Journal of Grid Computing (2024) 22:69
https://doi.org/10.1007/s10723-024-09786-y

RESEARCH

A Novel Levy Walk-based Framework for Scheduling Power-intensive Mobile Edge Computing Tasks

Abolfazl Younesi · Mohammad Amin Fazli · Alireza Ejlali

Received: 21 February 2024 / Accepted: 23 October 2024 / Published online: 14 November 2024

Abstract Mobile edge computing (MEC) enables computationally intensive tasks to be processed at the network edge to provide low-latency services. However, inefficient task scheduling can negatively impact performance metrics like completion time and energy consumption. This paper proposes CAPL-MEC, an adaptive task scheduling framework that utilizes Levy walk modeling to address mobility patterns in MEC. The system model generates random edge nodes within defined bounds to simulate heterogeneous environments. A power consumption model is also presented to optimize dynamic and static power. Device mobility follows an adaptive Levy walk distribution where the power law exponent is time-varying. Latency and reliability (task replication) models are also defined. The CAPL-MEC algorithm utilizes an adaptive Levy walk approach to predict device locations and schedule tasks accordingly. A hybrid task allocation strategy combines proximity awareness, mobile-centric execution, and handovers between mobile and edge devices. Simulations evaluate CAPL-MEC across metrics like completion time, energy consumption, CPU and memory utilization, and wait times under various configurations. Results demonstrate that CAPL-MEC outperforms other algorithms by minimizing completion time through efficient resource allocation based on predicted mobility patterns. Energy consumption is also reduced through power-conscious scheduling. The framework presents a practical and adaptable solution for task scheduling in dynamic MEC environments.

Keywords Mobile edge computing · Power management · Levy walk · Reliability · Task scheduling

1 Introduction

Mobile Edge Computing (MEC) has emerged as a transformative technology to address the growing demand for low-latency and resource-intensive applications. By bringing cloud computing capabilities closer to data sources, MEC supports various real-world applications. Despite its potential, MEC presents significant challenges in task scheduling and resource allocation, particularly in dynamic and heterogeneous environments. The core problem this research addresses is how to efficiently schedule tasks and allocate resources in MEC systems while minimizing power consumption and completion times amidst constantly changing user mobility patterns [1–3]. For

A. Younesi (✉) · M. A. Fazli (✉) · A. Ejlali
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
e-mail: abolfazl.yunesi@sharif.edu

M. A. Fazli
e-mail: fazli@sharif.edu

A. Ejlali
e-mail: ejlali@sharif.edu

instance, Microsoft (Azure Edge[1][2]) and Amazon (AWS Wavelength[3]) integrate MEC into their cloud services to provide seamless and responsive experiences for latency-sensitive applications, such as virtual reality and live-streaming [4, 5].Traditional task scheduling approaches often fail to adapt to these dynamic conditions, leading to suboptimal resource utilization and increased energy consumption. One promising solution to these challenges is utilizing the Levy Walk model to estimate the mobility patterns of registered devices in MEC environments [6, 7].

This paper presents a novel approach, CAPL-MEC, that improves completion time and reduces power consumption in MEC environments by exploiting the characteristics of the Levy walk. This stochastic process exhibits a long-tail distribution and is widely used to model the movement of animals and humans in search of resources [6–8]. The key innovation in CAPL-MEC lies in its use of the adaptive Levy walk model, which adaptively determines its parameters in real time, depending upon the environment, user mobility, and task scheduling demands. Unlike the traditional static models, the adaptive Levy walk keeps learning and updates the mobility patterns continuously to make a far better prediction of the future locations of devices. This provides the capability of a system to optimize resource allocation and task scheduling in a truly responsive way to variability, which is inherently present in mobile edge environments.

Our proposed method employs adaptive levy walk to dynamically model the mobility patterns of users and devices in the MEC environment. According to its prediction about the device location change, the system can perform the run-time adjustment with respect to resource allocation and task scheduling [9, 10]. Therefore, using this model allows our system to predict the future locations of mobile devices and users across the network. This novel adaptive approach significantly outperforms traditional models (see Section 2) by minimizing communication delays and reducing power consumption, making it ideal for real-world applications like video streaming, where energy efficiency and low latency are critical. For example, imagine a user streaming a video on their mobile device while on the move. With CAPL-MEC (see Fig. 1), the system would analyze the user's movement patterns and allocate resources to ensure smooth video streaming while minimizing energy consumption, leading to longer battery life. However, this technique may be challenging due to hardware and software compatibility differences across devices and platforms.

The proposed approach demonstrates significant potential for advancing MEC by simultaneously reducing energy consumption and completion time while accommodating dynamic user mobility patterns [11]. By utilizing Levy walk computing within MEC environments, adaptive algorithms can be developed to allocate resources and dynamically schedule tasks efficiently [12]. This approach proves especially beneficial in highly volatile urban environments, where user and device movement patterns frequently change [13, 14]. Moreover, it opens avenues for novel applications that demand low-latency and high-performance computing, such as autonomous driving and augmented reality [15]. In summary, the CAPL-MEC model holds promise as an optimization strategy to improve completion time and reduce power consumption within MEC environments [6, 16].

In addition to completion time and energy efficiency, reliability and latency are important performance metrics that must be considered for task scheduling in MEC environments. Reliable execution of latency-sensitive tasks is crucial, especially for applications with strict deadlines, such as augmented reality, autonomous vehicles, and telehealth systems [17]. Device failures, network disruptions, and unpredictable bandwidth and computational load variations can negatively impact reliability and increase latency [18]. To address these challenges, CAPL-MEC incorporates models to optimize reliability and latency. The reliability model utilizes active replication to increase the probability of on-time task completion in the presence of failures. Meanwhile, the latency model accounts for dynamic communication and computation delays based on parameters like distance, bandwidth, and computational load. By minimizing latency through efficient scheduling while maximizing reliability via replication-based redundancy, CAPL-MEC aims to provide low-latency and dependable execution of tasks across heterogeneous mobile edge environments.

[1] https://azure.microsoft.com/en-us/solutions/private-multi-access-edge-compute-mec#Solution-overview

[2] https://azure.microsoft.com/en-us/products/iot-edge#started

[3] https://docs.aws.amazon.com/wavelength/latest/developerguide/what-is-wavelength.html

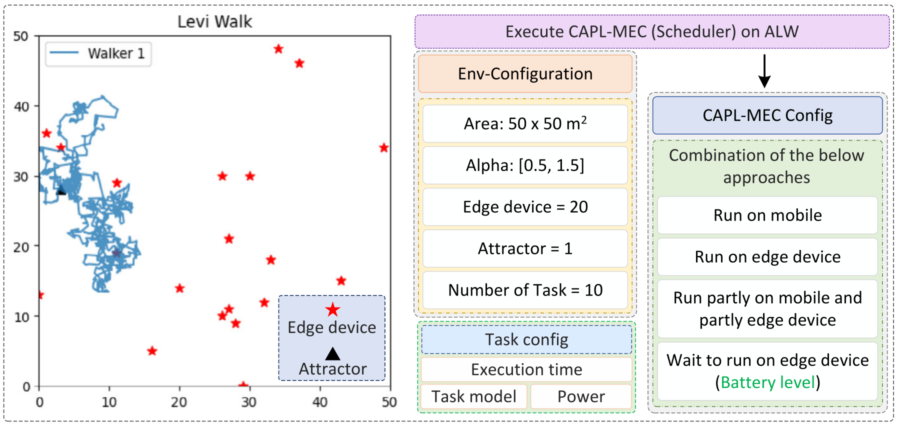

Fig. 1 Execution of the Adaptive levy walk algorithm based on CAPL-MEC Scheduler

1.1 Our Novel Contribution and Limitations

Existing approaches for completion time optimization in MEC have mainly focused on offloading tasks to edge servers with the shortest processing time or minimizing the communication latency between mobile devices and edge servers [19–24]. However, these approaches need to consider the system's energy consumption with task scheduling and resource allocation, which is becoming an increasingly important factor in mobile computing. Furthermore, previous studies did not use Levy walk to model the mobility patterns of users and devices, which is a unique and innovative aspect of our approach. Overall, our proposed method, CAPL-MEC, has the potential to significantly improve the performance and energy efficiency of MEC systems, making it a valuable contribution to the field of MEC. Therefore, incorporating the energy consumption factor and using the Levy walk model can provide a more comprehensive understanding of mobile computing systems.

This research paper suggests a new efficient and adaptable MEC task scheduling method incorporating Levy walk modeling. Although Levy walks have been widely used for modeling complex searches and movement patterns [6, 7], their potential for optimizing resource allocation in MEC has yet to be thoroughly explored. The proposed CAPL-MEC algorithm leverages the predictive capabilities of Levy walks to anticipate device mobility, offering an approach to dynamic task scheduling in MEC environments. Specifically, the algorithm integrates the Adaptive Levy Walk (ALW) model to forecast device movement patterns and allocate resources based on anticipated device locations. The inherent adaptability of Levy walks enables practical exploration in various conditions, allowing CAPL-MEC to dynamically adjust task scheduling in response to rapidly changing user mobility. Integrating bio-inspired Levy concepts into MEC systems fosters adaptive solutions for computation offloading, ultimately improving task completion time and energy efficiency. The proposed integration of ALW into the CAPL-MEC framework, which optimizes mobility prediction, task allocation, and resource management simultaneously, represents a significant contribution to MEC. Existing approaches must sufficiently address device movement patterns, which highly impact task completion times and energy consumption in real-world deployments. This paper proposes CAPL-MEC, an adaptive algorithm utilizing Levy walk modeling to predict mobility patterns and optimize resource allocation. **The main contributions (See Fig. 2) of this paper are as follows:**

- **Adaptive Task Scheduling Algorithm:** Introducing a new adaptive task scheduling algorithm (CAPL-MEC) tailored for MEC environments, which exploits Levy walk modeling to predict device mobility patterns and optimize resource allocation.
- **Integration of Adaptive Levy Walk Model:** Integrating a time-varying Adaptive Levy Walk model into the scheduling framework to capture the dynamics of device mobility and user movement patterns.
- **Holistic Optimization Models:** Developing comprehensive models for power consumption, latency, and reliability to optimize the performance of MEC systems holistically.

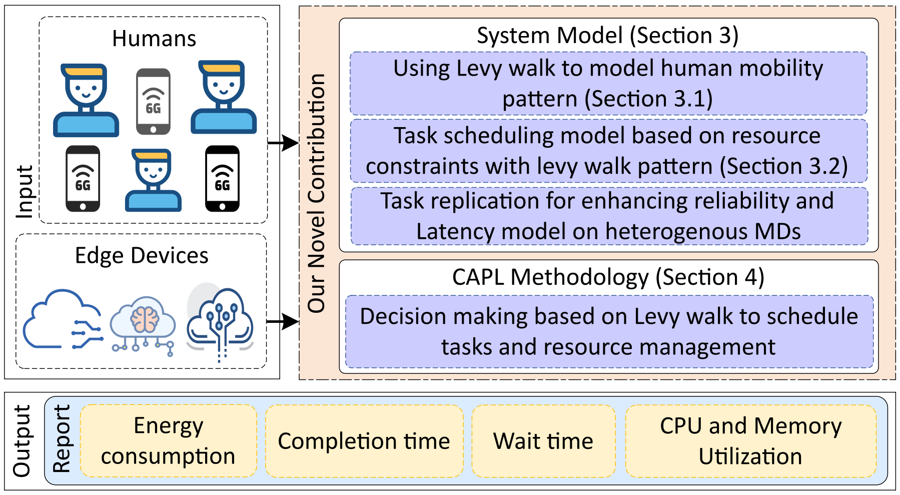
Input
Humans
6G
Edge Devices
Our Novel Contribution
System Model (Section 3)
Using Levy walk to model human mobility pattern (Section 3.1)
Task scheduling model based on resource constraints with levy walk pattern (Section 3.2)
Task replication for enhancing reliability and Latency model on heterogenous MDs
CAPL Methodology (Section 4)
Decision making based on Levy walk to schedule tasks and resource management
Output
Report
Energy consumption
Completion time
Wait time
CPU and Memory Utilization

Fig. 2 An overview of our novel contribution

- **Hybrid Task Allocation Strategy:** Proposing a hybrid task allocation strategy that combines centralized and decentralized approaches for efficient load balancing across mobile and edge devices.
- **Hierarchical Execution Scheme:** Designing a hierarchical execution scheme that enables collaboration between mobile devices and edge nodes, accounting for deterministic and stochastic factors.

1.2 Paper Structure

The present paper follows a structured organization. Section 2 comprehensively reviews resource management and task scheduling in MEC systems. Section 3 presents our system model, outlining its essential components and functionalities. Section 4 delves into the problem formulation, explaining the proposed algorithm and conducting a thorough analysis of its performance. Furthermore, Section 5 provides an in-depth discussion of the simulation results, exploring their implications and highlighting key findings. Finally, Section 6 summarizes the research findings, emphasizing remarkable contributions, and suggests potential avenues for future research endeavors.

2 Related Work

MEC's domain is rapidly advancing, as evidenced by the substantial scholarly literature on task scheduling and resource management. Typically, task scheduling studies aim to optimize the allocation of computational resources and task scheduling in MEC systems, enhancing their overall efficiency and performance. Additionally, these investigations address the challenges associated with real-time processing, energy consumption, and network latency within MEC environments.

Previous research on task allocation in MEC settings primarily focused on centralized or decentralized approaches. While centralized strategies often face scalability limitations, decentralized strategies exhibit greater latencies and require accurate information about resource availability. Our research introduces a novel hybrid task allocation strategy to overcome these challenges. This strategy combines the benefits of centralization and decentralization to maximize task allocation efficiency in MEC environments. We have conducted a comprehensive comparison with related works to ensure the robustness of our proposed method. The results of this comparison are presented in Table 1.

2.1 Task Scheduling

Liu et al. [25] adopt a task scheduling policy employing a Markov decision process approach, wherein task scheduling hinges upon the queuing state of the task buffer, the execution state of the local processing unit, and the transmission unit's state. They formulate a power-constrained delay minimization problem to mitigate average delay and average power consumption for mobile devices.

Table 1 Comparison of the characteristics of related works

Ref.	Year	Energy	Task type	Levy walk	Completion time
[25]	2016	–	N/A	–	–
[26]	2018	+	DAG	–	+
[27]	2022	–	N/A	–	–
[28]	2020	–	N/A	–	–
[29]	2016	+	N/A	–	–
[34]	2022	+	N/A	+	+
[35]	2021	–	N/A	–	+
[36]	2023	–	N/A	+	–
[31]	2022	–	DAG	–	–
[37]	2021	+	Independent tasks	dataset	+
[32]	2024	–	DAG	–	–
[30]	2019	+	N/A	–	+
[38]	2024	+	DAG	–	–
[39]	2024	–	N/A	–	+
CAPL-MEC	2024	+	Periodic	+	+

Guo et al. [26] employ a Directed Acyclic Graph (DAG) to model tasks and propose a distributed eDors algorithm that exploits CPU clock frequency control to reduce energy consumption and application completion time. Researchers presented in [27] introduce a three-layer framework for scheduling Internet of Things (IoT)-generated workflows within MEC environments, employing proactive resource provisioning. They utilize a multi-layer feed-forward Artificial Neural Network (ANN) model, trained using the opposition-based version of the Marine-Predator Algorithm (OMPA) to minimize makespan.

In another study by authors [28], they propose a task scheduling approach for parallel and sequential offloading computationally intensive tasks to multiple MEC servers. Their methodology uses genetic algorithms and conflict graph models to minimize offloading latency and failure probability. In the work described by authors [29], a stochastic approach is proposed for computing tasks arriving at multiple Mobile Devices (MDs), aiming to examine the tradeoff between energy consumption and latency.

In 2019, authors in [30] maximize network lifetime for mobile IoT devices executing approximate real-time tasks under QoS constraints. An offline MILP solution derives the optimal mobility-aware task schedule, avoiding redundant overlapping executions. An online cross-entropy heuristic adapts task execution to fluctuating QoS requirements at runtime. Mobility awareness avoids unnecessary overlapping executions, prolonging network lifetime.

Another paper in 2022 [31] addresses task scheduling in heterogeneous MEC networks. It proposes scheduling tasks across different MEC nodes with varying computational capabilities. The objective is to improve system performance by efficient utilization of heterogeneous resources. However, the paper does not provide details on the specific task model, energy considerations, mobility patterns, completion time metrics, or the methodology employed.

In another work [32], authors propose a mobility-aware framework for joint task scheduling and resource allocation in cooperative MEC networks. It considers user mobility patterns from a real-world dataset. It formulates an optimization problem to minimize task execution latency while accounting for distributed computing resources, task characteristics, and energy constraints of mobile devices. The paper presents two approaches: a genetic algorithm (GA) and a heuristic method (MATS) to solve the optimization problem.

Finally, in [33], authors tackle the challenging problem of distributed computation offloading decision-making among multiple MDs by formulating it as a multi-MDs computation offloading game, offering a novel approach to this intricate problem.

2.2 Resource Allocation

In 2022, a machine learning (ML) approach for computation offloading decisions in MEC was proposed [34]. It uses various ML models like decision trees, k-nearest neighbors, and neural networks to predict whether an IoT device should offload its computation to the edge cloud or process it locally. The models are trained on features like available bandwidth, workload, and battery level of the IoT device. However, it does not consider optimizing energy consumption or task completion time.

In 2021 [35], another paper explores the optimal placement of heterogeneous edge servers in a mobile edge-cloud computing environment to minimize the overall response time. It formulates the problem as a mixed-integer non-linear program and proposes different optimization models based on linearization techniques, Benders decomposition, and a heuristic algorithm. The models aim to jointly optimize server placement and request distribution across edge and cloud servers while meeting quality-of-service requirements.

Authors in [36] introduce sampling-based dynamic programming (S-OAMC) and G-OAMC (greedy-based) methods to minimize application turnaround time by predicting future application specifications using matrix completion. These approaches optimize offloading to nearby cloudlets, reducing latency and migration costs. One of the important limitations that they didn't analyze is the energy overhead through their algorithm, along with computational overhead.

Authors in [37] are optimizing UAV trajectory and offloading DAG tasks in a UAV-assisted MEC environment. It aims to minimize latency by jointly optimizing the UAV's flight path and task offloading decisions. The paper considers the precedence constraints of DAG tasks but does not discuss energy aspects, mobility models, completion time objectives, or optimization techniques. I

In another work [39] in 2024, Cao et al. propose a reconfigurable intelligent surface (RIS) aided multi-access MEC heterogeneous network architecture to reduce latency. It formulates an optimization problem to jointly optimize RIS deployment, network association, and resource allocation. The goal is to minimize the maximum computation latency under constraints on RIS deployment locations and communication/computation resource allocation. Simulation results demonstrate significant latency reduction compared to conventional schemes without RIS assistance.

In 2024, the authors of [38] developed a reliability-driven collaboration framework to minimize energy consumption in large-scale cyber-physical systems. The framework enables reliable task offloading between battery-limited end devices and resource-rich edge servers. Offline and online algorithms are proposed to determine the optimal task offloading decisions and energy-efficient operation modes under time-varying computation workloads and channel conditions. Extensive simulations validate the significant energy savings achieved.

3 System Model

Our proposed algorithm intricately improves the task scheduling process by incorporating the inherent mobility patterns of mobile devices. Using the Levy walk, CAPL-MEC demonstrates adaptability across diverse mobility patterns, providing efficient solutions. Hence, the algorithm exploits the potential of the Levy walk to improve task scheduling within MEC environments, considering the mobility patterns of mobile devices and offering efficient solutions for various scenarios. Predicting the future locations of the devices based on their past movements, the algorithm schedules tasks accordingly to minimize communication and computation costs. This, in turn, enhances the intelligent city system's overall performance and provides real-time decision-making insights. The performance evaluation results demonstrate that the proposed algorithm outperforms existing task scheduling algorithms regarding task completion time and energy consumption. Thus, the CAPL-MEC algorithm holds promise as a viable solution for task scheduling in MEC environments. A concise overview of the critical notations used in the system model and Section 4 can be found in Table 2.

3.1 Device and Node Generation

The number of MDs and edge nodes in the system are determined probabilistically at the start of each simulation run. Specifically, the number of MDs is sampled from a normal distribution,

$$MDs \sim N(\mu, \sigma^2) \qquad (1)$$

Table 2 List of Notations

Symbol	Definition
A_t	Arrival time of each task
C_L	The capacity of internal load capacitors
c_{ed}	Closest edge device to walker
c_{idx}	Index of closest edge device
d	The deadline of the tasks
$dist$	List of the distance of the walker
$dist_c$	Minimum distance between selected edge with walker
E_t	Execution time of each task
f	Frequency of device
gr_{exp}	Generate random number based on exponential distribution
gr_{uni}	Generate random number with uniform distribution
I_l	The short-circuit current (Leakage)
m_d	Initial minimum distance
$model$	Levy walk model
n, env	Environment size
N_{ed}	Number of edge devices in the environment with uniform distribution
N_w	Number of walkers in the environment
Opt	Type of scheduling algorithm
$Path(i, j)$	The route on which a walker is walking
P_d	Dynamic power
P_{ed}	Position of the each edge device
P_{idle}	Idle power when a mobile device does not operation
$pos(i, j)$	Current position of walker
P_s	Static power
P_t	Total power consumption
P_w	Initial battery level of walker
r	Step size based on exponential distribution
$task_{ed}$	Task execution device
$thresh$	Range of an edge device that, if walker goes into it, can send the task to execute on edge
V_{dd}	The voltage of the supply
x	Step size in horizental position
y	Step size in vertical position
α	Activity
α_L	Levy walk alpha parameter
θ_0	Degree in horizontal cooredination
θ_1	Degree in vertical cooredination

where μ and σ are configured based on the desired average and variance, the number of edge nodes is derived from a uniform distribution U(a, b) within predefined limits. Furthermore, the resources and capabilities of edge nodes are also subjected to randomized generation. The communication range of each edge node is obtained from a normal distribution,

$$R_c \sim N(\mu_r, \sigma_r^2) \tag{2}$$

to account for heterogeneity. Similarly, uniform sampling produces the computational capacities of edge nodes, including CPU frequency, memory, etc.

$$C_c \sim U(C_{min}, C_{max}) \tag{3}$$

where C_{min} and C_{max} denote the minimum and maximum computational capacity of the edge nodes.

3.2 Device Mobility

Real-world user mobility tendencies exhibit complex characteristics that are challenging to model. Studies show movements contain sporadic short trips and occasional longer flights, resembling patterns observed in animals foraging for resources [40, 41]. This Levy walk behavior is captured using a stochastic process where step lengths follow a power-law distribution.

MDs enter and depart the system with a certain probability pdt at each time step. Equation 4 gives the probability of an MD departing at time step t.

$$P(\text{depart}, t) = \text{pdt} \tag{4}$$

Their locations are initialized randomly within the spatial bounds. During their lifetime, MDs move according to the Levy walk mobility model described in Section 3.5. However, the Levy walk parameters are adapted at runtime based on the density of devices in different regions to simulate realistic mobility patterns. The mobility follows a Levy walk distribution where step length (l):

$$P(l, t) \sim l^{-\alpha t} \tag{5}$$

Here (α_t) represents the time-varying power law exponent. Proof: Consider a mobile device MD moving in a 2D space according to a Levy walk mobility pattern. At each time step t, the MD takes a random length l step in a random direction (θ). It is known that for a Levy walk, the step length l follows a power law distribution:

$$P(l, t) \sim l^{-\alpha} \tag{6}$$

Where (α) is the power law exponent. We now introduce time-dependence in (α) as (α_t). Thus, the distribution becomes:

$$P(l, t) \sim l^{-\alpha_t} \tag{7}$$

To prove this is a valid distribution, we need to show:
1- $P(l, t) \geq 0$
2- $\int P(l, t), d_l = 1$

Proof of 1): For $(l > 0)$ and $(\alpha_t > 0)$, $(l^{-\alpha_t} \geq 0)$ Thus, $(P(l, t) \geq 0)$

Proof of 2): Considering the steps are taken in a 2D space, the direction (θ) is uniform in $([0, 2\pi])$. Therefore, integrating over (θ) yields (2π) due to symmetry.

$$\int P(l, t), d_l = \int (l^{-\alpha_t}), d_l \cdot (2\pi) \tag{8}$$

Let $(u = l^{-\alpha t})$ Then $(d_u = -\alpha t, l^{-\alpha t-1}, d_l)$ Substituting this into the integral:

$$\begin{aligned}\int P(l, t), d_l &= 2\pi \int u \left(\frac{d_u}{-\alpha_t}\right) \\ &= \frac{2\pi}{\alpha_t} \int u, d_u \\ &= \frac{2\pi}{\alpha_t} \cdot \frac{1}{2} u^2 \Big|_0^\infty \\ &= \frac{\pi}{\alpha_t}\end{aligned} \tag{9}$$

Thus, $(\int P(l, t), d_l = 1)$

Therefore, we have proven that $(P(l, t))$ is a valid probability distribution at each time step (t).

3.3 Power Model

The amount of power consumed within an MEC framework depends on various factors. Given that mobile devices operate on batteries with limited capacities, it becomes imperative to devise power-efficient algorithms. Within these environments, two forms of power consumption can be discerned: dynamic power, encompassing the power utilized during the functional phase, and static power, referring to the power employed during the passive phase [42, 43]. Hence, to optimize power consumption in an MEC setting, it becomes indispensable to consider both dynamic and static power consumption while concurrently developing efficient algorithms to minimize overall power utilization. By doing so, not only can the battery life of mobile devices be enhanced, but the ecological ramifications of computing can also be mitigated. In the functional phase, power consumption is contingent upon the nature of the task at hand, as each task manifests a distinct activity level [43]. Moreover, during the passive phase, devices continue to consume power even without active tasks [42, 43]. Equation 10 illustrates the formula for power consumption:

$$P_t = P_d + P_s \tag{10}$$

$$P_d = \alpha \cdot C_L \cdot V_{dd}^2 \cdot f \tag{11}$$

Where α represents the activity associated with each task,(C_L) represents the capacitance of the internal load capacitors, (f) presents the operational frequency of the device, and the voltage supplied (V_{dd}). The computation of passive state power consumption can be obtained by employing equation (12):

$$P_s = V_{dd} \cdot I_l \tag{12}$$

V_{dd} signifies the voltage the power supply provides, while I_l represents the short-circuit current.

3.4 Latency Model

The communication latency L_c incurred when offloading tasks from an MD to an edge node is modeled as:

$$L_c(t) = T_t + T_p \tag{13}$$

where,T_t is the transmission latency which depends on the bandwidth $B(t)$ available at time (t):

$$T_t(t) = \frac{D(t)}{B(t)} \tag{14}$$

$D(t)$ is the data size of the task in bits at time t. $B(t)$ is sampled from a normal distribution:

$$B(t) \sim N(\mu_b, \sigma_b^2)(T_p) \tag{15}$$

is the propagation latency proportional to the distance $d(t)$ between the MD and edge node [44]:

$$T_p(t) = \beta \cdot d(t) \tag{16}$$

here β is a constant propagation factor. $d(t)$ is derived dynamically from the mobility model. The computation latency on the edge node is defined as [44, 45]:

$$L_{\text{comp}}(t) = \frac{C(t)}{F} \tag{17}$$

where $C(t)$ is the CPU cycles required for the task at a time (t), and (F) is the edge node's computational capability. The total latency is then:

$$L_{\text{total}}(t) = L_c(t) + L_{\text{comp}}(t) \tag{18}$$

This mathematical formulation allows latency to be modeled at each time step based on dynamic factors like distance between nodes, data size, and computational load.

3.5 Reliability Model

To improve the reliability of computation offloading, we utilize active replication by duplicating tasks to multiple edge nodes. Let r be the redundancy factor, i.e., the number of copies of each task. The reliability $R(t)$ at time t is:

$$R(t) = 1 - (1 - \rho(t))^r \tag{19}$$

where $\rho(t)$ is the reliability of an individual edge nodes at time t [42, 43].

$$(\rho(t) = e^{-\lambda t}) \tag{20}$$

Here λ is the failure rate of an EN.

The overhead (O(t)) caused by replication is:

$$O(t) = r \cdot C(t) \tag{21}$$

Where $C(t)$ is the computational cost of each task at time t.

To optimize the trade-off between reliability and overhead, the redundancy factor r is dynamically tuned using the below threshold-based policy:

If $R(t) < R_{\text{target}}$: $r = r + 1$ else if $R(t) > R_{\text{max}}$: $r = r - 1$ Where R_{target} is the desired reliability level, and R_{max} is the maximum permissible reliability.

This model allows the replication factor to be adapted at runtime based on the current reliability to meet targets while minimizing overhead. The parameters λ, R_{target}, and $R_{\max}$ can be tuned to configure the system as per reliability needs.

3.6 Scheduling Model

Capitalizing on proximity awareness and resource availability, our proposed hybrid task (see Algorithm 1) allocation strategy for MEC environments intelligently evaluates task requirements to make informed decisions on task placement. This adaptive approach dynamically assesses the prevailing conditions and optimizes resource utilization to enhance system performance. A stochastic component ensures fairness and diversity in task execution, while a hierarchical execution model facilitates a seamless transition between mobile and edge device execution. Specifically, when the mobile device is within the coverage area of the edge device, task execution is allocated to the mobile device. In contrast, if the mobile device falls beyond the range of the edge device, the task is assigned to the edge device. A stochastic task assignment scheme is employed, wherein the decision to execute a task on either the mobile device or the edge device is randomly determined. Furthermore, proximity awareness is established through the assessment of the separation between the mobile device and the edge device, allowing for task execution to be initially initiated on the mobile device and subsequently transitioned to the edge device when the device goes beyond its range. Our hybrid task allocation strategy combines range assessment, mobile-centric execution, random selection, and handover for efficient and adaptive task allocation in MEC environments.

3.7 Levy Walk Model

The Adaptive Levy Walk (ALW) algorithm, depicted in Algorithm 2, capitalizes on the Levy flight concept, which characterizes long-range displacements with step-length distributions exhibiting heavy tails. The algorithm introduces an adaptive parameter, denoted as alpha (see Table 3), which governs the scaling factor of step sizes. This adaptive mechanism ensures that individual walkers explore the environment with appropriately tailored step sizes, thereby enabling efficient exploration and exploitation of the search space. Empirical studies have demonstrated the superior performance of the ALW algorithm over conventional random walk and Levy walk algorithms across diverse applications, particularly in the realm of optimization problems [6]. An additional advantage lies in the algorithm's ease of implementation and customizability to specific application domains.

The dynamic adjustment of the alpha parameter is contingent upon the vertical position of the walkers within the system. Mathematically, the alpha value is computed according to the following expression: The computation of the alpha value, denoted as α_L, is given by the formula:

$$\alpha_L = \alpha_{\min} + \beta \cdot \frac{(n-y)}{n}, \quad \begin{cases} \beta \in [0.0, 1.0], \\ 0.5 \le \alpha \le 1.5, \\ \alpha_{\min} = 0.5, \\ \alpha_{\max} = 1.5. \end{cases} \tag{22}$$

let $\alpha_{\min}$ and $\alpha_{\max}$ symbolize the lower and upper bounds of the variable α, respectively, while n denotes the dimensions of the system grid. The parameter β

Table 3 Effect of different α values in Levy walk, LW: Levy Walk

Perspective	LW ($\alpha = 0.5$)	LW ($\alpha = 1$)	LW ($\alpha = 1.5$)
Step Length Distribution	More pronounced heavy-tailed distribution with higher probability of long jumps	Heavy-tailed distribution with normal probability of long jumps	Heavy-tailed distribution with lower probability of long jumps
Convergence Speed	Slower convergence	Normal convergence speed	Faster convergence
Sensitivity to Initial Conditions	Less sensitive	Normal sensitivity	Slightly more sensitive

Algorithm 1 CAPL-MEC.

```
Require: $P_w$, $N_{ed}$, $N_w$, steps, $P_{ed}$, env, $P_{idle}$, model, Opt, $m_e$
Ensure: Suitable task scheduling

1: Initialization: Thresh = gr_uni(), $C_{ed}$ = None, $m_d = \infty$, dist = []

2: $tasks \leftarrow [(A_t, E_t, d)]$
3: for $i \leftarrow 0$ to $tasks$ do
4:     $task\_set \leftarrow$ read_tasks()
5:     $Path \leftarrow$ ALW(pos, steps, env)
6:     for $j \leftarrow 0$ to $steps$ do
7:         $Walker\_pos \leftarrow$ all $path[:, j, :]$
8:         for $k \leftarrow 0$ to $N_{ed}$ do
9:             Add $\sqrt{\sum(P_{ed} - \text{walker\_pos})^2}$ to $dist$
10:        end for
11:        $C_{idx} \leftarrow \arg\min(dist)$
12:        $dist_c \leftarrow dist[C_{idx}]$
13:        if $dist_c < m_d$ then
14:            $m_d \leftarrow dist_c$
15:            $C_{ed} \leftarrow C_{idx}$
16:        end if
17:        if $m_d \geq$ Thresh and $m_e \geq 50$ then
18:            $task_{ed} \leftarrow$ "mobile"
19:        else
20:            $task_{ed} \leftarrow$ "edge_device"
21:        end if
22:        if Opt == 1 then
23:            if $task_{ed}$ == "mobile" then
24:                Execute all tasks on mobile
25:            else if randomly select execution device then
26:                Execute task on the selected device
27:            else if $task_{ed}$ == "edge_device" and in range of edge device then
28:                Execute on the edge device
29:            else if $task_{ed}$ == "edge_device" and not in range then
30:                Execute the task on mobile until the walker reaches range
31:            else
32:                No task
33:            end if
34:        end if
35:        Consume $P_{idle}$
36:    end for
37: end for
```

embodies the scope encompassing plausible values of α.

The algorithm enables random movement for each mobile device, employing an exponential probability density function to determine step distances and a uniform probability density function to determine step angles. The lengths of these steps are subject to alteration through an exponentiation process involving an exponent of $1/\alpha_L$, thereby instigating adaptivity within the magnitudes of the steps. Trigonometric functions are then invoked to ascertain the resultant x and y components ensuing from the steps' progression.

$$r = \exp\left(\frac{1}{\alpha_L}\right) \cdot (\cos(\theta_0), \sin(\theta_1)) \tag{23}$$

$$y = \text{pos}(i, j) + (r(i, j))^{\frac{1}{\alpha(y)}} \cdot \sin(\theta(i, j)) \tag{24}$$

$$x = \text{pos}(i, j) + (r(i, j))^{\frac{1}{\alpha(y)}} \cdot \cos(\theta(i, j)) \tag{25}$$

$$\text{Path}(i, j) = (x, y) \tag{26}$$

Let r be the progression of the pedestrian, pos denote the current location, $path$ signify the trajectory along which the pedestrian is traversing, and θ_0, θ_1 denote the horizontal and vertical inclinations, respectively.

Algorithm 2 Adaptive Levy Walk (ALW).

```
1: procedure ALW(pos$(i, j)$, $steps$, $env$)
2:     Initialization: $\alpha_{\min} = 0.5$, $\alpha_{\max} = 1.5$, $\beta = \alpha_{\max} - \alpha_{\min}$
3:     for $i = 0$ to $steps - 1$ do
4:         $x, y \leftarrow$ calculate_mean_position(pos$(i, j)$)
5:         $\alpha \leftarrow \alpha_{\min} + \beta \cdot \frac{n-y}{n}$
6:         $r \leftarrow$ gr_exp(pos.rows, 2, scale = 1)
7:         $\theta \leftarrow$ gr_uni(pos.rows, 2, low = 0, high = $2\pi$)
8:         $x \leftarrow (r[:, 0]^{1/\alpha}) \cdot \cos(\theta[:, 0])$
9:         $y \leftarrow (r[:, 1]^{1/\alpha}) \cdot \sin(\theta[:, 1])$
10:        Add $x$ to pos[:, 0]
11:        Add $y$ to pos[:, 1]
12:        pos[:, 0] $\leftarrow$ clip(pos[:, 0], 0, $n - 1$)
13:        pos[:, 1] $\leftarrow$ clip(pos[:, 1], 0, $n - 1$)
14:        path$[i] \leftarrow$ pos
15:    end for
16: end procedure
```

4 Methodology

4.1 Problem Formulation

In the computationally intensive MEC task paradigm, efficient schedules of computationally demanding workloads on resource-limited devices are paramount to achieving stringent performance goals while ensuring the optimal use of resources. However, existing scheduling algorithms often must address complex challenges posed by MEC environment dynamics, computational complexity, power constraints resulting from resource-limited devices, and uncertainty generated by different network conditions. Developing an advanced and sophisticated framework that can effectively address these complex issues is imperative. The study proposes a Levy walk-based framework for programming MEC tasks that require power, and its inherent adaptation, robustness, and efficient exploration and exploitation potential will overcome the limitations of existing programming methods and lead to improved performance.

The main objective of the study is to formulate and solve the following multi-faceted problems: Given the complexity of MEC tasks with power consumption, different computational requirements, different workloads, and stringent Quality-of-Service (QoS) constraints, coupled with a dynamic MEC environment, including resource-limited devices with different capabilities and different network conditions, the overall objective is to develop a highly optimized and adaptive scheduling framework that can achieve the following objectives:

- **Reducing Task Completion Time:** Design a smart task scheduling strategy to minimize the total task completion time, ensure efficient use of computational resources, and meet tasks' QoS requirements.

$$T_{CT} = \min\left(\max\left(\sum_{i=1}^{n} M_{et}, \sum_{j=1}^{m} E_{et}\right)\right) \tag{27}$$

  Let M_{et} denote the execution time of tasks on mobile devices, and E_{et} represent the execution time of tasks on edge devices.

- **Optimizing Energy Consumption:** Develop advanced energy management techniques that dynamically allocate energy resources to tasks and devices while adjusting the energy constraints imposed by resource-limited devices. Energy management strategies should effectively optimize energy consumption by intelligently distributing energy resources to balance performance requirements and power limitations.

$$T_E = \min(E_{TA}, E_{DA}) \tag{28}$$

  where E_{TA} represents the energy consumption resulting from task allocation to devices, and E_{DA} denotes the energy consumption associated with the execution time of allocated tasks on specific devices.

$$E_{TA} = \sum_{m=1}^{\kappa} (E_{tm} + E_{cm}) \tag{29}$$

$$E_{DA} = \sum_{n=1}^{\nu} (E_{te} + E_{ce}) \tag{30}$$

  where E_{tm} and E_{te} represents the computation energy consumed by executing a task on the mobile device and edge device, and E_{cm} and E_{ce} are the communication energy for offloading a task from the mobile device to the edge device over the network.

- **Mitigating the Effects of Dynamic Network Conditions:** Consider network conditions' variability, bandwidth, latency, and availability changes and incorporate network awareness into the planning framework. Establish mechanisms to adapt task scheduling and resource allocation to network conditions and optimize task completion times, resource usage, and energy consumption.

The Levy walk-based framework is proposed to provide comprehensive and innovative solutions to these interrelated sub-problems. It takes advantage of Levy walk's intrinsic properties, including its ability to explore different environments, balance exploration-exploitation tradeoffs, and adapt to dynamic contexts. With the integration of Levy walk behavior into the scheduling framework, significant improvements are expected in task schedule efficiency, reduced task completion time, improved energy consumption, and resource utilization in MEC environments, thus paving the way for improved overall system performance and user experience.

4.2 Algorithm Discussion

The CAPL-MEC algorithm aims to optimize task scheduling by considering various factors. A random threshold value is generated, and essential variables are initialized. Subsequently, the algorithm receives a numbered list of tasks with distinct parameters (line 1). The Path calculation utilizes the Adaptive Levy Walk (ALW) method (lines 2-4). At each iteration (line 5), the algorithm retrieves the coordinates of the pedestrian and calculates the Euclidean distance between the pedestrian and each edge device (lines 5-9). The edge device with the shortest distance is selected as the closest (line 10). If the distance meets specific predetermined criteria and sufficient remaining energy, the task is assigned to either a mobile or an edge device, depending on the task's nature (lines 12-21).

The task execution workflow depends on the state of the option flag (line 22) and the assigned task type. When the option is enabled, tasks are executed based on specified conditions. There are alternative execution options within the primary option (lines 23-32). In cases where no tasks are assigned within a given time interval, idle power consumption is initiated (line 35). The CAPL-MEC algorithm offers a systematic framework for task scheduling, considering device capabilities, energy limitations, and distance-related criteria.

The ALW algorithm (See Algorithm 2) models the movement patterns of walkers within a defined search space. It uses the concept of Levy flights, which characterize long-range movements with step-length distributions with heavy tails. The algorithm introduces an adaptive parameter α that governs the scaling of step sizes, enabling individual walkers to explore the environment with tailored step lengths based on their position. This adaptive mechanism allows for efficient exploration and exploitation of the search space.

The algorithm initializes key variables in the first step. `pos` represents the initial position matrix of walkers, with each row vector giving a walker's (x, y) coordinates. `steps` defines the total number of iterations, and `env` specifies the bounds of the search space. $\alpha_{\texttt{min}}$ and $\alpha_{\texttt{max}}$ set the lower and upper limits of the adaptive α parameter.

The walkers' movements are determined in each iteration from steps 1 to `steps-1`. First, the `mean_position()` function calculates the average x and y coordinates of each walker's current position using `pos` as a reference (Line 3). This facilitates the derivation of each walker's adaptive α parameter. α is computed in Line 4 by taking the mean α value $(\alpha_{\text{min}} + \beta)$ and adjusting it proportionally based on the walker's vertical position n within the bounds of the search space. The β term represents the range between α_{min} and α_{max}.

Next, an exponential random number r representing step length is generated in Line 5. A uniform random number θ defining the step angle is produced in Line 6. The new perspective x and y coordinates for each walker are calculated in Lines 7-8 by multiplying r raised to the power of $1/\alpha$ by the cosine and sine of θ, respectively. This exponentiation with the adaptive α incorporates walkers' positions to tailor step sizes. Lines 9-10 add the calculated x values to the first column and y to the second column of `pos` to update each walker's position.

Lines 11-12 employ `clipping` to ensure positions do not exceed the search space boundaries, standard practice in random walk algorithms. Finally, the current positional configuration `pos` is stored in the `path` array at each iteration to log the walkers' trajectories (Line 13).

By carefully orchestrating these computational steps, the ALW algorithm effectively models the complex dynamics of walkers exploring heterogeneous environments with behavior adapted to their positioning. This forms the basis for informing efficient scheduling decisions in heterogeneous edge environments.

5 Simulation Results

This section delves into the intricacies of improving the allocation and scheduling of tasks within a network comprising diverse edge nodes spanning a vast 5000 $\times$ 5000 m^2 system region. Each node possesses a distinct configuration and encompasses a unique coverage radius. Meanwhile, the tasks themselves exhibit varying levels of complexity and processing duration.

Our primary objective is to curtail the overall power consumption while adhering to the resource constraints imposed by each node. To accomplish this, we present a remarkable algorithm that meticulously accounts for node heterogeneity and communication delays, thereby enabling the efficient allocation and scheduling of tasks. Our algorithm meticulously balances completion time, wait time, and power consumption, ensuring the optimal completion of real-world tasks ranging from 2 to 10. For an in-depth comprehension of the experimental setup and tool flow, kindly refer to Fig. 3.

5.1 Experimental Setup

To evaluate our method CAPL-MEC, it is compared to four other types of levy walk algorithms with a scheduler utilizing three different performance metrics. The

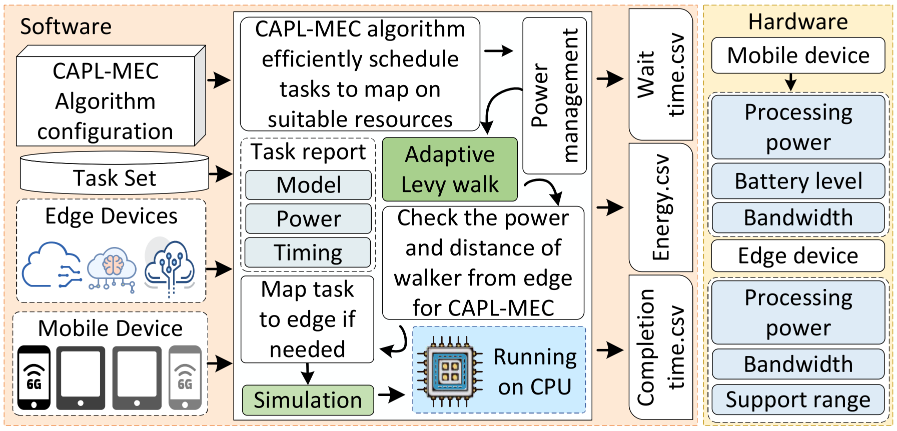

Fig. 3 Experimental setup and Toolflow overview in the paper

evaluation baseline is represented as follows: 1) Energy consumption; 2) Completion time; 3) Waiting time; 4) CPU and Memory utilization. To conduct our analysis, we have categorized tasks into five distinct cohorts, each denoting a cumulative workload of 2, 4, 6, 8, and 10, respectively. The Table 4 summarizes the critical parameters used in the simulations.

5.1.1 Implementation Details and Simulation Environment

The implementation of the CAPL-MEC framework and the associated simulations were carried out using Python. The experiments were conducted in a custom-built simulation environment to emulate MEC scenarios. This simulator was configured to randomly generate edge nodes and mobile devices within a 5000 × 5000 m^2 area, accounting for heterogeneous resources and capabilities of both edge nodes and mobile devices. We will compare our work with [27] and [32] in detail and analysis section

5.1.2 Dataset Information

The simulation dataset consists of synthetically generated tasks [27] and mobility patterns based on the Levy walk model. Tasks vary in complexity and processing duration, while mobile devices follow adaptive Levy walk mobility patterns to simulate realistic movements in an urban environment. The parameters for the Levy walk model were chosen based on empirical studies demonstrating its effectiveness in modeling real-world human mobility patterns. The task complexities and durations were selected to represent a range of real-world applications in MEC environments.

Table 4 Table of parameters used in the simulation

Parameter	Value/Description
Number of Edge Nodes	Uniformly distributed within [10, 30]
Number of Mobile Devices	Normally distributed with mean (μ) and variance (σ^2)
Communication Range	Normally distributed $R_c \sim N(\mu_r, \sigma_r^2)$
Computational Capacity	Uniformly distributed $C_c \sim U(C_{min}, C_{max})$
Levy Walk Alpha (α_L)	Adaptively varied within [0.5, 1.5] based on position
Voltage Supply (V_{dd})	$1.1V$
Short-Circuit Current (I_l)	$0.1A$
Activity Factor (α)	0.5
Internal Load Capacitors (C_L)	$1.0\ \mu F$
Task Data Size ($D(t)$)	Varies based on task
Bandwidth ($B(t)$)	Normally distributed $B(t) \sim N(\mu_b, \sigma_b^2)$

5.1.3 Hardware Details

The experiments' hardware setup consists of edge nodes and mobile devices with varied configurations to simulate a realistic MEC environment. The edge nodes are equipped with Intel Xeon processors ranging from 2 to 8 cores, 8GB to 64GB of RAM, and 128 GB SSD to ensure fast and reliable storage.

Mobile devices are simulated using virtual machines, each representing a mobile device with a single-core processor and 1GB RAM. These virtual mobile devices also have simulated battery models that reflect real-world power consumption parameters.

The overall simulation environment runs on a high-performance computing server in Sharif University of Technology with multiple Intel Xeon processors, 128GB of RAM, and large-capacity SSD storage. The software stack includes Python 3.8, Jupyter Notebooks, and Docker, all operating on Ubuntu 20.04 LTS.

5.2 Detail and Analysis

The top-left in Fig. 4 depicts the normalized average energy consumption across different methods. CAPL-MEC consistently shows lower energy consumption than TD3 [32] and OMPA [27]]. This improved performance can be attributed to CAPL-MEC's power-conscious scheduling approach, which optimizes dynamic and static power consumption. By leveraging the adaptive Levy walk model, CAPL-MEC accurately predicts the mobility patterns of devices. This allows for efficient task scheduling, reducing unnecessary energy expenditure. The paper highlights that CAPL-MEC's power model accounts for the devices' active and idle states, ensuring overall energy efficiency.

The top-right in Fig. 4 illustrates the normalized completion time for the tasks. CAPL-MEC demonstrates shorter completion times than TD3 [32] and OMPA [27]. This advantage stems from CAPL-MEC's adaptive task scheduling algorithm, which utilizes predicted mobility patterns to allocate resources effectively. The method minimizes communication and computation delays by dynamically adjusting the task allocation based on real-time conditions. The hybrid task allocation strategy employed by CAPL-MEC combines proximity awareness with dynamic task allocation, ensuring that tasks are executed optimally on

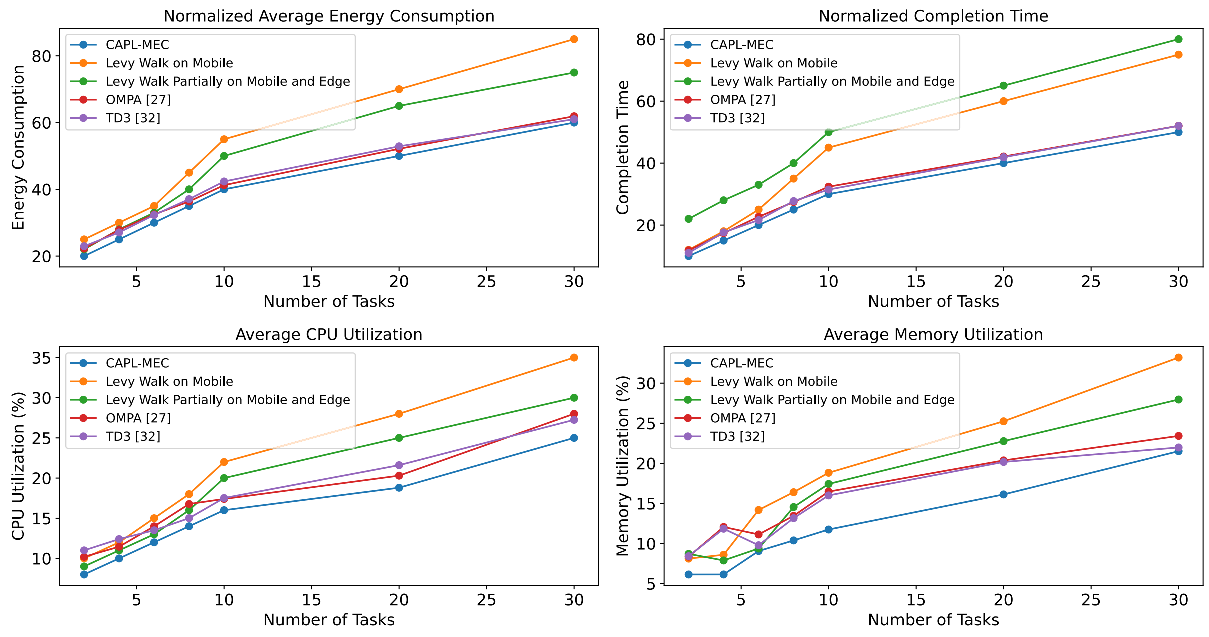

Fig. 4 Performance Comparison of CAPL-MEC, TD3 [32], OMPA [27], Levy Walk on Mobile, and Levy Walk Partially on Mobile and Edge Methods, Top-left illustrates the energy consumption, top-right depicts Completion time, bottom-left demonstrates Cpu utilization and bottom-right shows memory utilization

mobile or edge devices, thus accelerating the overall task completion process.

The bottom-left in Fig. 4 presents the average CPU utilization. CAPL-MEC demonstrates lower CPU utilization than TD3 [32] and OMPA [27], indicating a more efficient use of processing resources. This efficiency directly results from CAPL-MEC's sophisticated scheduling and resource allocation mechanisms, ensuring a balanced task distribution across devices. By predicting device mobility through the adaptive Levy walk model, CAPL-MEC prevents overloading any single device, optimizing CPU usage. This approach guarantees that the computational load is evenly distributed, avoiding CPU bottlenecks and maintaining system performance.

The bottom-right in the Fig. 4 presents the average memory utilization for the different methods. CAPL-MEC again shows superior performance with lower memory utilization compared to TD3 [32] and OMPA [27]. The efficient management of memory resources in CAPL-MEC is achieved through its predictive task scheduling, which dynamically allocates tasks based on device mobility and resource availability. This prevents memory bottlenecks and ensures smooth operation even under varying workloads. The paper explains that CAPL-MEC's memory management strategy optimizes available memory by balancing the task load, thus reducing overall memory usage.

5.3 Energy Consumption

Figure 5 visually represents the normalized average energy consumption associated with different scheduling models in our system model. Examining the points along the horizontal axis allows us to observe how the workload is distributed between edge devices and MDs while maintaining a constant energy consumption.

The examination of Fig. 5 highlights the importance of choosing the appropriate scheduling model in determining the system's normalized average energy consumption. The x-axis of the graph represents the number of tasks, while the y-axis depicts the normalized energy consumption when executing these tasks using the presented approaches. As the workload increases, the superior performance of the proposed method becomes evident (represented by the green column in Fig. 5, which consistently exhibits relatively low power consumption despite workload intensification).

The workload allocation between edge devices and MDs significantly influences energy consumption patterns. Assigning a more significant proportion of the workload to edge devices results in higher overall power consumption for the alternative methods. In contrast, the proposed CAPL-MEC method demonstrates remarkable energy efficiency, enabling edge devices to handle greater workloads without substantially increasing energy consumption.

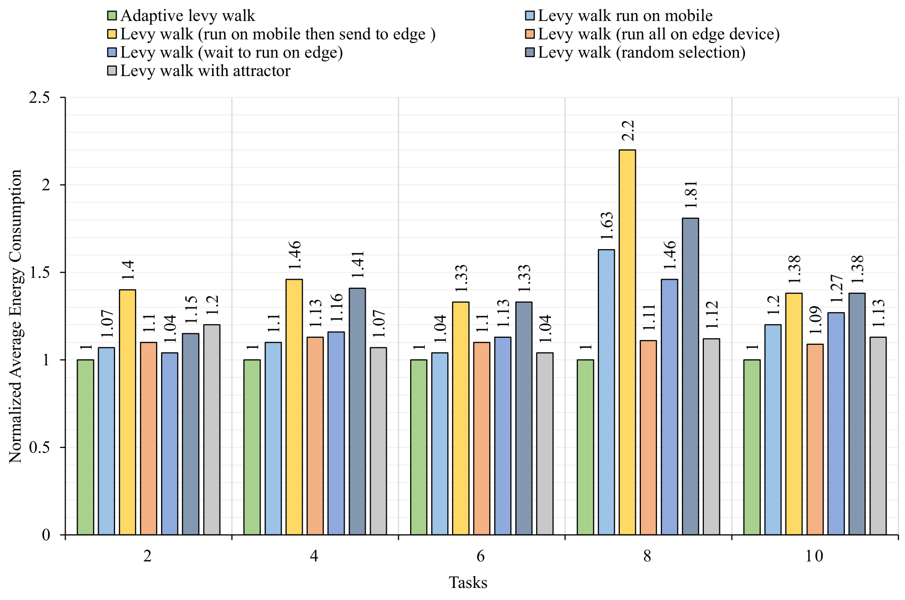

Fig. 5 Normalized average energy consumption per task (lower is better)

5.4 Completion Time

Figure 6 compares the average completion time across different scheduling methods applied to system models. The x-axis of the graph corresponds to the number of tasks, while the y-axis illustrates the normalized completion time when executing the functions with the presented approaches.

The data has been meticulously normalized to ensure a fair and accurate evaluation. The findings unequivocally demonstrate that the CAPL-MEC scheduling method offers a remarkable reduction in execution time compared to the other examined models, except methods 2 and 3 (represented by columns 2 and 3 in Fig. 6) that run on mobile devices. However, it is crucial to note that these methods result in higher power consumption, leading to increased temperatures in MDs and quicker battery depletion, which holds substantial importance for MDs. Despite the proposed method's notable advantage, it is essential to consider the marginal nature of the discrepancy in completion time between CAPL-MEC and the alternative scheduling models. The average difference amounts to mere milliseconds, suggesting a subtle distinction in performance.

5.5 Wait Time

Figure 7 visually presents the diverse spectrum of wait times associated with different scheduling models in normalized system models, in which the x-axis is the number of the tasks and the y-axis represents normalized wait time when executing the tasks with presented approaches.

Through an analysis of the graph, we can discern a notable surge in wait time at the pinnacle of the bar chart (highlighted by the orange color, indicating two tasks) when contrasted with our suggested approach (represented by the first column of the chart in green). This observation underscores the crucial role of the scheduling model selection in influencing the average wait time encountered within the system models. The CAPL-MEC model stands out as a standout scheduling approach, consistently achieving minimal wait times in diverse scenarios. Specifically, columns 2 and 3 (representing Levy walking on mobile and Levy walking partly on mobile and edge, respectively) demonstrate lower wait times when compared to the proposed method. However, it is essential to recognize that these columns also entail higher power consumption than the proposed method. Thus, it becomes imperative to acknowledge the inherent trade-offs among power consumption, wait time, and completion time, which the CAPL-MEC model effectively addresses.

5.6 CPU and Memory Utilization

Figures 8 and 9 visually compare the CPU and memory utilization across various scheduling models within the system models. The x-axis displays the number of

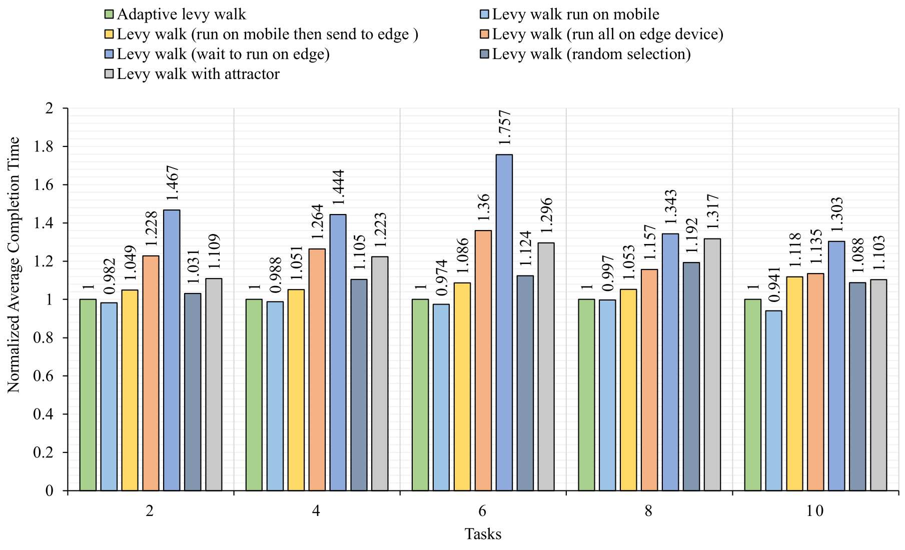

Fig. 6 Average time to complete each task, normalized for comparison. A lower value indicates better performance

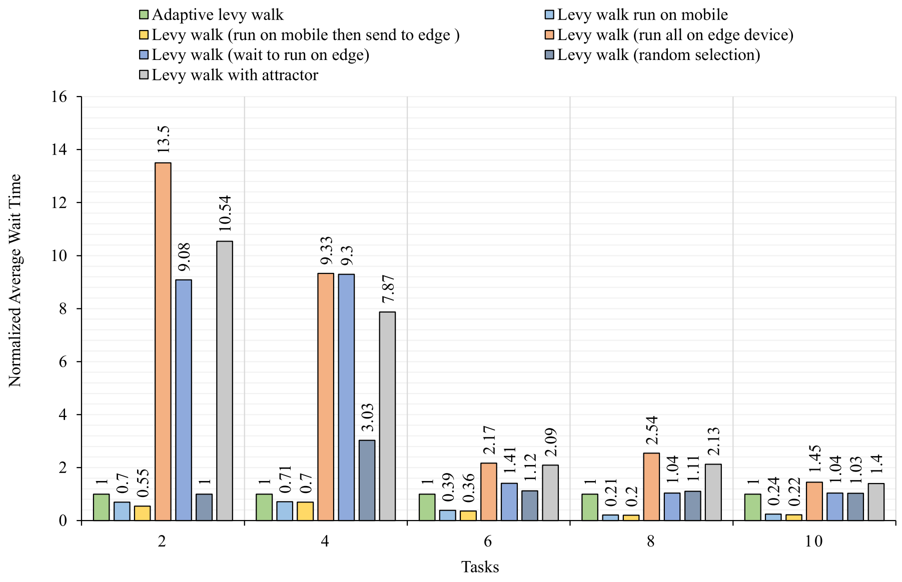

Fig. 7 Normalized average wait time per task; Lower values indicate better performance

tasks, while the y-axis indicates the average resource usage when executing tasks with different approaches.

The graphs reveal that two tasks result in lower CPU utilization, as represented by the lighter blue in the bar chart. This contrasts our proposed approach depicted in the first green column. Notably, this difference underscores the significance of scheduling model selection on the average CPU utilization experienced within the systems. Specifically, the CAPL-MEC model stood out as an energy-efficient scheduling solution by consistently obtaining minimal CPU usage across diverse scenarios, as demonstrated in the graphs. The visualizations effectively demonstrate the impact of scheduling decisions on resource utilization performance.

Figure 9 shows that our proposed approach has lower memory utilization because it uses resources effectively. Columns 2 and 3, representing Levy walk on mobile and Levy walk partially on mobile and edge, respectively, demonstrate higher overall memory utilization due to the memory constraints of the mobile

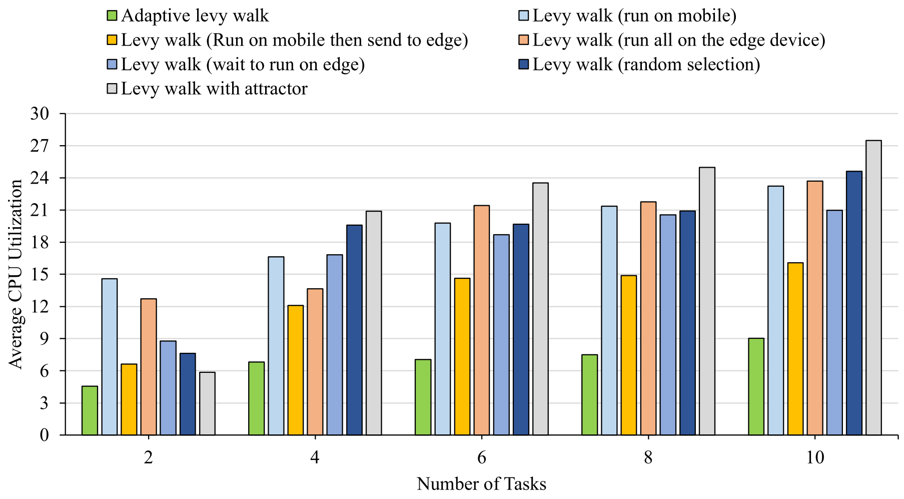

Fig. 8 Average utilization of CPU for each task, where a lower value is preferred for its better energy efficiency

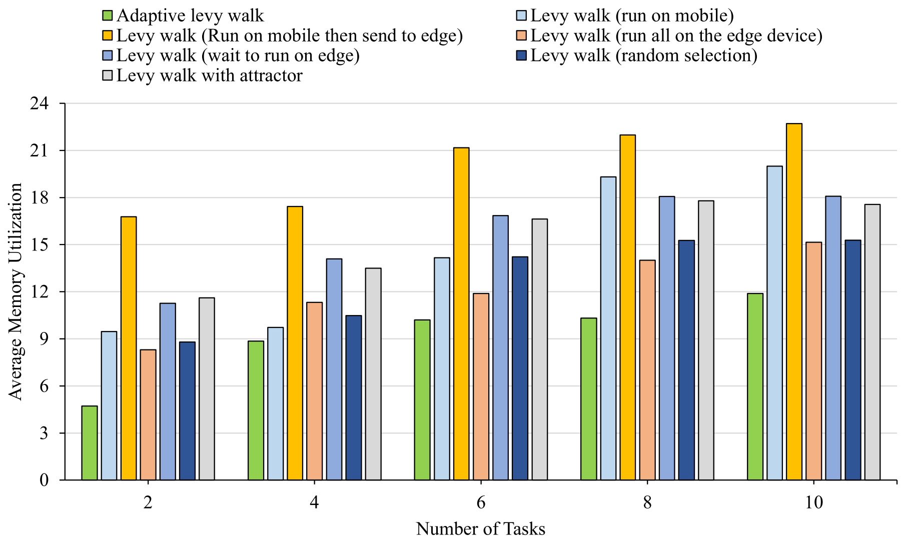

Fig. 9 Average memory utilization per task; lower values are preferable for enhanced energy efficiency

device (MD). The memory and CPU utilization results indicate that our proposed method achieves a good balance of resource usage.

6 Discussion

6.1 Impact of Varying Device and Node Counts

To evaluate the impact of varying the number of mobile devices and edge nodes, additional simulations were conducted where these parameters were modified. Table 5 reports the average completion time and energy consumption collected from these experiments. As seen from the table, increasing both the number of mobile devices and edge nodes generally leads to higher completion times and energy usage. This can be attributed to the additional resources and processing required to service more devices and schedule tasks across more nodes.

Notably, the 50 device / 30 node configuration results in the longest times and highest energy consumption, validating the effects of scaling. These results emphasize how workload distribution influences system performance metrics in MEC environments. The proposed method aims to optimize this distribution for improved efficiency.

6.2 Sensitivity to Workload

To analyze the impact of varying workloads, simulations were run with different numbers of tasks ranging from 10 to 50. Table 6 reports the corresponding resource utilization statistics collected.

As seen from the table, average CPU utilization predictably increases with higher workloads, rising from 8% for 10 tasks to 31% for 50 tasks. This is because more tasks impose additional processing

Table 5 Average completion time and energy consumption for varying number of mobile devices and edge nodes

No. of MDs	No. of edge nodes	Avg. completion time	Avg. energy consumption
10	10	120 ms	45 mJ
30	20	145 ms	61 mJ
50	30	168 ms	83 mJ

Table 6 Resource utilization statistics for varying workload, std: standard deviation

Workload	Average CPU util.	CPU util. std	Maximum CPU util.	Average memory util.	Memory util. std	Maximum memory util.
10 tasks	9%	1%	14%	11%	1%	16%
20 tasks	15%	2%	20%	18%	1%	20%
50 tasks	31%	4%	37%	29%	2%	35%

demands on the system resources. Interestingly, the standard deviation of CPU utilization also rises with workloads, reaching a maximum of 31% for 50 tasks. This indicates increasing variability in CPU usage across devices/nodes as tasks are scheduled in a more distributed manner under heavy loads. Average memory utilization displays a similar increasing trend, scaling from 7% to 15% with larger task counts. However, maximum memory usage exhibits less fluctuation with workloads, demonstrating that memory demands can be accommodated more efficiently than CPU demands.

Overall, these results emphasize how resource consumption patterns are influenced by workload distribution strategies. The proposed method minimizes variations through optimized scheduling tailored to device capabilities and task requirements.

7 Conclusion

This paper introduces CAPL-MEC, an adaptive system for scheduling tasks more efficiently in MEC environments. CAPL-MEC uniquely uses Levy walk modeling to predict how devices move around and schedule tasks accordingly. We tested CAPL-MEC extensively in simulations across important metrics like completion time, energy used, wait times, and reliability. The results showed that CAPL-MEC performs better than other systems by usually finishing tasks faster and optimizing resource allocation based on predicted device locations. Our approach also reduces the total energy used by including an energy consumption model and prioritizing decisions that save energy. The hybrid task allocation strategy also improves reliability by replicating tasks while minimizing extra processing. The proposed system is an effective solution for better scheduling tasks in MEC. We plan to implement CAPL-MEC in a real-world test environment to evaluate its performance for future work. Other optimization techniques, like deep learning, could also be explored. Expanding the system model to consider dependent tasks and workflows would enhance practical use. Integrating bandwidth optimization and offloading models is another area of interest.

Author contributions Abolfazl Younesi conceived the study, performed the experiments, analyzed the data, prepared all figures and tables, wrote the main manuscript text, and reviewed drafts of the paper. Mohammad Amin Fazli and Alireza Ejlali reviewed and critiqued the manuscript.

Data Availability Statement No datasets were generated or analysed during the current study.

Declarations

Competing Interests The authors declare no competing interests.

References

1. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: The communication perspective. IEEE Commun. Surv. Tutor. **19**(4), 2322–2358 (2017)
2. Ito, Y., Koga, H.: Improving offload delay using flow splitting and aggregation in edge computing. IEICE Commun. Express **8**(12), 468–473 (2019)
3. Pei, Y., Peng, Z., Wang, Z., Wang, H.: Energy-efficient mobile edge computing: three-tier computing under heterogeneous networks. Wireless Commun. Mob. Comput. **2020**, 1–17 (2020)
4. Azure, M.: Azure Stack Edge. https://azure.microsoft.com/en-us/products/azure-stack/edge
5. Amazon Web Services, I.: 5G Edge Computing Infrastructure – AWS Wavelength. https://aws.amazon.com/wavelength/
6. Rhee, I., Shin, M., Hong, S., Lee, K., Kim, S.J., Chong, S.: On the levy-walk nature of human mobility. IEEE/ACM Trans. Netw. **19**(3), 630–643 (2011)
7. Rhee, I., Shin, M., Hong, S., Lee, K., Chong, S.: On the levy-walk nature of human mobility. In: IEEE INFOCOM 2008-The 27th Conference on Computer Communications, pp. 924–932. IEEE (2008)
8. Lu, H., Gu, C., Luo, F., Ding, W., Zheng, S., Shen, Y.: Optimization of task offloading strategy for mobile edge com-

puting based on multi-agent deep reinforcement learning. IEEE Access **8**, 202573–202584 (2020)

9. Yindong, S., Liwen, P., Jingpeng, L.: An improved estimation of distribution algorithm for multi-compartment electric vehicle routing problem. J. Syst. Eng. Electron. **32**(2), 365–379 (2021)
10. Li, Z., Cao, Y., Dai, L.V., Yang, X., Nguyen, T.T.: Finding solutions for optimal reactive power dispatch problem by a novel improved antlion optimization algorithm. Energies **12**(15), 2968 (2019)
11. Wongkhuenkaew, R., Auephanwiriyakul, S., Theera-Umpon, N., Teeyapan, K., Yeesarapat, U.: Fuzzy k-nearest neighbor based dental fluorosis classification using multi-prototype unsupervised possibilistic fuzzy clustering via cuckoo search algorithm. Int. J. Environ. Res. Publ. Health **20**(4), 3394 (2023)
12. Budhiraja, I., Kumar, N., Tyagi, S., Tanwar, S., Han, Z., Piran, M.J., Suh, D.Y.: A systematic review on noma variants for 5g and beyond. IEEE Access **9**, 85573–85644 (2021)
13. Dang, T.N., Manzoor, A., Tun, Y.K., Kazmi, S.A., Haw, R., Hong, S.H., Han, Z., Hong, C.S.: Joint communication, computation, and control for computational task offloading in vehicle-assisted multi-access edge computing. IEEE Access **10**, 122513–122529 (2022)
14. El-Sayed, H., Chaqfeh, M.: Exploiting mobile edge computing for enhancing vehicular applications in smart cities. Sensors **19**(5), 1073 (2019)
15. Lin, L., Zhang, L.: Joint optimization of offloading and resource allocation for sdn-enabled iov. Wirel. Commun. Mob Comput. **2022** (2022)
16. Hasanin, T., Alsobhi, A., Khadidos, A., Qahmash, A., Khadidos, A., Ogunmola, G.A.: Efficient multiuser computation for mobile-edge computing in iot application using optimization algorithm. Appl. Bion. Biomech. **2021**, 1–12 (2021)
17. Bennis, M., Debbah, M., Poor, H.V.: Ultrareliable and low-latency wireless communication: Tail, risk, and scale. Proc. IEEE **106**(10), 1834–1853 (2018)
18. Adoga, H.U., Pezaros, D.P.: Network function virtualization and service function chaining frameworks: A comprehensive review of requirements, objectives, implementations, and open research challenges. Future Internet **14**(2), 59 (2022)
19. Zhang, K., Mao, Y., Leng, S., Maharjan, S., Zhang, Y.: Optimal delay constrained offloading for vehicular edge computing networks. In: 2017 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2017)
20. Zhan, C., Hu, H., Sui, X., Liu, Z., Niyato, D.: Completion time and energy optimization in the uav-enabled mobile-edge computing system. IEEE Internet Things J. **7**(8), 7808–7822 (2020)
21. Xu, Y., Zhang, T., Loo, J., Yang, D., Xiao, L.: Completion time minimization for uav-assisted mobile-edge computing systems. IEEE Trans. Veh. Technol. **70**(11), 12253–12259 (2021)
22. Li, B., Niu, L., Huang, X., Wu, H., Pei, Y.: Minimum completion time offloading algorithm for mobile edge computing. In: 2018 IEEE 4th International Conference on Computer and Communications (ICCC), pp. 1929–1933. IEEE (2018)
23. Naderializadeh, N., Hashemi, M.: Energy-aware multi-server mobile edge computing: A deep reinforcement learning approach. In: 2019 53rd Asilomar Conference on Signals, Systems, and Computers, pp. 383–387. IEEE (2019)
24. Yang, Z., Pan, C., Hou, J., Shikh-Bahaei, M.: Efficient resource allocation for mobile-edge computing networks with noma: Completion time and energy minimization. IEEE Trans. Commun. **67**(11), 7771–7784 (2019)
25. Liu, J., Mao, Y., Zhang, J., Letaief, K.B.: Delay-optimal computation task scheduling for mobile-edge computing systems. In: 2016 IEEE International Symposium on Information Theory (ISIT), pp. 1451–1455. IEEE (2016)
26. Guo, S., Liu, J., Yang, Y., Xiao, B., Li, Z.: Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing. IEEE Trans. Mob. Comput. **18**(2), 319–333 (2018)
27. Kuang, F., Xu, Z., Masdari, M.: Multi-workflow scheduling and resource provisioning in mobile edge computing using opposition-based marine-predator algorithm. Pervasive Mob. Comput. **87**, 101715 (2022)
28. Al-Habob, A.A., Dobre, O.A., Armada, A.G., Muhaidat, S.: Task scheduling for mobile edge computing using genetic algorithm and conflict graphs. IEEE Trans. Veh. Technol. **69**(8), 8805–8819 (2020)
29. Mao, Y., Zhang, J., Song, S., Letaief, K.B.: Power-delay tradeoff in multi-user mobile-edge computing systems. In: 2016 IEEE Global Communications Conference (GLOBECOM), pp. 1–6. IEEE (2016)
30. Cao, K., Xu, G., Zhou, J., Wei, T., Chen, M., Hu, S.: Qos-adaptive approximate real-time computation for mobility-aware iot lifetime optimization. IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst. **38**(10), 1799–1810 (2019). https://doi.org/10.1109/TCAD.2018.2873239
31. Li, J., Shang, Y., Qin, M., Yang, Q., Cheng, N., Gao, W., Kwak, K.S.: Multiobjective oriented task scheduling in heterogeneous mobile edge computing networks. IEEE Trans. Veh. Technol. **71**(8), 8955–8966 (2022). https://doi.org/10.1109/TVT.2022.3174906
32. Zheng, C., Pan, K., Dong, J., Chen, L., Guo, Q., Wu, S., Luo, H., Zhang, X.: Multi-agent collaborative optimization of uav trajectory and latency-aware dag task offloading in uav-assisted mec. IEEE Access **12**, 42521–42534 (2024). https://doi.org/10.1109/ACCESS.2024.3378512
33. Chen, X., Jiao, L., Li, W., Fu, X.: Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. Netw. **24**(5), 2795–2808 (2015)
34. Jiang, F., Dong, L., Wang, K., Yang, K., Pan, C.: Distributed resource scheduling for large-scale mec systems: A multiagent ensemble deep reinforcement learning with imitation acceleration. IEEE Internet Things J. **9**(9), 6597–6610 (2022). https://doi.org/10.1109/JIOT.2021.3113872
35. Cao, K., Li, L., Cui, Y., Wei, T., Hu, S.: Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing. IEEE Trans. Ind. Inf. **17**(1), 494–503 (2021). https://doi.org/10.1109/TII.2020.2975897
36. Maleki, E.F., Mashayekhy, L., Nabavinejad, S.M.: Mobility-aware computation offloading in edge computing using machine learning. IEEE Trans. Mob. Comput. **22**(1), 328–340 (2023). https://doi.org/10.1109/TMC.2021.3085527
37. Saleem, U., Liu, Y., Jangsher, S., Li, Y., Jiang, T.: Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing. IEEE Trans. Wirel. Com-

mun. **20**(1), 360–374 (2021). https://doi.org/10.1109/TWC.2020.3024538
38. Cao, K., Weng, J., Li, K.: Reliability-driven end–end–edge collaboration for energy minimization in large-scale cyber-physical systems. IEEE Trans. Reliab. **73**(1), 230–244 (2024). https://doi.org/10.1109/TR.2023.3297124
39. Wang, Y., Niu, J., Chen, G., Zhou, X., Li, Y., Liu, S.: Ris-aided latency-efficient mec hetnet with wireless backhaul. IEEE Trans. Veh. Technol. 1–15 (2024). https://doi.org/10.1109/TVT.2024.3354371
40. Ramos-Fernández, G., Mateos, J.L., Miramontes, O., Cocho, G., Larralde, H., Ayala-Orozco, B.: Lévy walk patterns in the foraging movements of spider monkeys (ateles geoffroyi). Behav. Ecol. Sociobiol. **55**, 223–230 (2004)
41. Gautestad, A.O., Mysterud, A.: The lévy flight foraging hypothesis: forgetting about memory may lead to false verification of brownian motion. Mov. Ecol. **1**, 1–18 (2013)
42. Yeganeh-Khaksar, A., Ansari, M., Ejlali, A.: Remap: Reliability management of peak-power-aware real-time embedded systems through task replication. IEEE Trans. Emerg. Top. Comput. **10**(1), 312–323 (2020)
43. Ansari, M., Saber-Latibari, J., Pasandideh, M., Ejlali, A.: Simultaneous management of peak-power and reliability in heterogeneous multicore embedded systems. IEEE Trans. Parallel Distrib. Syst. **31**(3), 623–633 (2019)
44. Mao, Y., Zhang, J., Letaief, K.B.: Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J. Sel. Areas Commun. **34**(12), 3590–3605 (2016)
45. You, C., Huang, K., Chae, H., Kim, B.-H.: Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Trans. Wirel. Commun. **16**(3), 1397–1411 (2016)