, Dai B, Liu X, Hu Z, Xu M, Guizani M. Enabling efficient scheduling in large-scale UAV-assisted mobile-edge computing via hierarchical reinforcement learning. IEEE Internet of Things Journal. 2021 Apr 7;9(10):7095-109.
2. Hilal AM, Alohali MA, Al-Wesabi FN, Nemri N, Alyamani HJ, Gupta D. Enhancing quality of experience in mobile edge computing using deep learning based data offloading and cyberattack detection technique. Cluster Computing. 2021:1-2.
3. Xiao H, Xu C, Ma Y, Yang S, Zhong L, Muntean GM. Edge intelligence: A computational task offloading scheme for dependent IoT application. IEEE Transactions on Wireless Communications. 2022 Mar 11;21(9):7222-37.
4. Guo M, Huang X, Wang W, Liang B, Yang Y, Zhang L, Chen L. Hagp: A heuristic algorithm based on greedy policy for task offloading with reliability of mds in mec of the industrial internet. Sensors. 2021 May 18;21(10):3513.
5. Ali A, Iqbal MM, Jamil H, Qayyum F, Jabbar S, Cheikhrouhou O, Baz M, Jamil F. An efficient dynamic-decision based task scheduler for task offloading optimization and energy management in mobile cloud computing. Sensors. 2021 Jul 1;21(13):4527.
6. Hussain W, Merigo JM, Gao H, Alkalbani AM, Rabhi FA. Integrated AHP-IOWA, POWA framework for ideal cloud provider selection and optimum resource management. IEEE Transactions on Services Computing. 2021 Nov 4.
7. Wang R, Zang C, He P, Cui Y, Wu D. Auction-based profit maximization offloading in mobile edge computing. Digital Communications and Networks. 2023 Apr 1;9(2):545-56.
8. Zhou H, Wu T, Chen X, He S, Guo D, Wu J. Reverse auction-based computation offloading and resource allocation in mobile cloud-edge computing. IEEE Transactions on Mobile Computing. 2022 Jul 18.
9. Keshavarznejad M, Rezvani MH, Adabi S. Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms. Cluster Computing. 2021 Sep:1-29.
10. Shen H, Jiang Y, Deng F, Shan Y. Task Unloading Strategy of Multi UAV for Transmission Line Inspection Based on Deep Reinforcement Learning. Electronics. 2022 Jul 12;11(14):2188.
11. Liu T, Guo D, Xu Q, Gao H, Zhu Y, Yang Y. Joint Task Offloading and Dispatching for MEC With Rational Mobile Devices and Edge Nodes. IEEE Transactions on Cloud Computing. 2023 May 26.
12. Gopi R, Suganthi ST, Rajadevi R, Johnpaul P, Bacanin N, Kannimuthu S. An enhanced green cloud based queue management (GCQM) system to optimize energy consumption in mobile edge computing. Wireless Personal Communications. 2021 Apr;117:3397-419.
13. Liao L, Lai Y, Yang F, Zeng W. Online computation offloading with double reinforcement learning algorithm in mobile edge computing. Journal of Parallel and Distributed Computing. 2023 Jan 1;171:28-39.
14. Zhou H, Jiang K, Liu X, Li X, Leung VC. Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing. IEEE Internet of Things Journal. 2021 Jun 22;9(2):1517-30.
15. Zalat MS, Darwish SM, Madbouly MM. An Adaptive Offloading Mechanism for Mobile Cloud

Computing: A Niching Genetic Algorithm Perspective. IEEE Access. 2022 Jul 19;10:76752-65.

16. Zhu SF, Cai JH, Sun EL. Mobile edge computing offloading scheme based on improved multi-objective immune cloning algorithm. Wireless Networks. 2023 May;29(4):1737-50.
17. Zhang X, Wu W, Liu S, Wang J. An efficient computation offloading and resource allocation algorithm in RIS empowered MEC. Computer Communications. 2023 Jan 1;197:113-23.
18. Huang M, Zhai Q, Chen Y, Feng S, Shu F. Multi-objective whale optimization algorithm for computation offloading optimization in mobile edge computing. Sensors. 2021 Apr 8;21(8):2628.
19. Azizi S, Othman M, Khamfroush H. DECO: A Deadline-Aware and Energy-Efficient Algorithm for Task Offloading in Mobile Edge Computing. IEEE Systems Journal. 2022 Jul 6;17(1):952-63.
20. Chu X, Leng Z. Multiuser computing offload algorithm based on mobile edge computing in the internet of things environment. Wireless Communications and Mobile Computing. 2022 Mar 3;2022:1-9.
21. Zhu SF, Sun EL, Zhang QH, Cai JH. Computing Offloading Decision Based on Multi-objective Immune Algorithm in Mobile Edge Computing Scenario. Wireless Personal Communications. 2023 May;130(2):1025-43.
22. Sun F, Zhang Z, Chang X, Zhu K. Towards Heterogeneous Environment: Lyapunov-orientated ImpHetero Reinforcement Learning for Task Offloading. IEEE Transactions on Network and Service Management. 2023 Apr 13.
23. Fang J, Shi J, Lu S, Zhang M, Ye Z. An efficient computation offloading strategy with mobile edge computing for IoT. Micromachines. 2021 Feb 17;12(2):204.
24. Ghaedi A, Bardsiri AK, Shahbazzadeh MJ. Cat hunting optimization algorithm: a novel optimization algorithm. Evolutionary Intelligence. 2023 Apr;16(2):417-38.