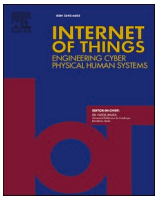
Internet of Things 31 (2025) 101587

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.sciencedirect.com/journal/internet-of-things



Self-learning adaptive power management scheme for energy-efficient IoT-MEC systems using soft actor-critic algorithm

Amir Masoud Rahmani [a,1], Amir Haider [b,1], Komeil Moghaddasi [c], Farhad Soleimanian Gharehchopogh [c], Khursheed Aurangzeb [d], Zhe Liu [e,f], Mehdi Hosseinzadeh [g,h,*]

[a] Future Technology Research Center, National Yunlin University of Science and Technology, Yunlin, Taiwan
[b] Department of AI and Robotics, Sejong University, Seoul, 05006, Republic of Korea
[c] Department of Computer Engineering, Urmia Branch, Islamic Azad University, Urmia, Iran
[d] Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, P. O. Box 51178, Riyadh, 11543, Saudi Arabia
[e] College of Mathematics and Computer, Xinyu University, Xinyu, 338004, China
[f] School of Computer Sciences, Universiti Sains Malaysia, Penang 11800, Malaysia
[g] School of Computer Science, Duy Tan University, Da Nang, Vietnam
[h] Jadara Research Center, Jadara University, Irbid 21110, Jordan

ARTICLE INFO

Keywords:
Energy management
Internet of things
Soft actor-critic algorithm
Context-aware optimization

ABSTRACT

The rapid increase of Internet of Things (IoT) devices in Mobile Edge Computing (MEC) environments requires effective energy management to ensure device operation and enhance network efficiency. IoT-MEC systems face challenges such as varying task loads, dynamic environmental conditions, and limited energy resources. These factors make it challenging to design adaptive and efficient energy strategies. Traditional methods, such as static scheduling and centralized control strategies, struggle to adapt to real-time fluctuations in task loads and network conditions, resulting in inefficient energy use, higher latency, and a lack of flexibility to respond to these demands in real-time. This paper proposes a self-learning power management model using the Soft Actor-Critic (SAC) algorithm. It is deployed on IoT devices to enable localized and context-aware power management. Our model includes modules for energy monitoring, adaptive task prioritization, and a self-adjusting reinforcement-learning mechanism, which dynamically fine-tunes energy policies based on real-time device conditions, allowing each device to optimize power use independently without heavy dependence on centralized control. MEC nodes gather data on battery health, load, and network conditions to support decentralized policy adjustments. Connected devices in simulated smart homes served as the primary context for evaluation. Experimental results show that our model achieves a 45 % reduction in energy consumption in smart home environments, a 49 % improvement in battery life (compared to baseline-like Local Computing), and high adaptability in diverse scenarios compared with other methods.

* Corresponding author.
E-mail addresses: rahmania@yuntech.edu.tw (A.M. Rahmani), amirhaider@sejong.ac.kr (A. Haider), k.moghaddasi@ieee.org (K. Moghaddasi), bonab.farhad@gmail.com (F.S. Gharehchopogh), kaurangzeb@ksu.edu.sa (K. Aurangzeb), liuzhe921@gmail.com (Z. Liu), mehdihosseinzadeh@duytan.edu.vn (M. Hosseinzadeh).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.iot.2025.101587

Available online 21 March 2025
2542-6605/© 2025 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

1. Introduction

IoT devices' rapid rise and use in MEC systems has revolutionized many sectors, from healthcare and smart cities to industrial automation [1]. These devices are designed to collect, process, and send data in real-time and often depend on MEC frameworks to offload data processing, which improves response times and reduces latency [2]. However, energy efficiency is a critical challenge in IoT-MEC ecosystems [3,4]. Most IoT devices rely on limited battery power. Many are deployed in environments where recharging is impractical, such as remote monitoring systems or smart agriculture [5,6]. Balancing device functionality while ensuring long battery life is essential yet challenging, particularly as these devices must adjust to fluctuating network conditions, diverse data loads, and external factors such as temperature variations or signal disruptions. For example, an intelligent surveillance camera and environmental sensors in pollution monitoring may have high data processing needs during peak hours. In such cases, efficient energy management is crucial to prevent rapid battery depletion. Therefore, innovative and adaptive energy management solutions are essential to ensure these devices maintain performance while conserving battery life under diverse and dynamic conditions [7–9]. Our study focuses on smart home IoT devices. We propose the SAC algorithm to optimize energy usage through localized decision-making. This approach ensures sustainable operation even during high-demand periods.

Traditional energy management strategies often struggle in these settings because they rely on fixed or reactive approaches, which lack the flexibility to adjust in real time to varying network loads and device usage [10,11]. Centralized methods are particularly problematic. They create bottlenecks and high latency, which impede timely energy adjustments required for sustained device functionality [12]. Moreover, these approaches struggle to scale efficiently as the rise of connected devices increases the computational load on central servers, possibly leading to higher energy consumption [13]. Reactive strategies, which respond only after detecting energy depletion, often result in delayed energy-saving actions, reducing battery life and impairing device performance in dynamic conditions. For MEC systems, which demand decentralized, scalable solutions, current methods lack the adaptability needed to optimize power use continuously across diverse devices and tasks [14].

Existing approaches for energy management in IoT-MEC systems often employ Reinforcement Learning (RL) and other adaptive methods to optimize power use [15–21]. Techniques like Q-learning and Deep Q-Networks (DQN) have shown promise in adapting to specific energy demands. However, they frequently struggle with real-time adaptability and scalability. For instance, they struggle with high-frequency state transitions, such as rapid shifts between idle and active modes, which is standard in IoT applications with fluctuating workloads. Additionally, many existing methods also rely heavily on centralized cloud resources. This reliance results in high communication costs and delays, which impede timely decision-making in energy-sensitive scenarios. Furthermore, the scalability of these solutions is limited by the computational load centralized systems impose, especially as the number of connected devices increases. As a result, existing models cannot often handle changing loads efficiently, falling short in adjusting energy management policies to account for varied device conditions and shifting network states. This leaves significant gaps in achieving optimal energy efficiency for dynamic IoT-MEC ecosystems. As illustrated in Fig. 1, the proposed architecture includes multiple MEC nodes, each connected to smart homes filled with various IoT devices. Each MEC node operates two phases: the SAC Algorithm in the Local Adaptation Phase for learning optimal energy management policies and the Energy Management Phase, which relies on localized data. This approach enables efficient, context-aware energy management for connected devices within smart homes.

This study develops a decentralized energy management model using the SAC algorithm at the heart of the system and tailors it to IoT devices within MEC systems. The main contributions of this study are as follows:

- We developed a self-learning SAC framework to enable IoT devices to autonomously optimize energy consumption in real time under dynamic operational conditions.
- We designed a decentralized, context-aware energy management architecture to improve adaptability, scalability, and real-time decision-making efficiency.

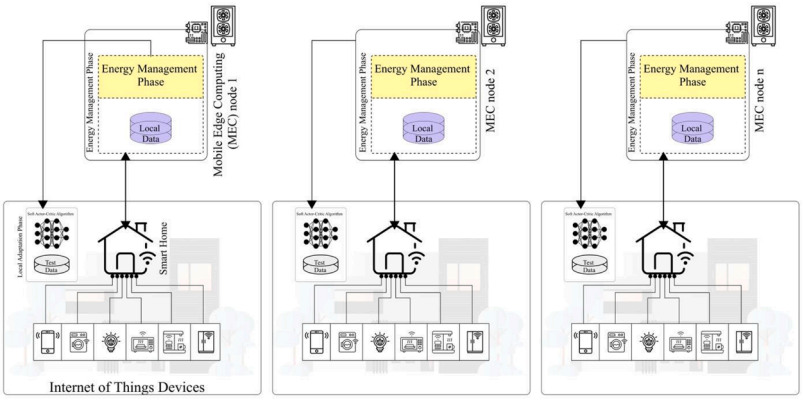


Fig. 1. Proposed system architecture.

- We created power management modules that dynamically adjust based on device load, battery status, and task priority, optimizing energy usage at the device level.

The proposed model can be applied in various IoT settings beyond smart homes. In smart agriculture, it could manage the energy usage of irrigation systems, weather sensors, and drones. These devices often work in remote areas with limited access to power. Efficient energy management ensures continuous operation and reduces maintenance costs. The model could optimize energy for streetlights, traffic sensors, and public surveillance systems in urban infrastructure. These devices experience fluctuating workloads based on traffic or time of day. Adapting in real time, the model can enhance performance, reduce energy waste, and ensure reliability in diverse, dynamic environments.

The rest of the paper is organized as follows: Section 2 presents related work, showing current approaches and highlighting the need for improved self-learning energy management in IoT-MEC systems. Section 3 explains our system model in detail. Section 4 covers the simulation setup and parameters, including network, device, and algorithm configurations. Section 5 presents the results and discussion, comparing our model with other RL methods to demonstrate its energy-saving capabilities and adaptability. Finally, Section 6 concludes the paper by summarizing key findings, acknowledging limitations, and suggesting future research directions.

2. Related works

This section reviews existing literature on energy management approaches for IoT devices with MEC. We examine various strategies, including centralized and decentralized methods, adaptive and RL-based techniques, and their applications in achieving energy efficiency. Moreover, we introduce related energy-saving schemes in different areas of study to provide context for the diverse approaches to energy optimization in IoT and MEC systems. Initially, Do et al. [19] proposed a deep deterministic policy gradient algorithm with prioritized experience replay (DDPG-PER) for resource allocation in Industrial IoT (IIoT) systems using MEC federation. This approach addresses the joint decision offloading and resource allocation challenges, aiming to minimize energy-delay costs in IIoT. They transformed the energy-delay optimization into a Markov Decision Process (MDP) by modeling real-world factors affecting system performance. The DDPG-PER method efficiently manages high-dimensional, continuous action and state spaces, showing effective energy-delay cost minimization through simulations. Ansere et al. [20] introduced an RL-based solution to optimize computation resources for MEC-aided IoT devices. They used a stochastic optimization approach with the Lyapunov technique to maximize long-term energy efficiency by managing transmission power, network stability, and latency. A double Deep Reinforcement Learning (DRL) approach was developed for real-time computation offloading, leveraging MDP to adapt to dynamic MEC environments. Their method minimizes completion time and energy consumption for stochastic tasks, achieving low computational complexity and enhanced energy efficiency compared to baseline policies. Xu et al. [22] presented a method for energy-efficient joint optimization in MEC-assisted IoT systems by allowing IoT devices to offload tasks to MEC servers. To tackle the complexity of optimizing sensing, communication, and computation resources amid increasing device interactions, they introduced a general mean field N-player Markov game (GMFG), reformulating it as an MFG with teams. They employed an MFG-based Actor-Critic algorithm (MFGAC), achieving reduced long-term system costs and improved performance over other schemes through simulations. Similarly, Xiao et al. [23] proposed a decentralized collaborative MADRL-based algorithm for energy-efficient resource allocation in heterogeneous MEC (Het-MEC) networks. The method employs a multi-actor shared-critic architecture and a regional training distributed execution framework. This approach addresses scalability and overhead issues associated with centralized schemes, aiming to stabilize model training and minimize information exchange. Through simulations, the algorithm demonstrated better performance over baseline methods.

Wu et al. [24] proposed a joint optimization model constrained by delay and energy consumption in the IIoT using queuing theory to manage task offloading in Edge computing-based IIoT environments. They developed a Multi-Agent DRL algorithm based on MAPPO, specifically adapted for EIIoT. By integrating queuing theory and optimizing neural network structures, MAQDRL dynamically adjusts offloading strategies, achieving optimal performance in multiuser scenarios. Chouikhi et al. [25] proposed a DRL-based computation offloading approach for IIoT systems to address latency, reliability, and energy consumption challenges. The model enables efficient task offloading to edge servers, maximizing task completion and reducing long-term energy use. With multiple agents, each IIoT device has its own DRL model trained in the cloud for real-time decision-making on the edge. Heidarpour et al. [26] presented DeepLM, a SAC-based DRL model, for network lifetime optimization in multiuser MEC-enabled IoT systems. This approach jointly optimizes task splitting, local CPU-cycle frequencies, bandwidth, and CPU-cycle frequency at the MEC server, addressing constraints like task queuing, bandwidth, and maximum CPU-cycle frequencies. DeepLM shows rapid convergence with minimal oscillations and extends the network lifetime. Lu et al. [27] introduced a resource allocation approach for cognitive radio-based MEC systems using non-orthogonal multiple access (NOMA) under time-varying channels. They introduced a generalized user grouping scheme, allowing secondary users (SUs) to participate in various NOMA groups and enabling partial offloading to multiple MEC servers. A decomposition-based soft actor-critic approach combines convex optimization with DRL to minimize long-term energy consumption, achieving up to 87.5 % energy savings compared to conventional user grouping schemes through optimized offloading ratios for SUs. Finally, Du et al. [21] proposed a solution for optimizing energy efficiency in UAV-based IoT data collection and offloading to MEC servers using access points equipped with powerful servers. This approach programs multiple UAV flight trajectories and efficiently allocates bandwidth to relay IoT data. Additionally, decentralized wireless charging stations are deployed to replenish UAV energy for continuous operations. A DRL method, specifically Deep Deterministic Policy Gradient (DDPG), addresses the complex programming challenge, enhancing UAV energy efficiency and data relayed and reducing data interaction energy consumption while ensuring geographic fairness. Wen et al. [28] proposed a three-tier satellite edge computing framework, to provide global computing

services for ground users. The framework incorporates a hybrid computing offloading architecture and reinforcement learning to minimize task latency, reduce energy consumption, and ensure task success rates. A multi-layer learning-based offloading algorithm enables higher-level agents to consult mist node agents. Similarly, Cheng et al. [29] introduced GeoScale, a method for microservice autoscaling in geo-distributed edge clouds to optimize average request response time under long-term cost constraints. Using the Lyapunov optimization framework, GeoScale decomposes the long-term problem into per-timeslot sub-problems, solved via a signomial geometric programming (SGP)-based algorithm. Huang et al. [30] presented a decentralized framework for MEC federation to optimize computation offloading, task migration, and resource allocation in IoT-enabled smart cities. Their approach minimizes latency, energy consumption, and load imbalance under migration cost constraints. They developed an MDs clustering matching algorithm and utilized Lyapunov optimization with a Transformer-based mobility prediction model. A DDPG-based framework addresses the problem efficiently. Zhou et al. [31] proposed an adaptive task offloading approach, ATO-SLA, for satellite edge computing to optimize user-perceived delay and energy consumption. Addressing workload imbalances caused by spatiotemporal variations on LEO satellites, the study modeled a spatiotemporal load-aware task offloading problem using the MDP. They developed a proximal policy optimization-based algorithm to solve the issue adaptively. Table 1 provides a comparative overview of the main contributions, strengths, and limitations of the related approaches and our study.

Unlike Do et al., our work differentiates itself from existing energy management research for IoT devices in MEC systems by focusing on a decentralized, adaptive SAC-based model explicitly tailored for real-time power optimization. Unlike Do et al. [19], who rely on a DDPG-PER approach for resource allocation in IIoT, our model is designed to perform continuous, localized energy management without requiring central federation, reducing the need for extensive communication. While Ansere et al. [20] and Xu et al. [22] leverage DRL-based offloading to enhance energy efficiency. Our model focuses on decentralized decision-making at the device level, which minimizes dependency on MEC servers. Existing works, such as Xiao et al. [23] and Wu et al. [24], emphasize collaborative learning and queuing theory in multi-agent systems but often require extensive overhead for data exchange. Our approach, by contrast, minimizes this overhead through context-aware, device-level decision-making while requiring only minimal MEC node interaction. Moreover, we integrate specific system metrics not addressed in prior studies, such as task prioritization, adaptive transition management between active and idle states, and real-time power adjustment through local SAC agents. The following section introduces our system model, detailing the architecture and mechanisms for optimized energy management in IoT-MEC environments.

3. System model

In this section, we introduce the system model and its architecture. The primary goal of this model is to optimize energy use on IoT

Table 1
Properties of the related studies and this paper.

Study	Focus Area	Key Strength	Limitation
Do et al. [19]	Resource allocation in IIoT using DDPG-PER	Efficient handling of high-dimensional spaces, minimizing energy-delay costs	Relies on central federation; lacks localized decision-making
Ansere et al. [20]	RL-based computation resource optimization for MEC-aided IoT	Stochastic optimization with Lyapunov technique; low computational complexity	Focuses on offloading; limited adaptability to dynamic operational conditions
Xu et al. [22]	Energy-efficient joint optimization in MEC-assisted IoT	Uses MFGAC for reduced long-term costs; scalability in multi-device settings	Limited focus on task prioritization and active-idle state management
Xiao et al. [23]	Collaborative MADRL for resource allocation in Het-MEC networks	Addresses scalability and overhead issues in collaborative environments	High communication overhead; lacks device-specific adaptability
Wu et al. [24]	Joint delay and energy optimization in IIoT using MAPPO	Dynamic adjustment of offloading strategies with queuing theory integration	Overhead due to queuing theory; limited focus on minimizing MEC-server interactions
Chouikhi et al. [25]	DRL-based computation offloading for IIoT systems	Multi-agent DRL for real-time task offloading in IIoT systems	Relies on cloud training; lacks efficient real-time adaptability at the edge
Heidarpour et al. [26]	SAC-based network lifetime optimization in multiuser MEC systems	Rapid convergence, minimal oscillations; extended network lifetime	Focused on network lifetime; does not address task-specific and priority-driven optimizations
Lu et al. [27]	Resource allocation in CR-MEC with NOMA using SAC	Achieves 87.5 % energy savings via optimized offloading ratios	Assumes fixed user grouping schemes; lacks dynamic adaptability to changing network conditions
Du et al. [21]	Energy efficiency in UAV-enabled IoT data collection using DDPG	Enhanced UAV energy efficiency; decentralized wireless charging	Device-specific energy assumptions; lacks real-time task prioritization
Wen et al. [28]	Three-tier satellite edge computing framework	Hybrid offloading with reinforcement learning; high task success rates	Limited adaptability to localized energy management and task prioritization
Cheng et al. [29]	Geo-distributed edge cloud microservice autoscaling	Optimizes response time and long-term cost with Lyapunov optimization	Relies heavily on global optimization; lacks localized, device-level adaptability
Huang et al. [30]	MEC federation for IoT in smart cities	Decentralized optimization; combines Lyapunov and DDPG; minimizes latency and energy	Computational overhead; limited focus on active-idle state transitions for energy efficiency
Zhou et al. [31]	Adaptive task offloading in satellite edge computing	Spatiotemporal load-aware optimization using PPO	Limited integration of real-time energy management and task priority considerations
This Paper	Decentralized SAC-based adaptive energy management for IoT-MEC	Localized decision-making, adaptive task prioritization, minimal overhead	Limited focus on real-world testing and large-scale deployment

devices. The system emphasizes localized energy-saving strategies, where each device manages its power usage independently, based on real-time environmental contexts. Each IoT device is equipped with components for energy management that monitor battery levels, process loads, and network conditions, enabling context-aware decision-making. With this setup, devices can intelligently allocate energy resources, dynamically switching between active and idle states to reduce unnecessary power usage. The MEC layer plays an important, non-intrusive role by providing minimal coordination without centralized cloud-based resources. MEC nodes gather contextual information from nearby IoT devices, such as network load and device battery health. This allows for local data aggregation and periodic updates to distributed policies. A self-learning mechanism powered by the SAC algorithm is deployed on each IoT device to optimize power management choices adaptively. SAC's RL framework forms a strong foundation for continuous learning and adaptability, as it handles continuous action spaces required for precise power adjustments. With an energy-aware reward function, each device learns to minimize power use while meeting task requirements, supporting sustainable operation over extended periods.

3.1. Network and device model

In this system model, IoT devices have limited energy and depend on adaptive power management to function efficiently in different settings. Each device executes only basic computations locally, focusing on using energy wisely while preserving essential operations.

The total energy consumption $E_i(t)$ of an IoT device i over time t is defined by summing the energy used in active and idle states as given by Eq. (1):

$$E_i(t) = \sum_{j \in \{\text{active, idle}\}} P_j \cdot T_j \tag{1}$$

where P_j is the power consumption in state j, and T_j is the time duration spent in that state.

The battery level $B_i(t)$ at any time, t is calculated by subtracting the energy consumed from the initial battery capacity B_{init} as modeled as Eq. (2):

$$B_i(t) = B_{\text{init}} - \sum_{k=1}^{t} E_i(k) \tag{2}$$

IoT devices can dynamically switch between active and idle states based on task priority α and remaining battery level $B_i(t)$. The probability P_{active} of remaining active is given by Eq. (3):

$$P_{\text{active}} = \frac{\alpha \cdot B_i(t)}{\alpha \cdot B_i(t) + B_{\max} - B_i(t)} \tag{3}$$

Define the energy efficiency η_i of device i as the ratio of successful task completion to energy used. This is represented as Eq. (4).

$$\eta_i = \frac{\sum_{n=1}^{N_{\text{tasks}}} O_{i,n}}{\sum_{k=1}^{t} E_i(k)} \tag{4}$$

where $O_{i,n}$ represents the output or performance metric of task n completed by device i, and N_{tasks} is the total number of tasks over the period t.

MEC nodes act as local coordinators with limited capabilities. They assist IoT devices by gathering data and updating energy management policies according to local conditions. These nodes are essential for maintaining network stability and energy efficiency without relying on centralized computation.

MEC nodes aggregate information from nearby IoT devices, such as battery levels B_i and network conditions C_i, to construct a detailed understanding of the network. The aggregated data $A(t)$ at time t is computed as Eq. (5).

$$A(t) = \sum_{i=1}^{N} \frac{B_i(t) \cdot C_i(t)}{N} \tag{5}$$

where N represents the total number of devices connected to the MEC node.

MEC nodes periodically send updated energy management policies $\pi_i(t)$ to devices based on aggregate data. Policy updates are defined as Eq. (6).

$$\pi_i(t) = \pi_i(t-1) + \sum_{j=1}^{M} \delta\pi_j(t) \tag{6}$$

where $\delta\pi_j(t)$ represents minor updates for M distinct conditions (e.g., battery level, network load), optimizing the energy allocation for each device.

To evaluate and adjust the energy management policy, MEC nodes compute a utility function $U(t)$ based on the average energy efficiency across all devices. This represented as Eq. (7).

$$U(t) = \frac{\sum_{i=1}^{N} \eta_i(t) \cdot P_{\text{active}}(t)}{N} \tag{7}$$

MEC nodes also balance the network load, controlling the frequency of policy updates based on overall system status. The coordination load L_{coord} is expressed as Eq. (8).

$$L_{\text{coord}} = \min\left(L_{\max}, \frac{\sum_{j=1}^{M} A(t)_j}{T_{\text{policy}}}\right) \tag{8}$$

where $L_{\max}$ is the maximum allowed load, and T_{policy} is the interval for policy updates. The communication model between IoT devices and MEC nodes is designed to be minimalistic, allowing devices to share only essential information. This approach conserves energy by reducing unnecessary data transmission, aligning with the goal of energy efficiency. The energy required for communication, E_{comm}, is calculated as the total energy used for transmission and reception, as shown in Eq. (9).

$$E_{\text{comm}} = \sum_{j \in \{\text{transmit, receive}\}} P_j \cdot T_j \tag{9}$$

where P_j and T_j are the power and time for transmission and reception.

To minimize communication overhead, the size of data packets S_{data} is adjusted based on network status as modeled as Eq. (10).

$$S_{\text{data}} = S_{\text{base}} + \sum_{j=1}^{M} \delta S_j \tag{10}$$

Where S_{base} is the minimal required data, and δS_j represents additional data, based on network conditions. To avoid network congestion, each device calculates a probability P_{transmit} of sending a packet, as described in Eq. (11), based on the network load $L(t)$.

$$P_{\text{transmit}} = \frac{1}{1 + e^{-k \cdot \left(\sum_{i=1}^{N} L_{\max} - L_i(t)\right)}} \tag{11}$$

where k is a scaling parameter, and $L_{\max}$ represents the peak load limit.

The overall network efficiency ρ, balancing throughput and energy use, is calculated as Eq. (12).

$$\rho = \frac{\sum_{i=1}^{N} S_{\text{data},i}}{\sum_{i=1}^{N} E_{\text{comm},i}} \tag{12}$$

Communication delay D for each device is impacted by the number of transmissions N_{transmit} and transmission time T_{transmit}. This is modeled as Eq. (13).

$$D = \sum_{i=1}^{N_{\text{transmit}}} T_{\text{transmit},i} \tag{13}$$

3.2. Power management and optimization modules

Efficient power management and optimization are essential for achieving energy efficiency and extending the operation of IoT devices within MEC-enabled networks. These modules monitor, allocate, and optimize real-time power usage, adapting to device status and contextual needs. This section discusses the Energy Management Module, Battery Health Monitor, and Power State Optimization Module, each with specific formulations that define their functionalities.

3.2.1. Energy management module

The Energy Management Module (EMM) continuously monitors and adjusts each IoT device's energy use based on its operational needs and context. The module dynamically allocates energy resources to balance energy consumption with task performance. The energy allocation $E_{\text{alloc},i}(t)$ for device i at time t is adjusted based on a weighted function of task priority α_i, network conditions $C_i(t)$, and its remaining battery $B_i(t)$. This is represented as shown in Eq. (14).

$$E_{\text{alloc},i}(t) = \alpha_i \cdot \frac{C_i(t)}{\sum_{j=1}^{N} C_j(t)} + (1 - \alpha_i) \cdot \frac{B_i(t)}{B_{\max}} \tag{14}$$

For each task k, which requires energy $E_{i,k}$ the module ensures sufficient allocation from available energy $E_{\text{avail},i}(t)$ by adjusting based on the device's priority and context. This is defined as Eq. (15).

$$E_{i,k} = \min\left(E_{\text{avail},i}(t), \frac{W_k}{\sum_{l=1}^{L} W_l} \cdot E_{\text{alloc},i}(t)\right) \tag{15}$$

where W_k is the weight of task k based on priority.

Define the consumption efficiency $\epsilon_i(t)$ as the ratio of successfully allocated energy to total available energy, indicating practical usage. This is presented as Eq. (16).

$$\epsilon_i(t) = \frac{\sum_{k=1}^{K} E_{i,k}}{E_{\text{avail},i}(t)} \tag{16}$$

Fig. 2 illustrates how the EMM monitors and manages the energy usage of IoT devices. It adjusts energy consumption based on task priority, network conditions, and remaining battery life. The energy allocated to each task is optimized for efficient usage, with real-time adjustments made through the Monitor and Adjust feedback loop.

3.2.2. Battery health monitor

The Battery Health Monitor (BHM) component tracks each IoT device's battery status, considering energy depletion rates and environmental impacts like temperature fluctuations or usage intensity. This module supports the Energy Management Module by providing real-time battery health data to guide energy allocation and preservation strategies. The depletion rate $D_i(t)$ at time t is calculated using Eq. (17), which measures the reduction in battery level over a specified period Δt:

$$D_i(t) = \frac{B_i(t) - B_i(t - \Delta t)}{\Delta t} \tag{17}$$

The Battery Health Index H_i reflects the overall health of the battery, calculated based on cumulative depletion and recharge cycles R_i over a period T as given by Eq. (18).

$$H_i = 1 - \frac{\sum_{k=1}^{R_i} D_i(k)}{T} \tag{18}$$

To maximize battery life, an optimal discharge rate $D_{\text{opt},i}$ is maintained. This is represented as Eq. (19).

$$D_{\text{opt},i} = \frac{B_{\text{init}} - B_{\text{min}}}{T} \tag{19}$$

where B_{init} and B_{min} are initial and minimum operational battery levels, respectively.

When battery levels approach a critical threshold B_{crit}, recharge scheduling S_{recharge} is initiated, optimizing the timing of recharge events. This is modeled as Eq. (20).

$$S_{\text{recharge}} = \left\{ t \middle| B_i(t) \leq B_{\text{crit}} \wedge t \bmod T_{\text{recharge}} = 0 \right\} \tag{20}$$

where T_{recharge} is the predefined recharge cycle.

The optimal discharge rate (Eq. 19) ensures smart home devices, such as thermostats or sensors, efficiently manage transitions between idle and active states. For example, during peak usage, the depletion rate (Eq. 17) quantifies the battery consumption within the interval Δt, and the Battery Health Index (Eq. 18) assesses long-term performance. When battery levels approach the critical threshold B_{crit}, recharge scheduling (Eq. 20) ensures timely recharging. These calculations dynamically inform energy allocation, maintaining device functionality during active periods while optimizing battery health and longevity.

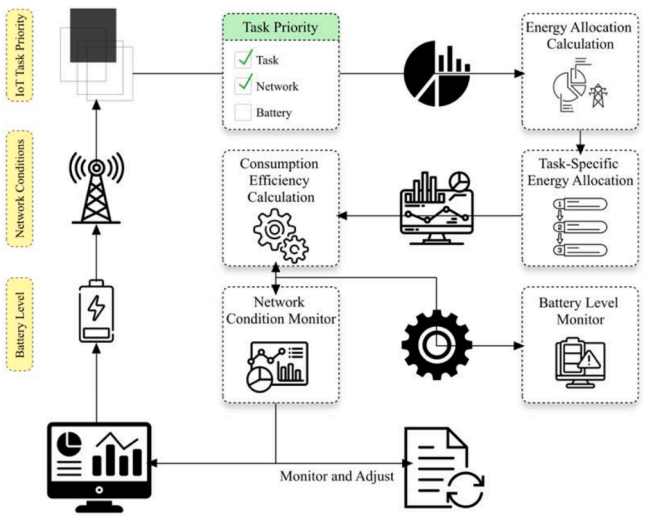


Fig. 2. The proposed energy management module.

3.2.3. Power state optimization module

The Power State Optimization Module (PSOM) controls the shift between idle and active states to minimize unnecessary power consumption. It evaluates task requirements, device context, and priority to select the most efficient state. This approach optimizes energy usage while ensuring task performance remains unaffected.

The probability $P_{\text{active}}(t)$ for a device i to remain in the active state at time t, it is derived based on the ratio of task importance I_k to energy efficiency $\eta_i(t)$ as Eq. (21).

$$P_{\text{active}}(t) = \frac{\sum_{k=1}^{K} I_k}{\eta_i(t)} \tag{21}$$

The optimal duration $T_{\text{opt},i}$ for remaining in a particular state is determined by the time it takes to reach maximum efficiency $\eta_{\max}$ as given by Eq. (22).

$$T_{\text{opt},i} = \arg\max_{t}(\eta_i(t) \cdot P_{\text{active}}(t)) \tag{22}$$

The power-saving factor ψ_i is defined as the relative difference in energy consumption between the idle and active states over time, calculated as Eq. (23).

$$\psi_i = \frac{P_{\text{idle}} \cdot T_{\text{idle}} - P_{\text{active}} \cdot T_{\text{active}}}{P_{\text{active}} \cdot T_{\text{active}}} \tag{23}$$

The delay D_{trans} in transitioning between states is impacted by task load L_k and available energy E_{avail}. This is given by Eq. (24).

$$D_{\text{trans}} = \frac{L_k}{E_{\text{avail}}} \cdot \sum_{j=1}^{J} P_j \tag{24}$$

where J denotes all active tasks influencing the delay.

To enhance power efficiency, the duration of idle time T_{idle} is dynamically adjusted based on the contextual variables such as environmental conditions $C_i(t)$. This is calculated as Eq. (25).

$$T_{\text{idle}} = T_{\text{base}} \cdot \left(1 - \frac{C_i(t)}{\sum_{j=1}^{N} C_j(t)}\right) \tag{25}$$

where T_{base} is the baseline idle duration.

As shown in Fig. 3, the module functions by constantly evaluating both the task requirements (e.g., throughput demands, deadlines) and the device context (e.g., priority level, current battery status, usage patterns) to determine how best to toggle or adjust each IoT device's power states in real-time. First, it computes active state probabilities—estimating the likelihood that a device will be needed during the next scheduling interval—using factors such as the frequency and duration of past usage and short-term forecasts of task arrival. Next, it calculates the power-saving factor, which quantifies the potential energy savings achievable by placing the device in a lower power state (like standby or deep sleep) while ensuring that any state transitions do not compromise task performance. At the same time, the module identifies optimal durations for these lower power states, taking into account the time it takes for a device to "wake up" (state transition delay) and the potential impact on task execution delays. Bringing these calculations together, the module then makes idle duration adjustments (e.g., lengthening or shortening the time spent idle before returning to active). It refines state transition delays to avoid unnecessary power toggling. This feedback loop continuously updates probabilities and power-saving factors

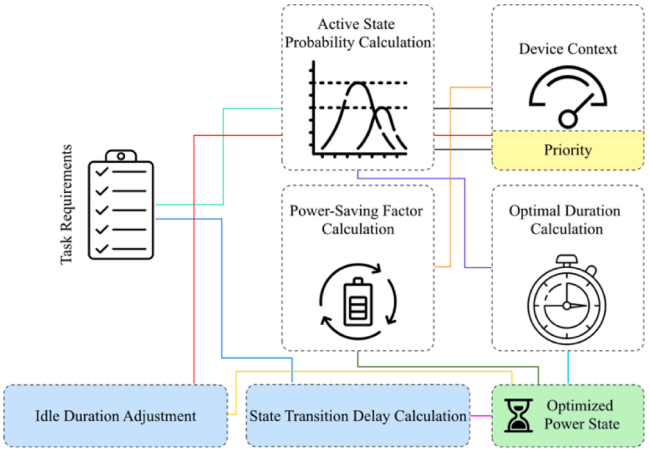


Fig. 3. The Proposed PSOM: Dynamic adjustments for efficient state transitions based on task and device context.

as new tasks and device statuses are reported. This ensures the optimized power state matches real-time conditions, significantly improving energy efficiency across the IoT network.

3.3. Self-Learning control mechanism

The Self-Learning Control Mechanism uses the SAC algorithm to help IoT devices optimize energy use independently. This mechanism includes two main parts: the SAC algorithm, which supports continuous learning, and a Contextual Adaptation Mechanism that adjusts energy policies in real-time based on the device's conditions. With this framework, each IoT device can change its energy allocation dynamically, achieving optimal power efficiency and better task performance.

3.3.1. SAC algorithm

The SAC algorithm is a model-free, off-policy RL method that optimizes energy allocation policies by balancing exploration and exploitation. It is particularly suited for this system because it operates on continuous action spaces, allowing fine-grained control over energy allocation decisions. The algorithm consists of an actor network for policy generation and two critic networks for value estimation, which guide the actor toward optimal decisions under varying conditions.

Policy Update: we update the SAC policy π_θ by minimizing the Kullback-Leibler divergence D_{KL}, which measures the difference between the current and optimal energy-efficient policies. This alignment ensures precise energy allocation under dynamic IoT conditions, as described in Eq. (26):

$$\pi_\theta = \arg\min_\theta \sum_{i=1}^{N} D_{\mathrm{KL}}\left(\pi_\theta(s_t|a_t)|\pi_{\mathrm{target}}(s_t|a_t)\right) \tag{26}$$

where π_{target} is the target policy, s_t is the device state and $a_t$is the action is taken.

Actor-Network Objective: The actor-network is optimized to maximize the expected reward $E[R_t]$, which in this case includes a penalty for high energy usage. Eq. (27) represents this objective:

$$\backslash J_\pi(\theta) = \sum_{t=0}^{T} E_{s_t,a_t\sim\pi_\theta}[R_t - \alpha \log \pi_\theta(a_t|s_t)] \tag{27}$$

where α is a temperature parameter that controls the exploration-exploitation trade-off.

Critic Network Update: The two critic networks Q_{ϕ_1} and Q_{ϕ_2} are optimized to minimize the Bellman error, ensuring accurate value estimation for energy efficiency, as described in Eq. (28).

$$J_Q(\phi_j) = \sum_{t=0}^{T} E_{(s_t,a_t)\sim\mathbb{D}}\left(Q_{\phi_j}(s_t,a_t) - \left(r_t + \gamma \underset{j=1,2}{min} Q_{\phi_j}(s_{t+1},a_{t+1})\right)\right)^2 \tag{28}$$

where $\mathscr{D}$ is the replay buffer, γ is the discount factor, and r_t is the reward for a state-action pair (s_t, a_t).

Reward Function for Energy Efficiency: The reward function $R(s_t, a_t)$ is designed to penalize high energy consumption while rewarding task success, as modeled in Eq. (29).

$$R(s_t,a_t) = \beta_1 \cdot \mathrm{TaskSuccess} - \beta_2 \cdot E_{\mathrm{alloc}}(a_t) \tag{29}$$

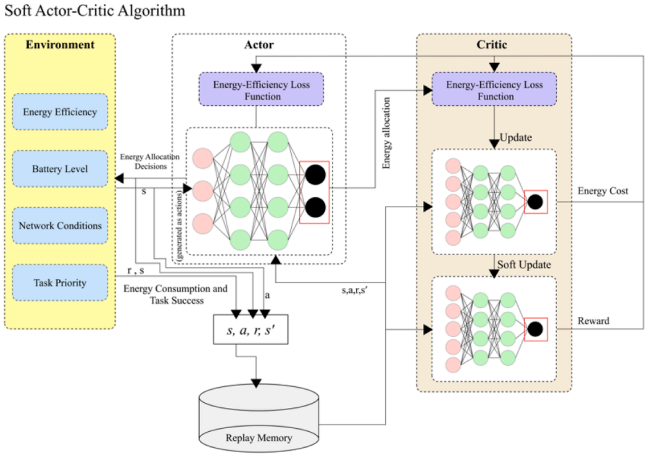


Fig. 4. The proposed SAC algorithms and agents architecture on IoT devices.

where β_1 and β_2 are weight parameters.

Entropy Maximization: To encourage exploration and prevent premature convergence, an entropy term $H(\pi)$ is added. This is calculated as Eq. (30).

$$J_{\text{entropy}} = -\sum_{t=0}^{T} H(\pi_\theta(a_t|s_t)) = -\sum_{t=0}^{T} E_{a_t\sim\pi}[\log\pi_\theta(a_t|s_t)] \tag{30}$$

This term ensures diverse action selection, leading to better adaptation in dynamic environments.

As depicted in Fig. 4, the Soft Actor-Critic (SAC) framework for energy management operates by taking in environmental inputs—such as battery level, task priority, and network conditions—as the current state s, which is then used by an actor-network to make energy-allocation decisions and by a critic network to evaluate the resulting energy efficiency. For each state, action a, reward r, and next state's$'$, a tuple is stored in a replay memory to decorate training samples. The actor's neural network processes the state through multiple hidden layers, ultimately outputting an energy-allocation policy (e.g., deciding how much power to allocate to tasks), guided by an energy-efficiency loss function combined with an entropy term that encourages exploration. Meanwhile, the critic network receives (s, α) pairs, calculates Q-values that capture both energy cost and task success and is updated based on a loss function comparing these Q-values to targets derived from the reward and next-state Q-values; soft updates help stabilize learning by gradually adjusting the critic's parameters to match a target network. During training, the actor samples experiences from the replay memory

Algorithm 1
Proposed Algorithm for IoT Device Energy Management.

Algorithm 1. Proposed Algorithm for IoT Device Energy Management	
Input	s_t: device state at time t (battery level, task priority, and load conditions). a_t: The actor-network took the action at time t . r_t: Reward received after executing a_t $\mathcal{D}$: replay buffer used to store experiences γ : Discount factor to balance immediate rewards versus long-term gains
Output	α : Temperature parameter to control the trade-off between exploration and exploitation Learning rates (updates for critic and actor networks), Batch size (number of samples for learning updates). I have Updated the Policy, Action for the next step, Critic Networks, Replay Buffer, and Temperature Parameter.
	Begin:
1	Initialize critic networks parameters φ1 **and** φ2, actor network parameters θ, Initialize temperature parameter α **for** entropy control Initialize replay buffer $\mathcal{D}$ to store experiences
2	For each **episode:**
3	Initialize initial state s0 based on current device conditions (battery, load, task priority)
4	For each **time step** t **in** the episode.
5	Sample action at ~ $\pi_\theta(a_t|s_t)$
6	Execute action at **and** observe reward r_t **and next** state s_{t+1}
7	Store transition (s_t, a_t, r_t, s_{t+1}) **in** replay buffer $\mathcal{D}$
8	Sample mini-batch of transitions (s_i, a_i, r_i, s_{i+1}) **from** replay buffer $\mathcal{D}$
9	For each sample **in** the mini-batch
10	Calculate target value y_t:
11	$y_t = r_i + \gamma * \mathbf{min}\left(Q\varphi1(s_{i+1}, a_{i+1}), Q\varphi2(s_{i+1}, a_{i+1})\right)$
12	Update critic networks φ1 **and** φ2 by minimizing the Bellman error loss
13	$J_Q(\varphi_j) = \Sigma\left(y_i - Q_{\varphi_j}(s_i, a_i)\right)^2$, for $j = \mathbf{1}, \mathbf{2}$
14	**End For**
15	Actor-Network Update
16	Calculate actor network loss
17	$J\pi_{(\theta)} = \Sigma E[r_t - \alpha * log\pi\theta(a_t|s_t)]$
18	Update actor network parameters θ to maximize $J\pi_{(\theta)}$
19	Adjust temperature α to optimize exploration-exploitation balance
20	$J_{entropy} = -\Sigma\ log\pi_\theta(a_t|s_t)$
21	**End For**
22	**End For**

refines its policy based on the critic's feedback, and continuously balances energy usage and task completion demands, improving overall energy efficiency in stable and dynamic conditions.

Algorithm 1 describes our proposed energy management for IoT devices using SAC. Its inputs include device conditions and outputs of updated policies, critic networks, and optimized exploration-exploitation through entropy adjustment.

The proposed SAC algorithm can be adapted to various IoT scenarios by dynamically adjusting the device states s_t and actions a_t based on task loads, energy constraints, and network congestion. In smart home environments, task prioritization during peak usage hours ensures that critical devices like surveillance cameras or thermostats maintain functionality while minimizing energy spikes.

3.3.1.1. Algorithm parameters and experimental setup. This section outlines the key parameters and configuration details used to implement and evaluate this study's SAC algorithm. These parameters were fine-tuned to balance computational efficiency, learning stability, and energy optimization for IoT-MEC systems.

Learning Rate (Actor and Critic Networks): The learning rate for the actor and critic networks was set to 0.005, balancing convergence speed and stability.

Discount Factor (γ): A discount factor of 0.95 was used to emphasize long-term rewards and ensure the model focuses on sustained energy efficiency.

Temperature Parameter (α): The entropy weight controlling the exploration-exploitation trade-off was dynamically adjusted to ensure optimal policy exploration without overfitting.

Batch Size: For each update, a mini-batch size of 64 transitions was sampled from the replay buffer, ensuring sufficient variance in the learning process.

Replay Buffer Size: The replay buffer size was set to 1000,000 transitions to store historical state-action-reward trajectories for efficient off-policy learning.

Critic Networks Architecture: Two critic networks with identical architectures were used, each comprising three hidden layers with 128 neurons and ReLU activation functions.

Actor Network Architecture: The actor network's architecture was similar to that of the critics, ensuring symmetry and efficient policy generation.

3.3.2. Contextual adaptation mechanism

The Contextual Adaptation Mechanism allows each IoT device to adjust its energy allocation policies dynamically based on real-time feedback from environmental and operational conditions, such as battery level, network status, and task load. This mechanism leverages the SAC policy learned by the device to interpret contextual data and make appropriate adjustments.

The reward function is scaled based on contextual variables C_t, such as battery level $B_i(t)$ and network load $L(t)$, to prioritize energy conservation, when necessary, as given by Eq. (31).

$$R'(s_t, a_t) = R(s_t, a_t) \cdot \left(1 - \frac{B_{\min}}{B_i(t) + \epsilon}\right) \cdot \left(1 - \frac{L(t)}{L_{\max} + \epsilon}\right) \tag{31}$$

where ϵ is a small constant for numerical stability.

The action a_t generated by the policy is adjusted based on the network load $L(t)$. This is represented as Eq. (32).

$$a_t' = a_t \cdot \left(1 - \frac{L(t)}{L_{\max}}\right) \tag{32}$$

Allowing for more conservative actions under high network congestion to conserve energy.

To preserve battery life, the policy introduces an adaptation factor γ_B, which scales actions according to current battery levels, modeled as Eq. (33).

$$\gamma_B(t) = \frac{B_i(t) - B_{\text{crit}}}{B_{\max} - B_{\text{crit}}} \tag{33}$$

This factor ensures that actions are adjusted proportionally to the remaining battery capacity. The scaled action is then calculated as Eq. (34).

$$a_t^{\text{adapted}} = a_t \cdot \gamma_B(t) \tag{34}$$

An environmental factor scales the policy output $\kappa(E_i)$, which adjusts energy allocation in response to external conditions such as temperature or device wear, is given by Eq. (35):

$$a_t^{\text{final}} = a_t^{\text{adapted}} \cdot \kappa(E_i) \tag{35}$$

where $\kappa(E_i) = \exp(-\lambda E_i)$ is an environmental sensitivity parameter.

To determine whether to execute an action under certain conditions, a probability P_{exec} is introduced by Eq. (36).

$$P_{\text{exec}}(a_t) = \sigma\left(\frac{B_i(t)}{B_{\max}} \cdot \frac{L_{\max} - L(t)}{L_{\max}}\right) \tag{36}$$

Where $\sigma(\cdot)$ increases action probability under favorable battery and network conditions.

Through the integration of the SAC algorithm with the Contextual Adaptation Mechanism, each IoT device is empowered to autonomously adjust energy usage in response to dynamic conditions, achieving optimal energy efficiency and extended device life in the proposed model.

Fig. 5 depicts the Contextual Adaptation Mechanism process for energy optimization in IoT devices within MEC systems. Real-time data on battery levels, network conditions, and task load flows into the SAC Policy, which makes energy allocation choices. Context variables, like network load and battery level, direct these choices, enabling the SAC Policy to adjust to changing conditions. The reward function encourages energy saving during critical moments, while the action adjustment and adaptation factor ensure conservative decisions when the battery is low or congestion is high. Once executed, the final decision reduces unnecessary power usage, improving energy efficiency. This mechanism lets IoT devices adapt energy use autonomously in real-time, supporting sustained operation and efficient energy management.

3.4. MEC coordination and policy distribution

The MEC Coordination and Policy Distribution components support energy-efficient operations for IoT devices. MEC nodes work as Data Aggregation Nodes, gathering only essential information from nearby IoT devices. This data helps balance energy locally and periodically distributes optimized energy management policies to each device. The Policy Update Distribution mechanism updates each device with a current, contextually optimized energy policy, promoting collaborative energy conservation across the network.

3.4.1. Data aggregation node

MEC nodes function as data aggregation hubs that collect minimal data from IoT devices in their vicinity. This data includes metrics such as battery levels, energy consumption rates, and network congestion, which inform localized energy management strategies. By aggregating this information, MEC nodes can make informed decisions about distributing policy adjustments to optimize energy usage across the network. The aggregated battery level $B_{\text{agg}}(t)$ at time t is calculated by Eq. (37).

$$B_{\text{agg}}(t) = \frac{1}{N}\sum_{i=1}^{N} B_i(t) \tag{37}$$

where $B_i(t)$ represents the battery level of each device i.

The average energy consumption rate $E_{\text{rate}}(t)$ across all devices in the MEC node's area, is computed as shown in Eq. (38).

$$E_{\text{rate}}(t) = \frac{1}{N}\sum_{i=1}^{N} \frac{E_i(t)}{T_{\text{sample}}} \tag{38}$$

where $E_i(t)$ does the device consume the energy i over the sampling period T_{sample}.

A network congestion indicator $C_{\text{net}}(t)$ is generated based on the average load across all connected devices, as given by Eq. (39).

$$C_{\text{net}}(t) = \frac{1}{N}\sum_{i=1}^{N} \frac{L_i(t)}{L_{\text{max}}} \tag{39}$$

where $L_i(t)$ is the load on device i and L_{max} is the maximum allowable load for any device.

MEC nodes compute a weighted context value $C_{\text{agg}}(t)$ by assigning weights w_i based on device priority or task criticality. This is calculated as Eq. (40).

$$C_{\text{agg}}(t) = \sum_{i=1}^{N} w_i \cdot \left(\frac{B_i(t) + E_i(t)}{L_i(t) + \epsilon}\right) \tag{40}$$

where ϵ is a small constant to avoid division by zero.

MEC nodes only aggregate data from devices that meet certain thresholds (e.g., low battery or high energy consumption) to reduce

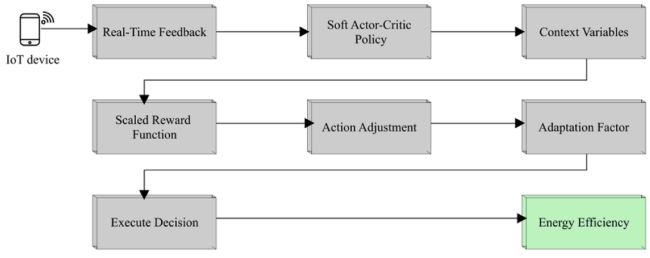


Fig. 5. Contextual Adaptation Mechanism for Energy Optimization in IoT Devices.

communication overhead. The filtered aggregation F_{agg} is defined as Eq. (41).

$$F_{\text{agg}}(t) = \sum_{i=1}^{N} 1_{\{B_i(t) \leq B_{\text{th}} \text{ o } E_i(t) \geq E_{\text{th}}\}} \cdot (B_i(t) + E_i(t)) \tag{41}$$

where B_{th} and E_{th} are the battery and energy thresholds, and 1 is an indicator function.

3.4.2. Policy update distribution

The Policy Update Distribution mechanism in MEC nodes periodically sends optimized energy management policies to nearby IoT devices. These policy updates are based on aggregated data and distributed without relying on a centralized server, allowing each device to adapt locally. This decentralized approach enhances scalability and reduces communication latency.

Policy updates are dispatched at predefined intervals T_{update}, optimized to balance responsiveness and network efficiency, given by Eq. (42).

$$T_{\text{update}} = \frac{\tau}{1 + C_{\text{net}}(t)} \tag{42}$$

where τ is a base update interval adjusted dynamically based on the network congestion indicator $C_{\text{net}}(t)$.

The updated policy $\pi_{\text{new}}(t)$ for the device, i is adjusted based on the weighted context $C_{\text{agg}}(t)$, it is calculated as Eq. (43).

$$\pi_{\text{new},i}(t) = \pi_{\text{old},i}(t) + \lambda \cdot C_{\text{agg}}(t) \tag{43}$$

where λ is a learning rate factor, and $\pi_{\text{old},i}(t)$ is the previous policy.

To prioritize devices with critical battery levels, the updated weight $W_{\text{update}}(i,t)$ for each device is scaled as Eq. (44).

$$W_{\text{update}}(i,t) = \frac{B_{\text{agg}}(t)}{B_i(t) + \epsilon} \tag{44}$$

which reduces the update frequency for devices with low batteries, conserving their energy.

MEC nodes selectively distribute updates to devices that exceed a predefined energy consumption threshold, represented by the indicator function $1_{\{E_i(t) > E_{\text{t}}\}}$ as described in Eq. (45).

$$\pi_{\text{dist},i}(t) = \pi_{\text{new},i}(t) \cdot 1_{\{E_i(t) > E_{\text{t}}\}} \tag{45}$$

When multiple MEC nodes are present, they collaboratively adjust policies based on consensus values π_{cons}, defined by Eq. (46).

$$\pi_{\text{cons}}(t) = \frac{1}{M} \sum_{m=1}^{M} \pi_{\text{new},m}(t) \tag{46}$$

Where M is the number of MEC nodes in the area, promoting consistency across devices.

MEC nodes assess the effectiveness of distributed updates by computing the energy efficiency improvement $\Delta\eta_i(t)$ for each device, modeled as Eq. (47):

$$\Delta\eta_i(t) = \frac{\eta_{\text{new},i}(t) - \eta_{\text{old},i}(t)}{\eta_{\text{old},i}(t)} \tag{47}$$

Where $\eta_{\text{new},i}(t)$ and $\eta_{\text{old},i}(t)$ represent energy efficiency before and after the update.

With Data Aggregation Nodes and Policy Update Distribution in the proposed architecture, MEC nodes enhance energy conservation across IoT devices by providing optimized, context-driven policies. This decentralized approach ensures that devices receive necessary adjustments without dependency on a central server, enabling adaptive, scalable energy management across the network.

3.5. System operation flow

The System Operation Flow outlines the stages of the proposed framework, covering everything from setup to adaptive control. This flow consists of three main stages: initializing device policies, adaptive learning, and convergence using the SAC algorithm with MEC support, as well as real-time energy management. These stages enable devices to adjust dynamically to environmental and operation changes.

3.5.1. Initialization

At the beginning of system operation, each IoT device initializes its energy management policy based on its current battery level, workload, and priority tasks. This initialization ensures that devices start with an optimized baseline for energy conservation.

Initial Policy Setup: The initial energy policy $\pi_{\text{init},i}$ for each device is calculated based on its battery level B_i, workload W_i, and task priority α_i, as shown in Eq. (48).

$$\pi_{\text{init},i} = f(B_i, W_i, \alpha_i) = \frac{\alpha_i \cdot B_i}{W_i + \epsilon} \tag{48}$$

where ϵ is a small constant for stability.

Baseline Energy Allocation: The baseline energy allocation $E_{\text{base},i}$ is set based on a weighted function of battery capacity $B_{\text{cap},i}$ and device workload, given by Eq. (49).

$$E_{\text{base},i} = \frac{\sum_{k=1}^{K} W_{k,i}}{B_{\text{cap},i}} \cdot B_i \tag{49}$$

where K is the total number of tasks assigned to device i.

Task Scheduling Priority: The priority of each task k is assigned based on energy and task urgency, with an initial priority weight $P_{k,i}$. This is modeled as Eq. (50).

$$P_{k,i} = \frac{B_i}{B_{\max}} \cdot \left(\frac{T_{\text{deadline},k}}{T_{\text{total},i}} \right) \tag{50}$$

Where $T_{\text{deadline},k}$ is the task deadline and $T_{\text{total},i}$ is the total expected operational time of the device.

3.5.2. Adaptive learning and convergence

The Adaptive Learning and Convergence phase involves each IoT device continuously refining its energy policy through the SAC algorithm. MEC nodes provide support by gathering local data, facilitating faster SAC model convergence. The SAC algorithm updates the device policy π_θ based on the reward function $R(s,a)$, where the reward incorporates energy efficiency and task success, given Eq. (51).

$$R(s_t, a_t) = \beta_1 \cdot \text{TaskSuccess} - \beta_2 \cdot \frac{E_{\text{consumed}}(t)}{E_{\text{alloc}}(t)} \tag{51}$$

with weights β_1 and β_2 balancing task success and energy usage.

The SAC algorithm adjusts the policy parameters θ using a gradient ascent approach to maximize the expected reward, modeled as Eq. (52).

$$\theta_{t+1} = \theta_t + \eta \cdot \nabla_\theta E[R(s_t, a_t)] \tag{52}$$

where η is the learning rate.

MEC nodes provide aggregated context values C_{agg} to aid in policy convergence, allowing devices to adapt faster. The convergence rate $\tau_{\text{conv},i}$ for device i is modeled as Eq. (53).

$$\tau_{\text{conv},i} = \frac{1}{1 + \lambda \cdot C_{\text{agg}}} \tag{53}$$

where λ is a scaling factor, and C_{agg} represents the aggregated context from neighboring devices.

To ensure stability, each device's policy is considered converged if the change in policy $\Delta\pi_i$ over successive steps is below a threshold ϵ_{conv}. This is calculated as Eq. 54.

$$\Delta\pi_i = \left| \pi_{i,t+1} - \pi_{i,t} \right| < \epsilon_{\text{conv}} \tag{54}$$

3.5.3. Real-Time Energy Management

In the Real-Time Energy Management phase, IoT devices continuously adapt their energy usage to current conditions, adjusting power levels and switching between idle and active states to optimize energy efficiency.

Each device adjusts its energy allocation $E_i(t)$ based on the battery level $B_i(t)$, network load $L(t)$, and task priority, represented as Eq. (55).

$$E_i(t) = \pi_i(t) \cdot \left(\frac{B_i(t)}{B_{\max}} \right) \cdot \left(1 - \frac{L(t)}{L_{\max}} \right) \tag{55}$$

The probability $P_{\text{active}}(t)$ of transitioning from idle to active state depends on the task demand $D_k(t)$ and current battery level, given by Eq. (56).

$$P_{\text{active}}(t) = \sigma\left(\frac{D_k(t)}{B_i(t) + \epsilon} \right) \tag{56}$$

The power allocation $P_{\text{alloc}}(t)$ for active tasks is determined by Eq. (57).

$$P_{\text{alloc}}(t) = P_{\text{idle}} + \sum_{k=1}^{K} P_k(t) \cdot 1_{D_k(t)>\delta} \tag{57}$$

where $P_k(t)$ is the power required for task k, δ is a threshold, and 1 is an indicator function for task activation.

Devices dynamically adjust task scheduling based on battery health $H_i(t)$, they were given by Eq. (58).

$$H_i(t) = \frac{B_i(t)}{B_{\text{init}}} - \frac{\sum_{j=1}^{J} D_j}{T_{\text{oper}}} \tag{58}$$

where B_{init} is the initial battery level, D_j is the energy drain from recent tasks and T_{oper} is the operational time.

To prevent energy-intensive tasks under high network load, a conditional function scales back energy allocation when $L(t)$ exceeds a threshold L_{th}. This is modeled as Eq. (59).

$$E_i^{\text{scaled}}(t) = E_i(t) \cdot 1_{L(t)\leq L_t} \tag{59}$$

Devices prioritize high-importance tasks, adapting energy policies $\pi_i(t)$ to ensure essential tasks continue even with low battery, as given by Eq. (60).

$$\pi_i(t) = \pi_{\text{base}}(t) \cdot \left(1 + \frac{\alpha}{B_i(t)}\right) \tag{60}$$

where α is a scaling factor based on task criticality.

3.6. Performance metrics

A set of key performance metrics is established to evaluate the effectiveness of the proposed IoT-MEC system. These metrics quantify the system's efficiency in conserving energy, prolonging device life, adapting to dynamic contexts, and managing power states.

Device energy consumption $E_{\text{total},i}$ measures the cumulative energy each IoT device i uses over a specified period T, modeled as Eq. (61).

$$E_{\text{total},i} = \sum_{t=0}^{T} E_i(t) \tag{61}$$

Where $E_i(t)$ is the energy consumed at time t.

Battery life improvement ΔB_i evaluates the increase in operational time for each device compared to a baseline (e.g., conventional energy management), given by Eq. (62).

$$\Delta B_i = \frac{B_{\text{system},i} - B_{\text{baseline},i}}{B_{\text{baseline},i}} \times 100 \tag{62}$$

where $B_{\text{system},i}$ is the battery life under the proposed system, and $B_{\text{baseline},i}$ is the battery life under baseline conditions.

Adaptability A_i measures how well the system responds to changes in environmental conditions such as battery level, network congestion, and task load, represented as Eq. (63).

$$A_i = \frac{\sum_{t=0}^{T} \left| E_i(t) - E_{\text{optimal},i}(t) \right|}{T} \tag{63}$$

Where $E_{\text{optimal},i}(t)$ represents the ideal energy allocation under contexts at time t.

Idle-to-active energy efficiency η_{active} is the ratio of energy saved during transitions between idle and active states relative to a baseline, as modeled in Eq. (64).

$$\eta_{\text{active}} = \frac{\sum_{t\in T_{\text{active}}} E_{\text{idle}} - E_{\text{active}}}{\sum_{t\in T_{\text{active}}} E_{\text{active}}} \tag{64}$$

Where E_{idle} and E_{active} represent the energy levels in idle and active states over active transition periods T_{active}.

4. Simulation setup and parameters

Proposed architecture simulations include 100 IoT devices and 10 MEC nodes across a ten km^2 area. We use Python for RL algorithms and OMNeT++ for network simulations. Each IoT device has a starting battery of 4000 mAh and adjusts its power use according to its state. Devices consume 10 W idle, 50–100 W during active processing, and 40 mW and 30 mW for data transmission and reception. They shift between active and idle states based on battery levels, task importance, and network conditions. In a smart home context, the simulation emphasizes managing diverse devices such as smart thermostats, surveillance cameras, bright lights,

door locks, entertainment systems, and more, ensuring efficient energy usage across fluctuating activity patterns. This helps maximize energy savings with a SAC algorithm in Python. Each MEC node has a 20 GHz computational capacity and acts as a coordinator, gathering data on network load and battery status. MEC nodes update policies in a distributed way without centralizing tasks. They adjust the interval for policy updates (policy) based on network conditions, with a maximum load ($L_{\max}$) of 500 tasks per second to keep network efficiency and energy balanced. The SAC algorithm on each IoT device is implemented in TensorFlow. This choice ensures scalability and compatibility with modern RL libraries, enabling efficient computation during simulations. The learning rate is set to 0.005 for steady progress, balancing convergence speed with stability to prevent oscillations in policy updates. The discount factor (γ) is 0.95 to focus on long-term energy savings, ensuring that energy-efficient decisions consider immediate benefits and sustained device operation over extended periods. The SAC policy has two critic networks and an actor-network to allocate energy based on real-time device conditions and environment. The likelihood of a device staying active (active) depends on its battery level (B_i), task priority (α), and network load. Policy updates are kept in a replay buffer, and each policy is tested over a 48-hour simulation, with data recorded every minute for detailed energy tracking. OMNeT++ manages the network communication between IoT devices and MEC nodes. Bandwidth changes from 10 Mbps to 100 Mbps, depending on distance, congestion, and environmental interference, while packet loss varies from 0.5 % to 3 % with network load. Each device adjusts packet sizes (data) according to network conditions, saving energy by reducing transmission frequency. Task arrival follows a Poisson distribution, with one task arriving every 10–15 s, doubling during peak times. Task sizes follow a normal distribution (mean 1 MB, standard deviation 0.5 MB), with deadlines ranging from 5 to 60 s based on urgency, affecting energy use and scheduling.

Although this setup focuses on a moderate-scale network of 100 IoT devices and 10 MEC nodes, its decentralized architecture and SAC-based optimization approach can be scaled to more extensive networks. By distributing policy updates through MEC nodes and leveraging localized decision-making, the framework minimizes communication overhead, enabling application to smart cities with thousands of devices. Table 2 presents the simulation parameters and their value.

5. Simulation results and discussion

In this section, we present and analyze the simulation results to assess the effectiveness of our proposed model. We conducted multiple simulations under different conditions to ensure the accuracy and reliability of our findings. We included realistic network parameters and device contexts. Cross-validation techniques were applied to check for consistency across scenarios. We also compared our results with established benchmarks (Local Computing, Random Scheduling, DQN, and PPO) to validate the effectiveness of our approach. The benchmarks used for comparison were carefully selected to represent a diverse range of approaches commonly applied in IoT energy management. Local Computing and Random Scheduling were chosen as traditional, non-optimized baselines, highlighting the improvements offered by RL-based models. DQN and PPO were included as they represent state-of-the-art RL methods widely used in similar domains, providing a robust point of comparison for our proposed algorithm.

In the analysis of Device Energy Consumption, as shown in Fig. 6, we investigate the performance of various algorithms under two scenarios: High-Activity Environment and Variable Network Load. In the first scenario, characterized by frequent and intensive device activity, SAC demonstrates notable energy efficiency, converging to a lower energy consumption rate of approximately 15,811 Joules by the end of 24 hours. PPO and DQN follow closely, with final consumption values of 16,812 and 17,801 Joules, respectively, showing the convergence typical of RL algorithms under high load. Random Scheduling and Local Computing, however, do not exhibit convergence, with Local Computing reaching the highest consumption of 29,019 Joules, indicating its inefficiency in high-activity settings. In the second scenario, focusing on variable network load, SAC again performs best, converging at 14,839 Joules. PPO and DQN also show efficiency but at slightly higher consumption levels, stabilizing around 17,881 and 16,805 Joules, respectively. Random Scheduling and Local Computing maintain higher, variable consumption rates, with Local Computing reaching 27,069 Joules.

Table 2
Simulation parameters and their value.

Parameter	Value/Range
Number of IoT Devices	100
Initial Battery Capacity	4000 mAh
Dynamic Active Power Consumption	50–100 W based on task load
Idle Power Consumption	10 W
Transmission Power	40 mW
Reception Power	30 mW
MEC Node Computational Capacity	20 GHz
Maximum Coordination Load	Dynamic, capped at 500 tasks/s
Simulation Duration	48 hours
Learning Rate (SAC)	0.005
Discount Factor (γ)	0.95
Dynamic Task Arrival Rate	1 task every 10–15 s, +100 % during peak periods
Task Size (mean, std dev)	Mean: 1 MB, Std Dev: 0.5 MB
Task Deadlines	5 to 60 s
Bandwidth	10 to 100 Mbps, varies dynamically
Packet Loss Rate	0.5 % to 3 % dynamically
Probability of Active State	Calculated dynamically
Energy Efficiency	Calculated during transitions

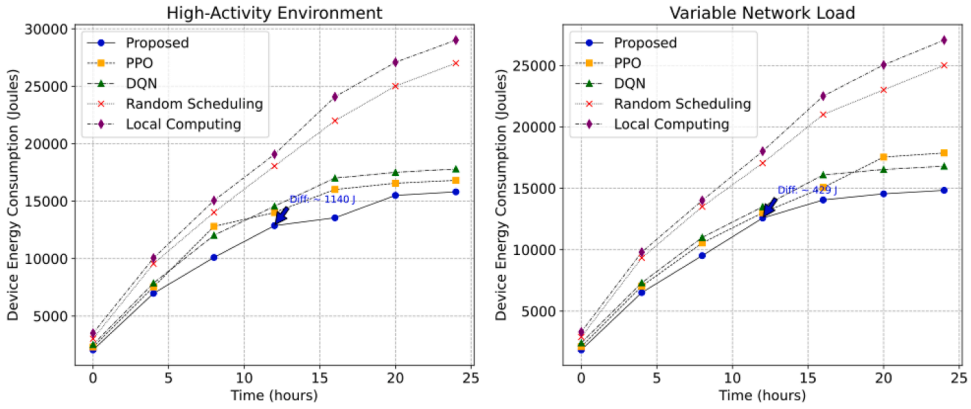


Fig. 6. Device energy consumption under high-activity and variable network load environments across algorithms.

These findings underscore SAC's superior adaptability and energy efficiency across different environmental conditions, highlighting its potential for reducing energy consumption in dynamic IoT-MEC environments.

The analysis of Table 3 provides insights into the comparative performance of various algorithms under high-activity IoT environments. The table highlights key energy efficiency metrics such as total energy consumption, average consumption rate, percentage reduction compared to Local Computing, convergence points, variance, and peak energy consumption. The proposed algorithm demonstrates the lowest total energy consumption of 15,811 Joules (J), significantly outperforming Local Computing, which exhibits the highest consumption of 29,019 J. This efficiency translates into an average consumption rate of 658.8 J/hr, substantially lower than Local Computing's 1209.1 J/hr. Regarding energy reduction relative to Local Computing, the proposed model achieves a 45.5 % reduction, while PPO and DQN achieve 42.1 % and 38.7 %, respectively. Random Scheduling provides a marginal decrease of 6.9 %. Convergence points for the proposed algorithm and PPO occur after 20 hours, with DQN converging earlier at 16 hours, indicating their stability over time. The variance in energy consumption is lowest for the proposed algorithm (21,487,274 J^2), demonstrating stable performance, whereas Random Scheduling and Local Computing exhibit significantly higher variance, suggesting fluctuating energy demands. The peak energy consumption aligns with the final consumption values for each algorithm.

The Battery Life Improvement analysis in Fig. 7, evaluates algorithmic performance under two distinct scenarios: High Task Load with Frequent Offloading and Moderate Task Load with Energy-Saving Strategies. In the High Task Load with Frequent Offloading scenario, IoT devices face heavy computational demands, requiring frequent offloading of tasks to maintain functionality, which strains battery life and challenges energy management efficiency. This scenario simulates a high-activity environment where algorithms must adapt quickly to continuous energy demands. In contrast, the Moderate Task Load with Energy-Saving Strategies scenario involves a steady, lower task load where devices benefit from proactive energy-saving measures, allowing algorithms to optimize battery usage gradually and consistently across extended periods. In the first scenario, our algorithm gradually improves with high task demand and frequent offloading, reaching a battery life increase of approximately 33 % by 24 hours. PPO and DQN algorithms follow, with final improvements of 30.21 % and 28.01 %, respectively, showing comparable but slightly lower efficiency. Random Scheduling and Local Computing exhibit significantly lower improvements, stabilizing at approximately 16.11 % and 14.51 % battery life gains, respectively, indicating their inefficiency in high-demand environments. In the second scenario, characterized by moderate task load and more consistent energy-saving conditions, SAC shows a substantial increase in battery life, achieving about 49 % improvement by the end of the interval. PPO and DQN also perform well, with final improvements of about 45 % and 42 %, respectively. Random Scheduling and Local Computing again lag, reaching only 20 % and 17.55 % battery life improvements.

The observed differences, such as SAC's superior energy efficiency and adaptability, can be attributed to its applied optimization methods, ability to handle continuous action spaces, and entropy-regularized policy, ensuring stable learning even in dynamic environments.

The statistical analysis of Battery Life Improvement metrics, as presented in Table 4, offers a detailed comparison of algorithmic

Table 3
Comparative analysis of energy consumption metrics across different algorithms in high-activity IoT environments.

Method	Total Energy Consumption (J)	Avg. Consumption Rate (J/hr)	Reduction vs. Local (%)	Convergence Point (hours)	Variance (J^2)	Peak Energy Consumption (J)
Proposed	15,811	658.79	45.5	20	21,487,274	,15811
PPO	16,812	700.5	42.1	20	25,446,572	,16812
DQN	17,801	741.70	38.7	16	28,392,761	,17801
Random Scheduling	27,012	1125.5	6.9	-	64,293,504	,27012
Local Computing	29,019	1209.12	0	-	74,808,214	,29019

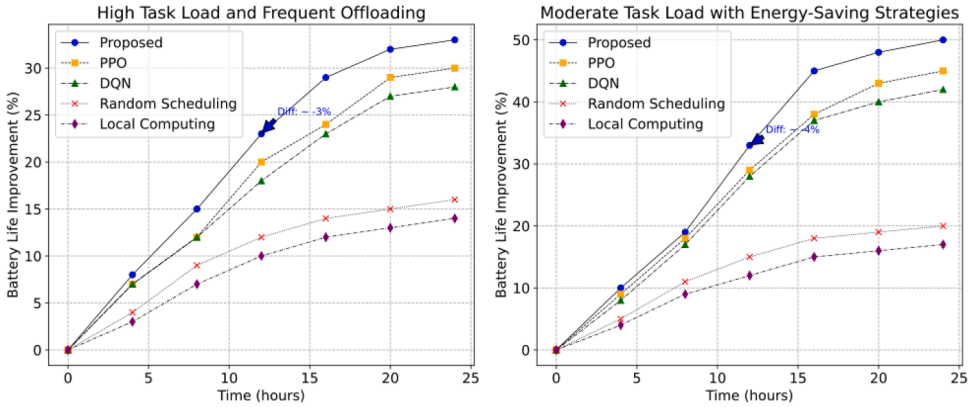


Fig. 7. Battery life improvement comparison across algorithms under high task load with offloading and moderate task load with energy-saving strategies.

Table 4
Battery life improvement metrics comparison across algorithms.

Method	Final Battery Improvement (%)	Avg. Improvement Rate (% per hr)	Improvement vs Local (%)	Variance in Improvement (%)
Proposed	33	1.375	135.7	138.85714
PPO	30	1.25	114.3	111.95918
DQN	27.9	1.166	98	95.67346
Random Scheduling	16.1	0.66	14.3	31.14285714
Local Computing	14.5	0.58	0	24.24489796

performance under various conditions. The proposed SAC algorithm achieves the highest final battery improvement of 33 %, outperforming Random Scheduling (16.1 %) and Local Computing (14.5 %). It also demonstrates the highest average improvement rate of 1.375 % per hour, showcasing its rapid energy optimization capabilities, compared to Local Computing's 0.58 %. Regarding relative improvement over Local Computing, SAC achieves a 135.7 % improvement, highlighting its significant battery-saving advantages. PPO and DQN also perform well with 114.3 % and 98 % improvements, respectively. Variance analysis reveals that the proposed algorithm and PPO maintain high but controlled fluctuations (138.9 % and 112 %, respectively), reflecting consistent performance. In contrast, Local Computing displays the lowest variance of 24.2 %, albeit at the expense of overall efficiency.

Fig. 8 assesses the algorithms' responsiveness across two scenarios: Dynamic Network Load with Frequent Congestion and Gradual Changes in Battery Levels and Task Load. The Dynamic Network Load with Frequent Congestion scenario simulates a high-variability environment where network conditions rapidly change due to congestion, challenging each algorithm to maintain adaptability amid fluctuating resources. In this setting, devices experience unpredictable shifts in connectivity and load, requiring efficient energy management to stabilize performance. The Gradual Changes in Battery Levels and Task Load scenario, on the other hand, involve a more controlled environment with slow, predictable changes in battery levels and workload, allowing algorithms to optimize adaptability gradually without sudden resource demands. In the first scenario, the Proposed method demonstrates robust adaptability,

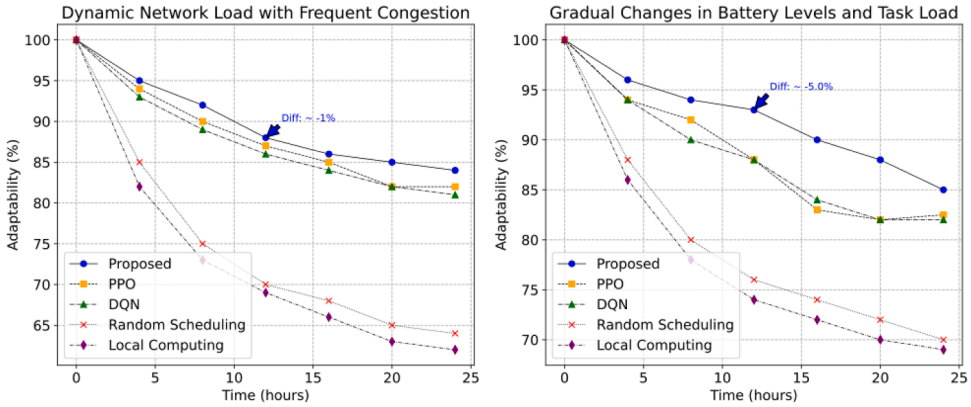


Fig. 8. Adaptability comparison of algorithms under dynamic network load with frequent congestion and gradual changes in battery levels and task load.

maintaining a high-efficiency level close to 84 % by the end of the interval, with PPO and DQN following at 82 % and 81 %. Random Scheduling and Local Computing show substantial adaptability degradation, ending at about 65 % and almost 61 %, respectively, indicating their struggles in highly variable conditions. All algorithms perform better in the second scenario, which involves more stable conditions with gradual battery and task load changes. SAC maintains an adaptability rate of about 85 %, followed closely by PPO and DQN.

In the Idle-to-Active Energy Efficiency analysis, as depicted in Fig. 9, two scenarios were examined to evaluate energy management during task transitions. The Frequent Short Tasks with High Transition Rate scenario simulates an environment where devices frequently switch between idle and active states due to numerous short tasks, placing a high demand on energy efficiency during transitions. In contrast, the Longer Tasks with Fewer Transitions scenario involves extended task durations and fewer state changes, allowing algorithms to focus on sustained energy management over longer active periods. Here, the Proposed algorithm maintains the highest energy efficiency, reaching by the end of the period, with PPO and DQN also performing well but with slightly lower final efficiencies. Random Scheduling and Local Computing demonstrate lower efficiency, struggling to adapt effectively to frequent transitions. In the second scenario, all algorithms perform better with fewer transitions associated with longer tasks.

Our proposed model consistently performs better in analyzing the results across all metrics due to its advanced self-learning framework, which dynamically adapts energy policies based on real-time conditions. The Device Energy Consumption analysis reveals that the proposed model minimizes energy usage most effectively, converging at lower levels under high-activity and variable network load scenarios. This advantage arises from our model's ability to make fine-grained adjustments in power management, unlike Random Scheduling and Local Computing, which lack adaptability and consume significantly more energy. The Battery Life Improvement metric further supports the proposed scheme's strength in extending device operation. Under high task loads, our model steadily improves battery life through rapid, context-aware energy optimization. In contrast, it shows substantial adaptability under moderate loads, surpassing other methods by a wide margin. The energy efficiency metrics analyzed in this study demonstrate technical superiority and significant potential for real-world impact. For example, the observed 45.5 % reduction in energy consumption directly translates into measurable cost savings for IoT system operators. This efficiency could lower electricity costs in a smart home environment by reducing the power requirements of devices like thermostats, surveillance cameras, and lighting systems. Furthermore, the substantial 49 % improvement in battery life implies reduced frequency of recharging or replacing batteries, leading to lower operational expenses and enhanced user convenience. Adaptability analysis shows the proposed method's robust response in dynamically changing environments with frequent congestion, maintaining higher adaptability rates than PPO and DQN and significantly outperforming traditional algorithms. This is due to SAC's framework, which efficiently manages network and task load fluctuations. Finally, the Idle-to-Active Energy Efficiency analysis underscores SAC's strength in managing frequent transitions. The proposed model maintains higher efficiency in scenarios requiring rapid task switching by optimizing transitions between active and idle states, minimizing unnecessary energy loss. Our algorithm outperforms others for scenarios with fewer transitions by focusing on sustained energy allocation during longer tasks.

The choice of SAC for our proposed model is driven by its unique advantages in the IoT-MEC context. SAC is particularly effective in handling continuous action spaces, critical for fine-grained control over energy allocation decisions in dynamic environments. Unlike PPO, which operates on discrete or bounded action spaces and often struggles with stability in highly dynamic scenarios, SAC uses an entropy-regularized objective function. This ensures better exploration of the action space, making the proposed model highly adaptable to fluctuating network conditions and task loads. Additionally, while DQN performs well in discrete action scenarios, they are less suited for continuous control problems inherent in IoT-MEC systems. DQN relies on action discretization, which can lead to suboptimal energy allocation in tasks requiring precise adjustments. In contrast, SAC's actor-critic framework optimizes energy usage with minimal oscillations, enabling efficient transitions between active and idle states. This makes the proposed algorithm particularly well-suited for decentralized, real-time energy management in diverse IoT environments, where adaptability and stability are essential. This ensures optimal energy efficiency in different settings. For instance, the model in smart home systems reduces the need for frequent device recharging by optimizing power usage dynamically, extending device lifespans. The proposed model may maintain operational stability and prevent energy drain in industrial IoT setups with high-frequency tasks, even during fluctuating network loads. Also, it should be tested in this area to ensure its effectiveness. Its adaptability to congestion is also valuable in urban

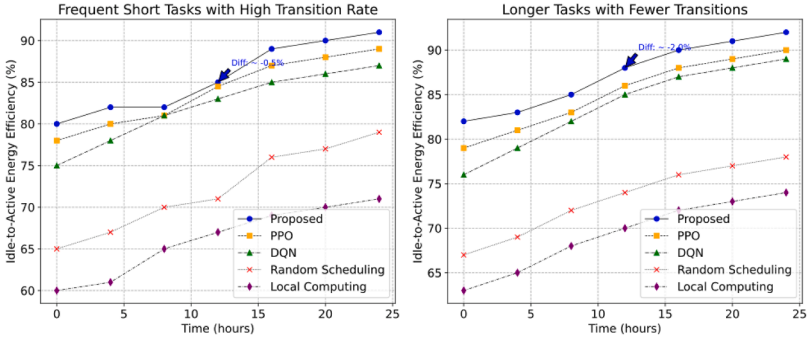


Fig. 9. Idle-to-active energy efficiency comparison of algorithms under frequent short tasks with high transition rates and longer tasks with fewer transitions.

infrastructure monitoring, where IoT devices face unpredictable communication delays. The model's quick response and intelligent power management help reduce energy waste in IoT systems. It adjusts power use based on task importance, battery levels, and network conditions, making devices last longer while working efficiently. This method reduces costs by extending device life, decreasing the need for frequent battery changes, and conserving energy in large setups like smart homes and factories.

Furthermore, several key features of its design support the scalability of the proposed framework to more extensive networks with higher device density and task complexity. The framework's reliance on decentralized energy management allows IoT devices to operate autonomously, reducing the computational and communication burden on MEC nodes. The SAC algorithm's entropy-regularized objective function also ensures stable learning and adaptability even in dynamic environments with high device density. By leveraging localized decision-making, MEC nodes can efficiently distribute energy policies without creating bottlenecks. Furthermore, the modular architecture of the power management and contextual adaptation mechanisms allows for seamless integration of additional devices or tasks. These elements enable the framework to handle the challenges of scaling to dense networks while maintaining energy efficiency and system stability.

While the study primarily addresses smart home IoT environments, the proposed SAC-based framework can be adapted to other IoT scenarios. For example, in 5G-enabled environments, the model can exploit the enhanced ultra-reliable low-latency communication (URLLC) features of 5 G to support rapid policy updates and real-time task offloading among densely deployed IoT devices. The reward function would require recalibration to include latency-sensitive metrics, such as task completion time and network throughput, ensuring efficient energy allocation under high-frequency state transitions. Additionally, the distributed MEC coordination mechanism must scale to handle the increased volume of data and computational demands inherent in 5 G networks. However, the framework may struggle for highly mobile IoT applications like drones in logistics networks due to frequent state transitions and elevated data exchange demands. Recalibrating the SAC parameters for rapid mobility scenarios and integrating predictive mobility models with policy updates could mitigate these challenges while ensuring scalability and adaptability. Another area to consider is LPWANs, where devices prioritize energy conservation over high-speed communication, the framework's entropy tuning parameters and learning rates must be optimized to balance the sparse transmission patterns and constrained power budgets. The communication model would need an adaptation to reflect LPWAN characteristics, such as higher packet loss rates and limited payload sizes. Integrating ultra-low-power energy allocation modules with the existing SAC structure could extend battery life further. The dynamic duty cycling of LPWAN devices based on task criticality and network conditions would enhance the system's adaptability.

6. Conclusion and future works

This paper presented a self-learning, adaptive power management framework for IoT-MEC systems using an optimized SAC algorithm. Our proposed model demonstrated optimized performance in multiple metrics, achieving the lowest total energy consumption, with the highest battery life improvement. In IoT-enabled smart homes, this energy reduction translates to extended operation of various devices, such as smart thermostats, surveillance cameras, and intelligent lighting systems. For instance, a smart thermostat can continue optimizing indoor temperatures for more extended periods without frequent recharging, while surveillance cameras maintain uninterrupted monitoring and recording, enhancing home security. Similarly, intelligent lighting systems can operate more efficiently, providing ambient or automated lighting while reducing electricity usage. Despite these promising results, our study has limitations. First, the simulations were conducted in moderate-scale environments, which may not entirely reflect the complexities and demands of large-scale IoT deployments. Second, the model's adaptability to varying high device mobility remains unexplored and may impact system stability and energy efficiency in real-world scenarios. Third, our framework does not address security concerns that could arise in distributed MEC settings, leaving potential vulnerabilities unmitigated. Fourth, real-world IoT implementations often face hardware limitations, communication delays, and integration challenges with existing MEC systems, which were not fully addressed in this simulation-based study. Mitigation strategies such as incorporating hardware resource profiling, latency-aware communication protocols, and compatibility layers for integration with diverse MEC platforms should be explored to enhance the deploy ability of the proposed model. Future work could address these limitations and evaluate scalability by applying the framework to industrial environments or larger-scale deployments and incorporating hybrid RL approaches (combining model-based and model-free techniques) or multi-Agent RL (to enable collaborative optimization across devices) to improve decision-making in highly dynamic environments.

CRediT authorship contribution statement

Amir Masoud Rahmani: Writing – review & editing, Resources. **Amir Haider:** Writing – review & editing, Data curation. **Komeil Moghaddasi:** Writing – original draft, Visualization, Methodology, Conceptualization. **Farhad Soleimanian Gharehchopogh:** Writing – original draft, Validation, Methodology, Conceptualization. **Khursheed Aurangzeb:** Writing – review & editing, Resources. **Zhe Liu:** Writing – review & editing, Resources. **Mehdi Hosseinzadeh:** Writing – review & editing, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This Research is funded by Researchers Supporting Project Number (RSPD2025R947), King Saud University,Riyadh, Saudi Arabia

Data availability

No data was used for the research described in the article.

References

[1] S. Dong, et al., Task offloading strategies for mobile edge computing: A survey, Comput. Networks 254 (2024) 110791, https://doi.org/10.1016/j.comnet.2024.110791, 2024/12/01/.
[2] K. Moghaddasi, S. Rajabi, F.Soleimanian Gharehchopogh, M. Hosseinzadeh, An energy-efficient data offloading strategy for 5G-enabled vehicular edge computing networks using double deep q-network, Wireless Personal Comm. 133 (3) (2023) 2019–2064, https://doi.org/10.1007/s11277-024-10862-5, 2023/12/01.
[3] Q. Chen, Z. Guo, W. Meng, S. Han, C. Li, T.Q.S. Quek, A survey on resource management in joint communication and computing-embedded SAGIN, IEEE Comm. Surveys Tutorials (2024) 1, https://doi.org/10.1109/COMST.2024.3421523. -1.
[4] K. Moghaddasi, S. Rajabi, F.S. Gharehchopogh, Multi-objective secure task offloading strategy for blockchain-enabled IoV-MEC systems: a double deep Q-network approach, IEEE Access 12 (2024) 3437–3463, https://doi.org/10.1109/ACCESS.2023.3348513.
[5] K. Moghaddasi, S. Rajabi, F.S. Gharehchopogh, A. Ghaffari, An advanced deep reinforcement learning algorithm for three-layer D2D-edge-cloud computing architecture for efficient task offloading in the Internet of Things, Sustain. Comput.: Inf. Syst. 43 (2024) 100992, https://doi.org/10.1016/j.suscom.2024.100992, 2024/09/01/.
[6] S. Zeadally, F.K. Shaikh, A. Talpur, Q.Z. Sheng, Design architectures for energy harvesting in the Internet of Things, Renew. Sustain. Energy Rev. 128 (2020) 109901, https://doi.org/10.1016/j.rser.2020.109901, 2020/08/01/.
[7] S.S. Sefati, et al., A comprehensive survey on resource management in 6G network based on internet of things, IEEE Access 12 (2024) 113741–113784, https://doi.org/10.1109/ACCESS.2024.3444313.
[8] T. Ahmad, D. Zhang, Using the internet of things in smart energy systems and networks, Sustain. Cities Society 68 (2021) 102783, https://doi.org/10.1016/j.scs.2021.102783, 2021/05/01/.
[9] W. Cai, et al., A review on methods of energy performance improvement towards sustainable manufacturing from perspectives of energy monitoring, evaluation, optimization and benchmarking, Renew. Sustain. Energy Rev. 159 (2022) 112227, https://doi.org/10.1016/j.rser.2022.112227, 2022/05/01/.
[10] H. Min, A.M. Rahmani, P. Ghaderkourehpaz, K. Moghaddasi, M. Hosseinzadeh, A joint optimization of resource allocation management and multi-task offloading in high-mobility vehicular multi-access edge computing networks, Ad. Hoc. Netw. 166 (2025) 103656, https://doi.org/10.1016/j.adhoc.2024.103656, 2025/01/01/.
[11] A.M. Rahmani, J. Tanveer, F.S. Gharehchopogh, S. Rajabi, M. Hosseinzadeh, A novel offloading strategy for multi-user optimization in blockchain-enabled Mobile Edge Computing networks for improved Internet of Things performance, Comput. Electr. Eng. 119 (2024) 109514, https://doi.org/10.1016/j.compeleceng.2024.109514, 2024/10/01/.
[12] Y. Himeur, A.N. Sayed, A. Alsalemi, F. Bensaali, A. Amira, Edge AI for internet of energy: challenges and perspectives, Internet. Things 25 (2024) 101035, https://doi.org/10.1016/j.iot.2023.101035, 2024/04/01/.
[13] J. Yang, A.A. Shah, D. Pezaros, A survey of energy optimization approaches for computational task offloading and resource allocation in MEC networks, Electronics (Basel) 12 (17) (2023) 3548 [Online]. Available, https://www.mdpi.com/2079-9292/12/17/3548.
[14] X. Wang, et al., Wireless powered mobile edge computing networks: a survey, ACM Comput. Surv. 55 (13s) (2023) 263, https://doi.org/10.1145/3579992. Article.
[15] L. Xu, M. Qin, Q. Yang, K.S. Kwak, Learning-aided dynamic access control in MEC-enabled green IoT networks: a convolutional reinforcement learning approach, IEEE Trans. Veh. Technol. 71 (2) (2022) 2098–2109, https://doi.org/10.1109/TVT.2021.3135885.
[16] J. Xu, B. Ai, L. Chen, Y. Cui, N. Wang, Deep reinforcement learning for computation and communication resource allocation in multiaccess MEC assisted railway IoT networks, IEEE Trans. Intell. Transp. Syst. 23 (12) (2022) 23797–23808, https://doi.org/10.1109/TITS.2022.3205175.
[17] H. Zhou, K. Jiang, X. Liu, X. Li, V.C.M. Leung, Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing, IEEE Internet. Things J. 9 (2) (2022) 1517–1530, https://doi.org/10.1109/JIOT.2021.3091142.
[18] X. Chen, G. Liu, Energy-efficient task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge networks, IEEE Int. Things J. 8 (13) (2021) 10843–10856, https://doi.org/10.1109/JIOT.2021.3050804.
[19] H.M. Do, T.P. Tran, M. Yoo, Deep reinforcement learning-based task offloading and resource allocation for industrial IoT in MEC federation system, IEEE Access 11 (2023) 83150–83170, https://doi.org/10.1109/ACCESS.2023.3302518.
[20] J.A. Ansere, et al., Optimal computation resource allocation in energy-efficient edge IoT systems with deep reinforcement learning, IEEE Trans. Green Comm. Network. 7 (4) (2023) 2130–2142, https://doi.org/10.1109/TGCN.2023.3286914.
[21] T. Du, X. Gui, X. Teng, K. Zhang, D. Ren, Dynamic trajectory design and bandwidth adjustment for energy-efficient UAV-assisted relaying with deep reinforcement learning in MEC IoT system, IEEE Int. Things J. (2024) 1, https://doi.org/10.1109/JIOT.2024.3421616. -1.
[22] R. Xu, Z. Chang, Z. Han, S. Garg, G. Kaddoum, J.J.P.C. Rodrigues, Energy-efficient joint optimization of sensing and computation in MEC-assisted IoT using mean-field game, IEEE Int. Things J. (2024) 1, https://doi.org/10.1109/JIOT.2024.3443701. -1.
[23] Y. Xiao, Y. Song, J. Liu, Collaborative multi-agent deep reinforcement learning for energy-efficient resource allocation in heterogeneous mobile edge computing networks, IEEE Trans. Wireless Commun. 23 (6) (2024) 6653–6668, https://doi.org/10.1109/TWC.2023.3335597.
[24] G. Wu, Z. Xu, H. Zhang, S. Shen, S. Yu, Multi-agent DRL for joint completion delay and energy consumption with queuing theory in MEC-based IIoT, J. Parallel Distrib. Comput. 176 (2023) 80–94, https://doi.org/10.1016/j.jpdc.2023.02.008, 2023/06/01/.
[25] S. Chouikhi, M. Esseghir, L. Merghem-Boulahia, Energy-efficient computation offloading based on multiagent deep reinforcement learning for industrial internet of things systems, IEEE Int. Things J. 11 (7) (2024) 12228–12239, https://doi.org/10.1109/JIOT.2023.3333044.
[26] A.R. Heidarpour, M.R. Heidarpour, M. Ardakani, C. Tellambura, M. Uysal, Soft actor–critic-based computation offloading in multiuser MEC-enabled IoT—A lifetime maximization perspective, IEEE Int. Things J. 10 (20) (2023) 17571–17584, https://doi.org/10.1109/JIOT.2023.3277753.
[27] B. Lu, J. Fang, X. Hong, J. Shi, Energy saving computation offloading for dynamic CR-MEC systems with NOMA via decomposition based soft actor–critic, Expert Syst. Appl. 255 (2024) 124455, https://doi.org/10.1016/j.eswa.2024.124455, 2024/12/01/.
[28] W. Wen, H. Cui, T. He, Multi-layer reinforcement learning assisted task offloading in satellite edge computing, IEEE Trans. Veh. Technol. (2024).
[29] K. Cheng, et al., GeoScale: microservice autoscaling with cost budget in geo-distributed edge clouds, IEEE Trans. Parallel Distrib. Syst. 35 (4) (2024) 646–662, https://doi.org/10.1109/TPDS.2024.3366533.
[30] H. Huang, W. Zhan, G. Min, Z. Duan, K. Peng, Mobility-aware computation offloading with load balancing in smart city networks using MEC federation, IEEE Trans. Mob. Comput. 23 (11) (2024) 10411–10428, https://doi.org/10.1109/TMC.2024.3376377.
[31] J. Zhou, Y. Zhao, L. Zhao, H. Cai, F. Xiao, Adaptive task offloading with spatiotemporal load awareness in satellite edge computing, IEEE Trans. Network Sci. Eng. 11 (6) (2024) 5311–5322, https://doi.org/10.1109/TNSE.2024.3368086.