Measurement: Sensors 31 (2024) 100939

Contents lists available at ScienceDirect

Measurement: Sensors

journal homepage: www.sciencedirect.com/journal/measurement-sensors

EdgeAISim: A toolkit for simulation and modelling of AI models in edge computing environments

Aadharsh Roshan Nandhakumar [a,d], Ayush Baranwal [a,d], Priyanshukumar Choudhary [b,d], Muhammed Golec [c,d,*], Sukhpal Singh Gill [d]

[a] Indian Institute of Information Technology, Allahabad, India
[b] National Institute of Technology, Rourkela, India
[c] Abdullah Gul University, Kayseri, Turkey
[d] School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK

ARTICLE INFO

Keywords:
Edge AI
Edge computing
Artificial intelligence
Toolkit
Machine learning
Cloud computing
Simulation
Modelling
EdgeAISim
Python

ABSTRACT

To meet next-generation Internet of Things (IoT) application demands, edge computing moves processing power and storage closer to the network edge to minimize latency and bandwidth utilization. Edge computing is becoming increasingly popular as a result of these benefits, but it comes with challenges such as managing resources efficiently. Researchers are utilising Artificial Intelligence (AI) models to solve the challenge of resource management in edge computing systems. However, existing simulation tools are only concerned with typical resource management policies, not the adoption and implementation of AI models for resource management, especially. Consequently, researchers continue to face significant challenges, making it hard and time-consuming to use AI models when designing novel resource management policies for edge computing with existing simulation tools. To overcome these issues, we propose a lightweight Python-based toolkit called EdgeAISim for the simulation and modelling of AI models for designing resource management policies in edge computing environments. In EdgeAISim, we extended the basic components of the EdgeSimPy framework and developed new AI-based simulation models for task scheduling, energy management, service migration, network flow scheduling, and mobility support for edge computing environments. In EdgeAISim, we have utilized advanced AI models such as Multi-Armed Bandit with Upper Confidence Bound, Deep Q-Networks, Deep Q-Networks with Graphical Neural Network, and Actor-Critic Network to optimize power usage while efficiently managing task migration within the edge computing environment. The performance of these proposed models of EdgeAISim is compared with the baseline, which uses a worst-fit algorithm-based resource management policy in different settings. Experimental results indicate that EdgeAISim exhibits a substantial reduction in power consumption, highlighting the compelling success of power optimization strategies in EdgeAISim. The development of EdgeAISim represents a promising step towards sustainable edge computing, providing eco-friendly and energy-efficient solutions that facilitate efficient task management in edge environments for different large-scale scenarios.

1. Introduction

In the contemporary digital landscape, edge computing has emerged as a seminal approach that significantly reshapes data handling, processing, and analysis [1]. It represents a pivotal departure from the conventional centralized data processing infrastructure of distant cloud data centers [2]. Edge computing involves a decentralized approach to computing, focusing on processing and analyzing data closer to where it originates, near the network's edge, rather than relying on central cloud data centers [3]. The decentralized approach of edge computing plays a pivotal role in managing the substantial data volumes generated by the Internet of Things (IoT) [4]. In IoT, edge computing efficiently handles data from myriad devices. Its significance extends to diverse applications such as augmented reality/virtual reality (AR/VR), autonomous vehicles, smart cities, telecommunications, healthcare, and video surveillance, all of which necessitate rapid data processing [5].

* Corresponding author. School of Electronic Engineering and Computer Science, Queen Mary University of London, London, E1 4NS, UK.
E-mail addresses: aadh2001@gmail.com (A.R. Nandhakumar), baranwalayush25@gmail.com (A. Baranwal), priyanschoudhary100@gmail.com (P. Choudhary), m.golec@qmul.ac.uk (M. Golec), s.s.gill@qmul.ac.uk (S.S. Gill).

https://doi.org/10.1016/j.measen.2023.100939
Received 2 August 2023; Received in revised form 28 September 2023; Accepted 12 November 2023
Available online 17 November 2023

1.1. Challenges

The rising demand for edge computing is propelled by the ever-increasing volume of data in the digital age, primarily attributed to the IoT [6]. The rapid proliferation of edge computing has ushered in a new era of decentralized data processing, promising reduced latency, enhanced privacy, and improved efficiency [7]. This transition, however, presents significant challenges, with power consumption taking the forefront [8]. It becomes imperative to optimize energy usage for sustainability, cost reduction, and resource allocation efficiency for modern edge computing systems [9]. In summary, the imperatives of sustainability, cost-effectiveness, and efficient resource allocation become increasingly pronounced as the popularity of edge computing continues to soar [10]. Effective resource management is vital for maintaining consistent performance and extending the lifespan of edge servers while mitigating thermal constraints [11]. Researchers are utilising Artificial Intelligence (AI) models to solve the above-mentioned challenges of resource management in edge computing systems [12]. However, existing simulation tools are only concerned with typical resource management policies, not the adoption and implementation of AI models for resource management, especially [13]. Consequently, researchers continue to face significant challenges, making it hard and time-consuming to use AI models when designing novel resource management policies for edge computing with existing simulation tools [14–17].

1.2. Existing solutions

EdgeSimPy [17] is a Python-based framework tailored for modeling and simulating resource management policies within edge computing ecosystems and it solves the issues of existing simulators [14–16]. Distinguished by its modular architecture, EdgeSimPy encompasses functional abstractions for various components, including edge servers, network devices, and applications. This versatile tool empowers users to explore and optimize resource allocation strategies, enhancing the efficiency and performance of edge computing environments. In this section, we highlights the limitations of existing simulators, particularly EdgeSimPy and elucidates how our proposed toolkit, EdgeAISim overcomes these deficiencies through the integration of AI techniques tailored for edge computing environments.

1.2.1. Limited realism and Adaptability

Traditional simulators face difficulties representing the dynamic, heterogeneous edge environment realistically and adapting to diverse scenarios.

EdgeAISim, powered by AI, self-optimizes for enhanced realism. Reinforcement learning equips it to dynamically adapt resource management, ensuring a more realistic and adaptable depiction of edge computing dynamics in evolving conditions.

1.2.2. Inefficient resource management

Conventional simulators often use basic resource allocation methods, neglecting power optimization and task migration, resulting in suboptimal resource use.

EdgeAISim, driven by AI, dynamically manages resources, reducing power consumption. It employs techniques like Multi-Armed Bandit and Actor-Critic Reinforcement Learning for intelligent resource allocation, ensuring efficiency.

1.2.3. Lack of adaptation to emerging AI trends

Existing simulators often lack the capacity to assess emerging AI technologies and advanced reinforcement learning in enhancing edge computing efficiency.

Whereas, EdgeAISim positions itself at the forefront of research and development in edge computing. It can explore the synergy between AI and edge computing, enabling researchers to assess the potential benefits of cutting-edge AI techniques in real-world edge scenarios.

1.3. Our contributions

This paper meticulously explores the vast spectrum of applications of edge computing, accentuating its pivotal role in addressing the torrential influx of data generated during the IoT era. Furthermore, we introduce EdgeAISim, an advanced framework that offers cost-effective and sustainable resource management for edge computing systems. By focusing on sustainability, cost-effectiveness, and resource optimization, EdgeAISim strives to address a critical challenge in the evolving realm of edge computing. The ***main contributions*** of this work are:

- We proposed EdgeAISim, a Python-based framework that addresses the challenges of task migration and resource management in dynamic edge computing environments. Leveraging advanced algorithms and simulations, EdgeAISim enables seamless task migration and workload balancing across edge servers.
- We integrated advanced AI models such as Multi-Armed Bandit with Upper Confidence Bound, Deep Q-Networks, Deep Q-Networks with Graphical Neural Network, and Actor-Critic Network to optimize power usage while efficiently managing task migration within the edge computing environments.
- We compared the performance of these proposed models of EdgeAISim with the baseline, which uses a worst-fit algorithm-based resource management policy in different settings. Experimental results indicate that EdgeAISim exhibits a substantial reduction in power consumption, highlighting the compelling success of power optimization strategies in EdgeAISim. EdgeAISim minimizes power consumption, promoting sustainability, and reducing the environmental impact of energy-intensive infrastructures.

1.4. Article organization

The rest of the paper is organised as follows: Section 2 presents related work. Section 3 discusses background, including definitions and concepts. Section 4 presents the architecture of EdgeAISim. Section 5 discusses the design and implementation of EdgeAISim. Section 6 presents the experimental setup and results. Finally, Section 7 concludes the paper and highlights future directions.

2. Related work

The advent of edge computing has sparked the creation of numerous edge simulators. In this section, we provide an introduction to simulation tools dedicated to edge computing and subsequently conduct a comparative analysis, focusing on the distinct features and contributions of EdgeAISim as compared to these existing solutions. By evaluating capabilities of the EdgeAISim, we aim to highlight its potential advantages and novel contributions to the field of edge simulation. Calheiros et al. [18] developed CloudSim, a versatile simulator for cloud computing infrastructure. CloudSim aids researchers in conducting comprehensive studies on resource allocation, task scheduling, and energy consumption. It enables in-depth simulations, enhancing the development of cloud-based applications and the evaluation of cloud management strategies. Further, Sonmez et al. [15] extends the CloudSim [18] and address the limitations of cloud and network simulators in modeling edge environments. While cloud simulators like CloudSim lack user mobility and wireless support, network simulators may not cover edge servers and users. They introduce EdgeCloudSim [15], a simulator enabling user mobility, edge device power modeling, and network management for edge computing scenario prototyping. Moreover, Mahmud et al. [14] introduced iFogSim2, an enhanced version of iFogSim [16]. This toolkit emphasizes mobility, clustering, and microservice management in edge and fog computing environments. Researchers can explore the impact of mobility, clustering strategies, and

microservice management in these dynamic scenarios. iFogSim2 empowers researchers to optimize edge and fog computing applications for improved mobility, clustering, and microservice utilization.

Zeng et al. [19] introduced IOTSim, a dedicated simulator for analyzing IoT applications. This tool enables researchers to study the behavior and performance of IoT applications comprehensively. IOTSim facilitates resource management, communication protocols, and data processing evaluations in IoT scenarios, empowering researchers to optimize IoT solutions and enhance IoT applications' efficiency. Further, Jha et al. [20] tackle the complex Cloud-Edge-IoT ecosystem, involving resource allocation across diverse devices with various network and messaging protocols. To address the limitations of edge simulators for these protocols, they present IoTSim-Edge. This simulator enables experimentation with edge resource management, considering factors like energy use, application composition, user mobility, and communication protocols. Alwasel et al. [21] propose IoTSim-Osmosis, a novel simulator designed for Osmotic Computing scenarios to focus on workload migration between cloud data centers and edge devices based on performance and security events. This simulator includes various models for data transmission, energy consumption, and application performance. The authors also demonstrate IoTSim-Osmosis in a case study, showcasing its ability to model policies optimizing performance, energy usage, and cost in Cloud-Edge scenarios.

Wang et al. [22] introduces a versatile edge caching simulator offering key contributions. It facilitates quick mobile network setup, heterogeneous device simulation, and various scenarios. Its unique algorithm access framework simplifies integration. The simulator is highly flexible, supports custom models, and includes a learning-based caching algorithm for cloud collaborative intelligence, as demonstrated through performance evaluations. Qayyum et al. [23] critique existing edge simulators for their limitations in capturing network infrastructure nuances. They present FogNetSim++, an innovative simulator that addresses these issues by modeling power consumption, supporting various communication protocols, and simulating mobile user handovers. They demonstrate its effectiveness in a practical use case for designing edge-specific placement and scheduling policies.

Lera et al. [24] present YAFS, an edge simulator focused on evaluating allocation decisions in composite applications within edge infrastructures. YAFS employs routing policies to oversee communication among application modules, enabling the allocation of application components to diverse edge devices. The simulator showcases applications, including dynamic scheduling, infrastructure resilience, and user mobility support. Recently, Souza et al. [17] presents a Python-based framework called EdgeSimPy for modeling and simulating resource management policies within edge computing ecosystems. EdgeSimPy encompasses functional abstractions for various components, including edge servers, network devices, and applications.

2.1. Critical analysis

Table 1 shows the comparison of EdgeAISim with existing frameworks and simulators. Our research paper stands out in the field due to its innovative approach, which specifically targets the optimization of power consumption in edge computing systems using AI models, especially reinforcement learning techniques. This breadth of capabilities of EdgeAISim makes it a versatile tool for modeling and evaluating various edge computing scenarios. In contrast, while other simulators like CloudSim [18] and EdgeCloudSim [15] excel in specific areas such as task scheduling, they often lack the holistic approach that EdgeAISim provides. iFogSim2 [14], while encompassing energy management and service migration, lacks AI and network flow scheduling. To harness the full potential of edge computing, it necessitates advanced simulation tools capable of addressing its unique challenges to simulate AI based energy-efficient resource management polices for edge computing systems, which is also missing in EdgeSimPy [17]. Furthermore, our proposed simulator, EdgeAISim has ability to simulate the full spectrum of edge computing functionalities positions it as a promising choice for researchers seeking to conduct in-depth assessments of edge computing systems and applications, making it stand out in the landscape of edge computing simulation tools.

3. Background: definitions and concepts

This section discusses important definitions and concepts to understand this work.

3.1. Edge computing

Fig. 1 shows the basic architecture of edge computing, where it interacts with the IoT layer and the cloud computing layer. Edge

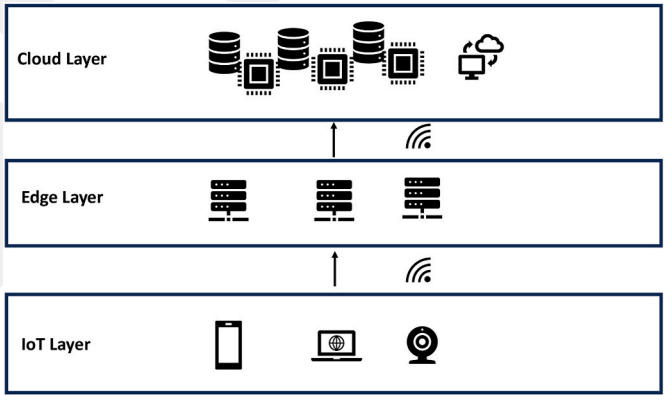

Fig. 1. Edge computing environments.

Table 1
Summary of built-in features supported by existing simulators and EdgeAISim.

Simulator	AI	Task Scheduling	Energy Management	Service Migration	Maintenance Operation	RL Algorithms	Mobility Support	Network flow Scheduling
CloudSim [18]	×	✓	✓	×	×	×	✓	×
EdgeCloudSim [15]	×	✓	✓	×	×	×	✓	✓
IOTSim [19]	×	✓	×	×	×	×	✓	✓
IFogSim2 [14]	×	✓	✓	✓	✓	×	✓	✓
SimEdgeIntel [22]	×	×	×	✓	×	✓	✓	✓
IoTsim-Edge [20]	×	✓	✓	×	×	×	✓	✓
IoTsim-Osmosis [21]	×	✓	✓	×	×	×	✓	✓
FogNetSim++ [23]	×	✓	✓	×	×	×	✓	×
YAFS [24]	×	✓	✓	×	×	×	✓	×
EdgeSimPy [17]	×	✓	×	✓	✓	×	✓	✓
EdgeAISim (this paper)	✓	✓	✓	✓	✓	✓	✓	✓

computing is a computing model that places resources closer to the network edge, enabling faster data processing and real-time capabilities [25]. It brings computing power and services nearer to data sources and end-users, reducing latency and improving application performance.

3.2. Problem formulation: task migration in edge computing

The task migration problem refers to the challenge of efficiently relocating and resuming the execution of computing tasks or processes from one computing resource, such as a server or a Virtual Machine (VM), to another while minimizing disruption and optimizing resource utilization [26]. In the realm of edge computing, the task migration challenge takes on heightened significance. Here, tasks frequently require dynamic relocation among edge devices for a range of purposes, including load distribution, fault resilience, energy conservation, and adapting to evolving resource requirements [27]. Proficient task migration strategies hold paramount importance within edge computing, as they serve to curtail service interruptions, mitigate data transfer overhead across potentially constrained network links, and enhance the efficient use of resources [28]. These strategies are indispensable for sustaining continuous and responsive edge services, particularly for applications demanding real-time performance and low-latency responsiveness.

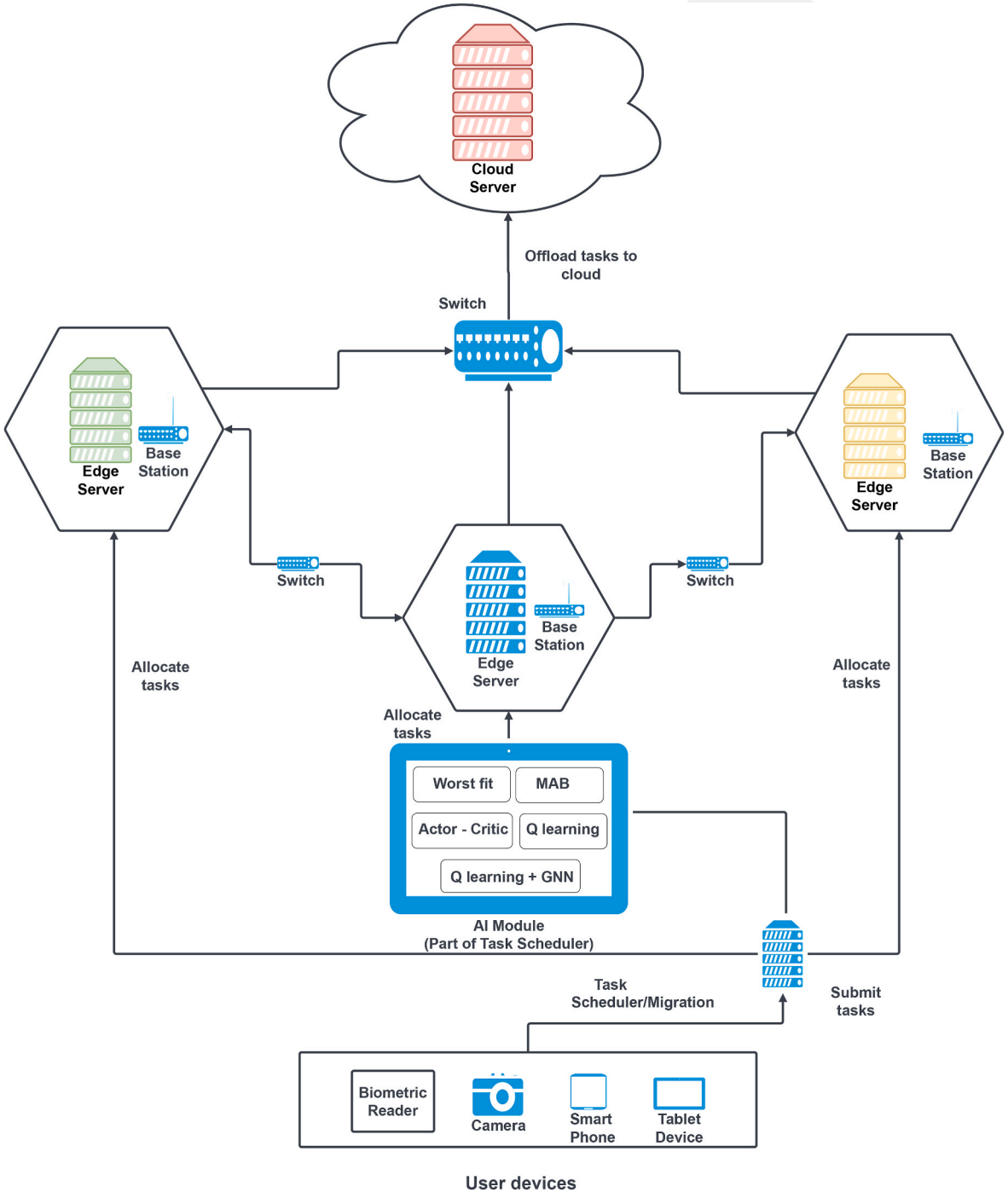

Fig. 2. EdgeAISim architecture.

4. EdgeAISim architecture

Fig. 2 shows the system architecture of EdgeAISim. In EdgeAISim, we extended the basic components of the EdgeSimPy framework [17] and developed new AI-based simulation models for task scheduling, energy management, service migration, network flow scheduling, and mobility support for edge computing environments. A basic edge cloud network consists of the following components.

4.1. Base stations

Base stations provide network connectivity to mobile computing devices within their coverage area. The entire map is divided into cells, where each base station assumes coverage of each cell. A mobile computing device anywhere in the cell is assumed to have equal connectivity to the cell's base station.

4.2. Network switches

Network switches are used to provide connections, typically wired, between base stations and edge servers. Task migrations are typically modeled as network flows, whose duration is determined by bandwidth scheduling. The Max-Min fairness algorithm [29] is used for bandwidth scheduling in network switches.

4.3. Modelling of resources

Edge servers are used to host services. Power consumption is modeled using three built-in power consumption models: LinearPowerModel, QuadraticPowerModel, and CubicPowerModel, which

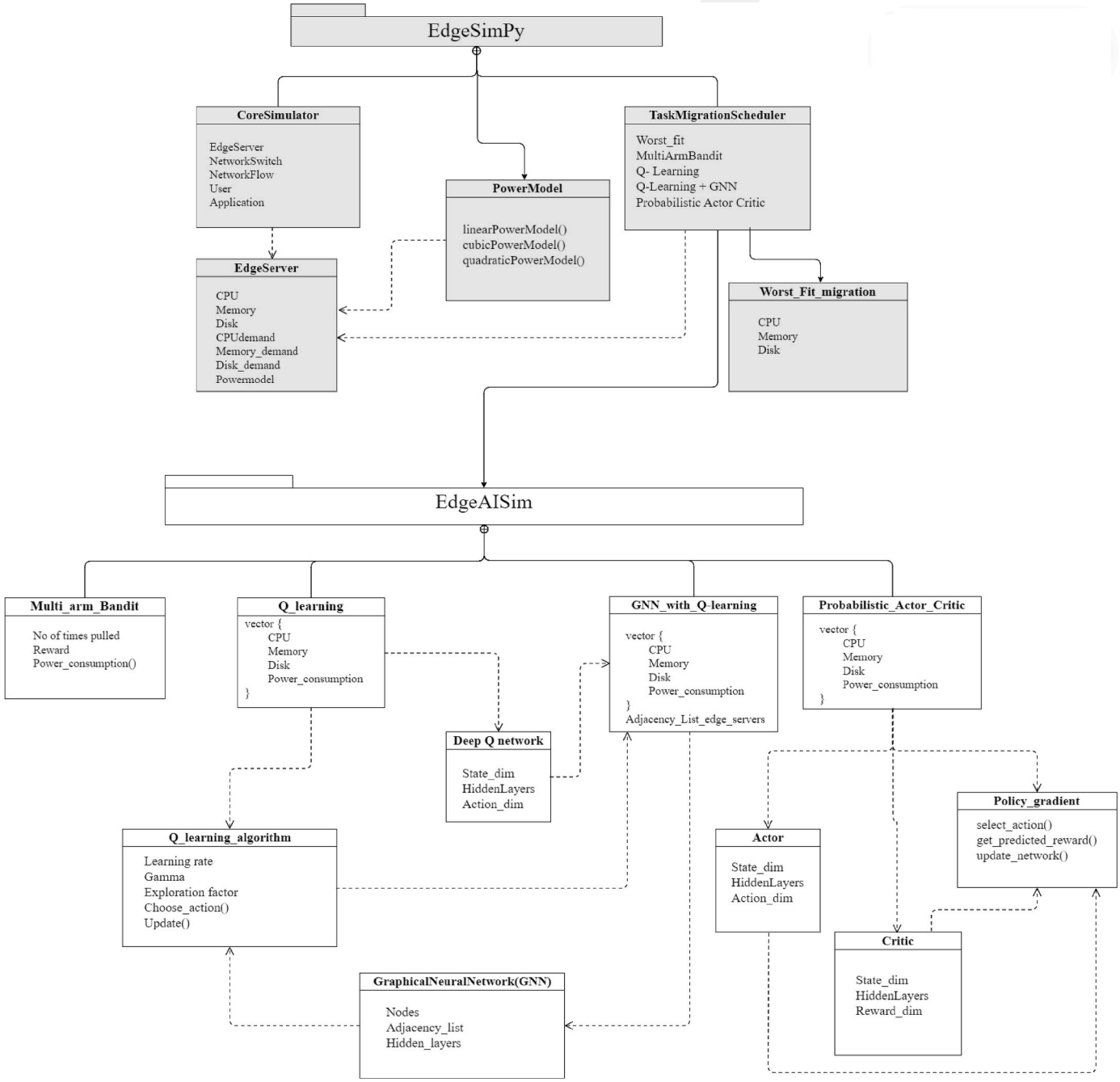

Fig. 3. Fundamental classes of EdgeAISim.

are dependent on 3 parameters: CPU, RAM, and hard disk utilization. In a linear power model, the parameters are dependent upon each other in a degree 1 polynomial equation. In a quadratic power model; the parameters are dependent upon each other in a degree 2 polynomial equation, whereas in a cubic power model, the parameters are dependent upon each other in a degree 3 polynomial equation. Edge servers run virtualized containers, enabling them to run multiple services at the same time.

4.4. Users

Users are the consumers of the services hosted on the edge Servers. Users move according to a defined mobility model, changing their access base station for the consumption of services present on the edge server.

4.5. Modelling of tasks

Tasks are modeled as applications or services. These applications or services have definite resource requirements, which are CPU demand and memory demand. When these are allocated to a specific edge server, the edge server will begin consuming the resources and, consequently, the change is reflected in its power consumption.

4.6. Modelling of AI models

The AI module consists of Reinforcement Learning (RL)-based task migration algorithms for the effective allocation of tasks, which is explained in Section 5.

5. Design and implementation of EdgeAISim

This section discusses the design and implementation of EdgeAISim.

5.1. Design description

The fundamental classes of EdgeAISim are shown in Fig. 3. The following are the main classes of EdgeAISim.

5.1.1. Core simulator

This class serves as the central orchestrator within the EdgeAISim system. Its primary role is to facilitate the seamless coordination and operation of various integral components, including network switches, edge servers, network flows, users, and more. One of its crucial functions is to instantiate instances of these component classes, creating individual entities representing network switches, edge servers, network flows, and users. Once these entities are created, this coordinating class assigns each of them to their designated positions and roles within the system.

5.1.2. PowerModel: modelling of energy consumption

This class is responsible for the practical implementation of several power models within the context of edge server and potentially other system components as well. These power models are essential tools for simulating and estimating power consumption. In the context of edge servers, power consumption is a crucial factor, as they often operate in resource-constrained and energy-sensitive environments. The class takes on the task of translating theoretical power models into functional code, allowing the simulation to accurately predict how much power an Edge server (or other components) will consume under different conditions.

5.1.3. Task migration scheduler

This class serves as a pivotal component within the edge computing system, responsible for the allocation of tasks to different edge servers. Its primary role is to make intelligent decisions about how to distribute tasks efficiently among available servers. The AI module, consisting of varying RL algorithms, is implemented by extending this class. Section 5.2 discusses the modelling and implementation of AI models for task migration.

5.2. Modelling and implementation of AI models for task migration

A task may be added to a migration queue due to Quality of Service (QoS) requirements, and the migration function called, which defines the migration algorithm. In this work, we have implemented a baseline resource management policy using a worst-fit algorithm [30,31]. Further, four advanced AI models (RL algorithms) such as Multi-Armed Bandit with Upper Confidence Bound, Deep Q-Networks, Deep Q-Networks with Graphical Neural Network, and Actor-Critic Network are utilized to optimize power usage while efficiently managing task migration within the edge computing environment. Each time the migration algorithm is called is modeled as one timestep. At each timestep, the inverse of the power consumption of each server is added and the total is used as our reward. The RL algorithm attempts to maximize our reward, and hence minimize our power consumption.

5.2.1. Baseline model: worst-fit migration algorithm

We have considered a worst-fit method [30,31] to develop a baseline policy for resource management. An algorithmic representation of Worst-Fit Algorithm is shown in Algorithm 1. This is a simple migration algorithm, where we sort the edge server by the difference between the available CPU and CPU demand in a decreasing order. Each service is migrated into the first server which has the available system requirements.

Algorithm 1. Worst-Fit Algorithm for Task Migrations in Edge-Cloud Computing

Input: Set of tasks $\mathcal{T}$, Set of edge servers $\mathcal{E}$, Migration threshold $M_{\text{threshold}}$
Initialize: Mapping of tasks to edge servers, $Mapping \leftarrow \{\}$
for each task $t_i \in \mathcal{T}$ **do**
  Find Candidate Edge Servers:
  $CandidateServers \leftarrow$ Find all edge servers $e_j \in \mathcal{E}$ with sufficient resources to accommodate t_i
  Select Worst-Fit Edge Server:
  Sort $CandidateServers$ in decreasing order of available resources
  $e_{\text{worst}} \leftarrow$ First server with the available resources among $CandidateServers$
  Offload task t_i to edge server e_{worst}
  $Mapping[t_i] \leftarrow e_{\text{worst}}$
end for
Output: Final mapping of tasks to edge servers $Mapping$

5.2.2. Proposed model: Multi-Armed Bandit with upper confidence bound

Multi-Arm Bandit refers to a decision-making problem where an agent must choose between multiple options (arms) with unknown rewards, aiming to maximize cumulative rewards by balancing exploration and exploitation [32]. *Upper Confidence Bound (UCB)* is a popular algorithm used in the Multi-Armed Bandit problem to balance exploration and exploitation [33]. An algorithmic representation of Upper Confidence Bound (UCB) Algorithm for Multi-Arm Bandit (MAB) is shown in Algorithm 2. It calculates an upper confidence bound for each action's expected reward, allowing the agent to prioritize actions with potentially higher payoffs while also exploring actions with uncertain rewards to gather more information. In our case, we initially set up action-value

estimates (Q(a)) and action counts (N(a)) for all arms (a). The exploration parameter ($c > 0$) is chosen, and the timestep (t) is initialized as 1. Throughout our experiments, we iterate the following steps until the timestep (t) reaches the maximum limit (T). At each timestep, we select the arm (At) that maximizes the UCB value, computed as the sum of the action-value estimate (Q(a)) and an exploration term, which involves the exploration parameter (c) and the square root of the natural logarithm of the action count (N(a)). Subsequently, we execute the chosen arm and observe the resulting reward (Rt). We then update the action count (N(At)) and action-value estimate (Q (At)) for the selected arm based on the observed reward. The action count for the chosen arm is incremented by 1, and the action-value estimate is updated using the incremental update formula. This process is repeated until the maximum time (T) is reached, allowing us to efficiently explore and exploit different arms dynamically and enabling effective decision-making in uncertain environments.

Complexity Analysis: Considering the overall operations, the dominant time complexity of the algorithm is determined by the arm selection step, which is O(K). Other operations within the loop are constant time operations and do not significantly contribute to the overall time complexity. Therefore, the time complexity of the given algorithm is O(K*T), where K is the number of arms and T is the total number of timesteps the algorithm runs.

Algorithm 2. Upper Confidence Bound (UCB) Algorithm for Multi-Arm Bandit

Initialize: Action-value estimates $Q(a)$ and action counts $N(a)$ for all arms a
Initialize: Exploration parameter $c > 0$
Initialize: Timestep t as 1
while $t \leq T$ **do**
  Choose arm A_t with highest UCB value:
  $A_t \leftarrow \arg\max_a \left(Q(a) + c\sqrt{\frac{\ln(t)}{N(a)}} \right)$
  Take action A_t and observe reward R_t
  Update action count:
  $N(A_t) \leftarrow N(A_t) + 1$
  Update action-value estimate:
  $Q(A_t) \leftarrow Q(A_t) + \frac{1}{N(A_t)}(R_t - Q(A_t))$
end while

We model a Multi-Armed Bandit for each service, where each arm represents an edge server. These arms can be 'pulled' i.e. the task can be migrated to the server, and a reward (that is the inverse of the current power consumption of the server) is received. The number of times each service has been migrated to the server is stored, such that, over time, the servers which have minimal power consumption while running the service is chosen with greater frequency.

5.2.3. Proposed model: deep Q-networks

Deep Q-Networks is a reinforcement learning algorithm that combines Deep Neural Networks (DNN) with the Q-Learning technique [34]. An algorithmic representation of Deep Q-Networks is shown in Algorithm 3. It enables agents to learn optimal action-selection strategies in environments with large state spaces. By approximating the action-value function using deep neural networks, Deep Q-Learning can handle complex and high-dimensional input data. Before beginning the training process, we initialise the Q-networks and learning rate (α). We sample a state (S) from the environment and select an action (A) using a greedy strategy based on the current Q-network during each episode of the training. The following state (S') and its accompanying reward (R) are then observed from the outside world. The Q-Learning equation, which incorporates the learning rate (α) and the discount factor (γ), is used to update the Q-values for the current state-action pair (S, A). Until convergence or a predetermined number of episodes (numEpisode), the process iterates over several episodes. By interacting with the environment, this method enables the agent to discover the best Q-values for the task at hand and gradually improve its performance.

Complexity Analysis: The time complexity for each iteration of the while loop (one episode) is O (|A|), where | A| is the number of actions. Since the while loop runs for "numEpisodes" times (N), the overall time complexity of the algorithm is O(N * |A|).

Algorithm 3. Deep Q-Learning

Initialize : $Q - network$
Initialize : learning rate α
while episode < num_episodes **do**
  $S \leftarrow$ sample state from environment
  $A \leftarrow \epsilon$ - greedy policy from Q network
  $S', R \leftarrow$ Next state and reward from environment
  $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma max_a(Q(S', a) - Q(S, a))]$
end while

We use a Deep Q-Network to determine the best server to migrate the services in the queue. The input to the Q-Network consists of a feature vector, consisting of the available CPU, available RAM, available Disk space, and the current power consumption of the edge server. The feature vector of all the edge server are concatenated and fed into the Q-Network. The Q-Network outputs a Q-value for each edge server. The edge server with the maximum Q-value is chosen for the service to be migrated to and the inverse of the power consumption is summed up and given as the reward.

5.2.4. Proposed model: deep Q-networks with graphical neural Network

Graphical Neural Network (GNN) is a neural network architecture specifically designed for processing graph-structured data [35]. It leverages the graph structure to capture dependencies and relationships between entities, enabling efficient representation and learning from graph data. GNN have shown promising results in tasks such as node classification, link prediction, and graph generation. An algorithmic representation of Graph Neural Network (GNN) for Message Passing is shown in Algorithm 4. In our work, we employ a GNN to update node features in a given graph = (V, E). We use node features $\mathscr{E}$ and edge features **X** as input. The GNN consists of multiple layers with message aggregation and node feature update functions. For each layer, we aggregate messages mv from neighboring nodes for each node v, ($v \in \mathscr{V}$), using a message aggregation function >(v,**X**). Then, we update node features **X**'V using an update function UpdateNodeFeatures (v,**X**, mv). This process is repeated for the specified number of layers, and the updated node features $\mathbf{X}'$ are used in the next layer. The GNN captures graph information and improves node representations for downstream tasks. Experimental evaluation on benchmark datasets demonstrates its effectiveness.

Complexity Analysis: The total time complexity for each layer is dominated by the message aggregation step because it takes O(d) time per node, whereas the node feature update step takes O (1) time per node. The given algorithm consists of "numlayers" iterations, where each iteration involves both message aggregation and node feature update steps. Thus, the overall time complexity for the algorithm is: O (numlayers × |V| × d) Where,

- |V| is the number of nodes in the graph.
- d is the average degree of nodes in the graph (assuming it's a constant).

Algorithm 4. Graph Neural Network (GNN) for Message Passing

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, Node Features $\mathbf{X}$, Edge Features $\mathbf{E}$
Output: Updated Node Features $\mathbf{X}'$
Initialize: GNN layers, message aggregation and update functions
for layer = 1 to num_layers **do**
  Message Aggregation:
  for each node $v \in \mathcal{V}$ **do**
    $\mathbf{m}_v \leftarrow \text{AggregatesMessages}(v, \mathcal{N}(v), \mathbf{X})$
  end for
  Update Node Features:
  for each node $v \in \mathcal{V}$ **do**
    $\mathbf{X}'_v \leftarrow \text{UpdateNodeFeatures}(v, \mathbf{X}, \mathbf{m}_v)$
  end for
  $\mathbf{X} \leftarrow \mathbf{X}'$ {Update node features for the next layer}
end for

We model the edge servers and the links between edge servers as a Graph, and feed it into a graphical neural network. Each edge server is a node with a feature vector, consisting of the available CPU, available RAM, available Disk space, and the current power consumption of the edge server. The GNN uses message passing to aggregate information from adjacent edge servers, and finally returns a feature vector, to be fed into the Deep Q-Network.

5.2.5. Proposed model: actor-critic Network

Actor-Critic is also a reinforcement learning framework that combines elements of both value-based and policy-based methods [36]. An algorithmic representation of Actor-Critic Algorithm with Probabilistic Actions is shown in Algorithm 5. It consists of two components: the actor, which selects actions based on a policy, and the critic, which estimates the value function or action-value function. The actor learns to improve the policy, while the critic provides feedback to the actor by estimating the quality or advantage of chosen actions. The actor network $\pi_\theta(a|s)$ with parameters θ and the critic network $V_\varphi(s)$ with parameters φ are initialized. Additionally, learning rates α_θ and α_φ, as well as the discount factor γ, are set. During the training process, episodes are executed, where each episode involves interacting with the environment and collecting states, actions, and rewards at each time step. We compute the discounted rewards-to-go for each step in the episode using the discount factor γ. The critic network is updated using the Advantage function, which is the difference between the discounted reward and the estimated state value from the critic network. The actor network is updated using the log-likelihood gradient, which is scaled by the Advantage function. By iteratively updating the actor and critic networks, our methodology enables the reinforcement learning system to learn effective policies for the given task. The outer loop runs for a specified number of episodes, and the inner loop runs within each episode until it is completed. In each step, the actor network samples a probabilistic action based on the current state, and the critic network estimates the state value. The algorithm then computes discounted rewards-to-go for each step in the episode and calculates the Advantage function. Using these values, it updates the critic and actor networks through gradient ascent.

Complexity Analysis: The overall time complexity of the algorithm is approximately $O(E^*L^2)$, where E is the number of episodes and L is the average episode length.

Algorithm 5. Actor-Critic Algorithm with Probabilistic Actions

Initialize: Actor network $\pi_\theta(a|s)$ with parameters θ, Critic network $V_\phi(s)$ with parameters ϕ
Initialize: Learning rates α_θ, α_ϕ, Discount factor γ
while episode < num_episodes **do**
  Initialize episode-specific lists: states = [], actions = [], rewards = []
  Receive initial state s
  while episode not finished **do**
    Sample probabilistic action $a \sim \pi_\theta(a|s)$
    Execute action a, observe reward r and next state s'
    Append s to states, a to actions, and r to rewards
    $s \leftarrow s'$
  end while
  Compute the discounted rewards-to-go for each step in the episode:
  for t from T to 0 **do**
    {where T is the last time step in the episode} $G_t = r_t + \gamma \cdot G_{t+1} UpdateG_t withdiscountfactor$
  end for
  for t from 0 to T **do**
    Compute the Advantage function:
    $A_t = G_t - V_\phi(s_t)$ {Using the critic network to estimate the state value}
    Update the Critic network:
    $\phi \leftarrow \phi + \alpha_\phi \cdot \nabla_\phi V_\phi(s_t) \cdot A_t$
    Update the Actor network using the log-likelihood gradient:
    $\theta \leftarrow \theta + \alpha_\theta \cdot \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot A_t$
  end for
end while

Each edge server's CPU demand, memory demand, hard disk demand and power consumption is concatenated into a feature vector and fed into a actor deep network, as well as a critic deep network. The actor outputs the probability of choosing each server, while the critic outputs the 'Q - value', the value of being in the state as given by the probability vector. The probability vector is sampled to choose the edge server to place the service in while the Q-value, along with the reward, is used for updation.

5.3. Communication among entities

Fig. 4 illustrates the simulation data flow among users, scheduler and AI module in the EdgeAISim. The scheduler receives requests from the user, processes them, and then chooses the appropriate AI module. Then the edge server is managed and the task is scheduled by the AI Module. Results from completed tasks are sent to the scheduler, which subsequently sends them to the user.

6. Experiments and evaluation

In our research, we employed a comprehensive approach to measure the power consumption of individual edge servers and the total power consumed at each time step in the edge computing system. This allowed us to gain valuable insights into the energy usage patterns of the entire system. By closely monitoring the power consumption of each edge server, we could pinpoint variations in energy usage across different resources, leading to a deeper understanding of the system's overall efficiency. Simultaneously, tracking the total power consumption over

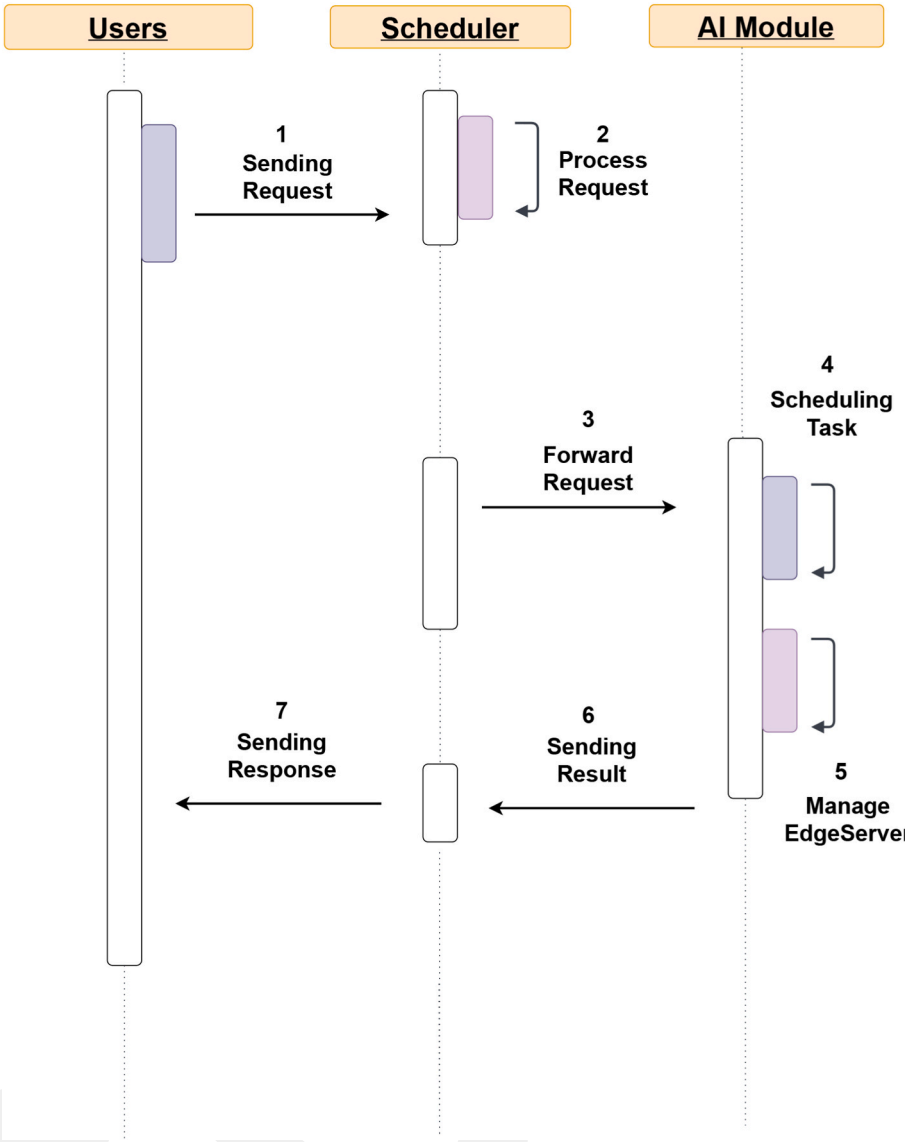

Fig. 4. Simulation data flow among users, scheduler and AI module.

time helped us analyze energy consumption trends and assess the system's performance in terms of power management and optimization strategies. This data-driven analysis formed the foundation for making informed decisions to improve energy efficiency and optimize the edge computing infrastructure.

6.1. Experimental setup

We utilized an Intel HexaCore i7 - 8750 H processor, which uses coffee lake architecture [37], with an NVIDIA 1050 Ti GPU with 16 GB of RAM and 4 GB Video RAM (VRAM), using Pascal architecture [38] on the Linux Operating System to run our simulations. We used the PyTorch library in order to implement the deep reinforcement learning algorithms. Table 4 shows the hyperparameters and their corresponding values, which are used for these experiments.

6.2. Workloads/dataset

We considered a scenario where there are 6 edge servers with the CPU, memory and disk capacities as specified in Table 2 and CPU, Memory and disk demands from pre-existing workloads as specified in Table 3.

Table 2
Server specifications.

Server	CPU	Memory	Disk
Edgeserver1	8	16384	131072
Edgeserver2	8	16384	131072
Edgeserver3	8	8192	131072
Edgeserver4	8	8192	131072
Edgeserver5	12	16384	131072
Edgeserver6	12	16384	131072

Table 3
Server demands.

Server	CPU Demand	Memory Demand	Disk Demand
Edgeserver1	0	0	0
Edgeserver2	0	0	0
Edgeserver3	0	0	0
Edgeserver4	0	0	0
Edgeserver5	1	1024	1017
Edgeserver6	0	0	0

Table 4
Hyperparameters and values.

Hyperparameter	Hyperparameter value
α (learning rate)	0.05
ϵ (exploration factor)	1
γ (Future reward weight factor)	0.9
ϵ decay (How much epsilon decreases)	0.997
Dropout probability in NN	0.5

6.3. Experimental results

This section presents the experimental results for different AI models (RL algorithms) such as Multi-Armed Bandit with Upper Confidence Bound, Deep Q-Networks, Deep Q-Networks with Graphical Neural Network, and Actor-Critic Network and baseline model using Worst-Fit Migration Algorithm.

6.3.1. Baseline model: worst-fit migration algorithm

Fig. 5 shows the power consumption for 6 servers, with total power consumption for the baseline which is worst-fit migration algorithm based resource management policy [30,31]. Total power consumption in the baseline worst fit migration rapidly increases from the start till it reaches about 6000 Watt (W). This is due to the algorithm allocating tasks to the server with the most resources available, without regard for power consumption.

6.3.2. Proposed model: Multi-Armed Bandit (MAB) with upper confidence bound (UCB)

Fig. 6 shows the power consumption for 6 servers, with total power consumption for MAB with UCB. The observed stagnation of power consumption at around 6000 W, which is comparable to the baseline, presents significant implications for the multi-arm bandit algorithm that was specifically designed to minimize power consumption through task migration. The algorithm's primary objective is to dynamically allocate computational tasks among multiple resources to achieve optimal power efficiency. However, the plateau in power consumption suggests that the current approach might have reached its limits in terms of further power reduction. This raises critical considerations for the algorithm's effectiveness and the potential need for reassessing its underlying assumptions or exploring alternative strategies. This stagnation shows a 0 or negligible percentage of improvement.

6.3.3. Proposed model: deep Q-networks

Fig. 7 shows the power consumption for 6 servers, with total power consumption for Deep Q-Networks. The observed minimization of total power consumption is a significant achievement for the Deep Q-Network algorithm designed to optimize power usage through task migration. The algorithm's ability to more prominently utilize the servers with lower power consumption, such as Edgeserver-1, suggests that it has successfully learned to make intelligent decisions regarding task allocation. By leveraging the power of deep reinforcement learning, the algorithm demonstrates its capacity to adaptively distribute tasks across servers, ultimately leading to reduced energy consumption. There is an initial high allocation by Q-Learning, which leads to a power consumption of 17500W, which however drops significantly to around 2500 W. This is an improvement of about 58 %, as compared to 6000 W in baseline.

6.3.4. Proposed model: deep Q-networks with graphical neural network (GNN)

Fig. 8 shows the power consumption for 6 servers, with total power consumption for Deep Q-Networks with GNN. The successful minimization of total power consumption and the clear trend of favoring servers with lower power consumption (EdgeServer-3 and EdgeServer-4) are remarkable outcomes for the Deep Q-Networks algorithm enhanced

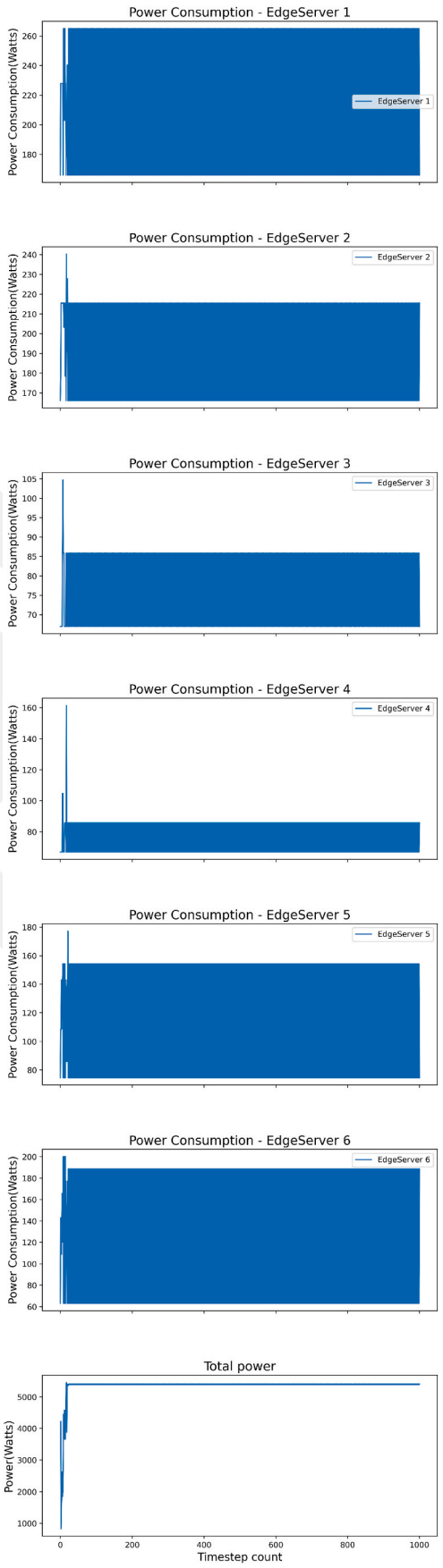

Fig. 5. Power consumption for 6 servers, with total power consumption for the worst-fit migration algorithm.

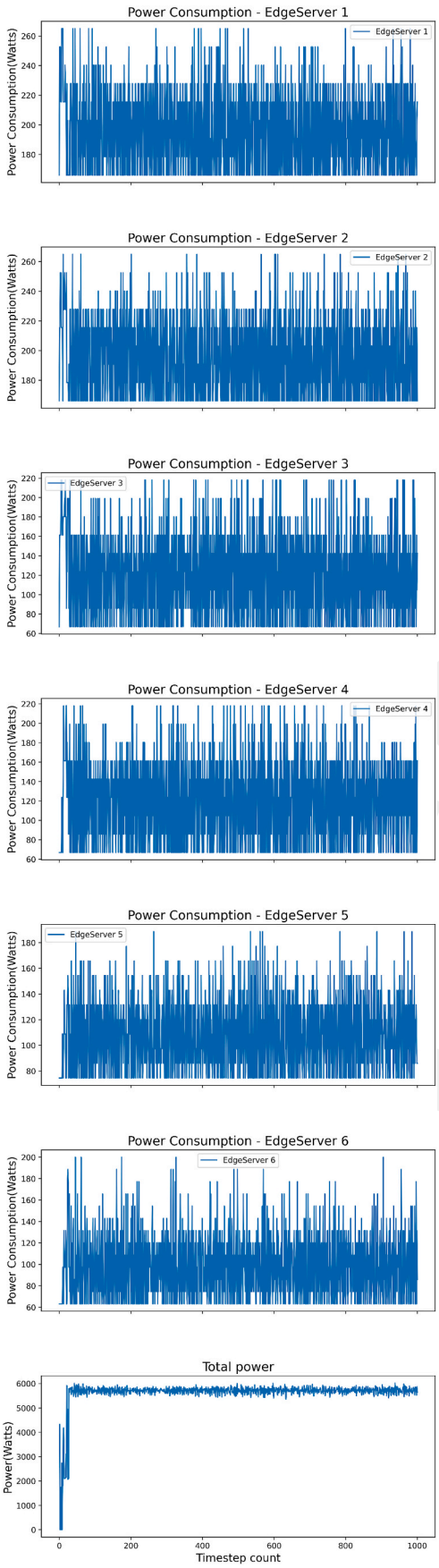

Fig. 6. Power consumption for 6 servers, with total power consumption for MAB with UCB.

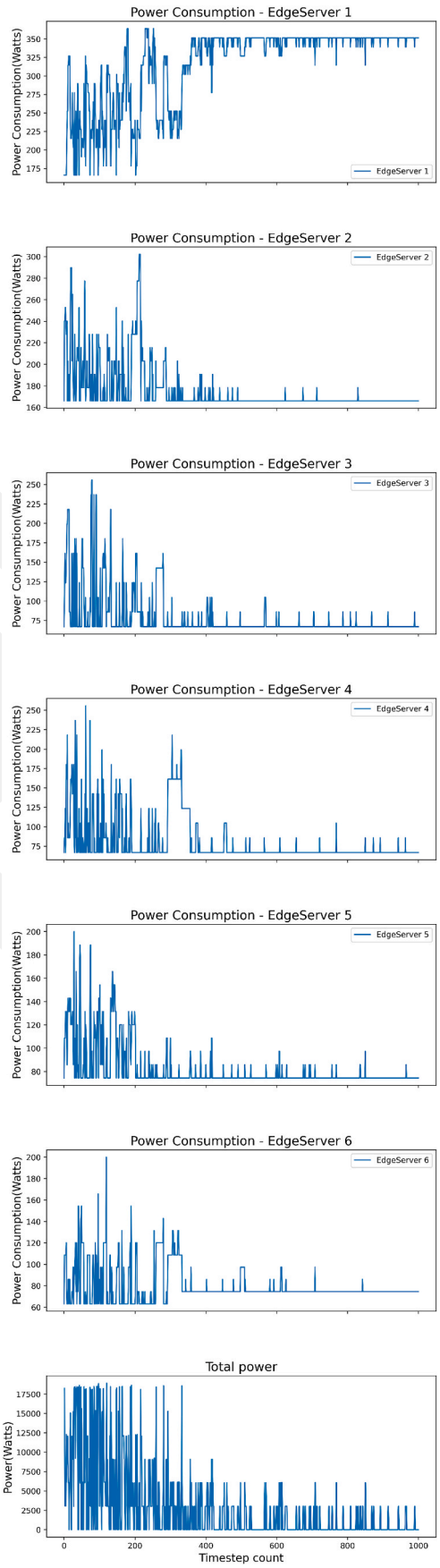

Fig. 7. Power consumption for 6 servers, with total power consumption for Deep Q-Networks.

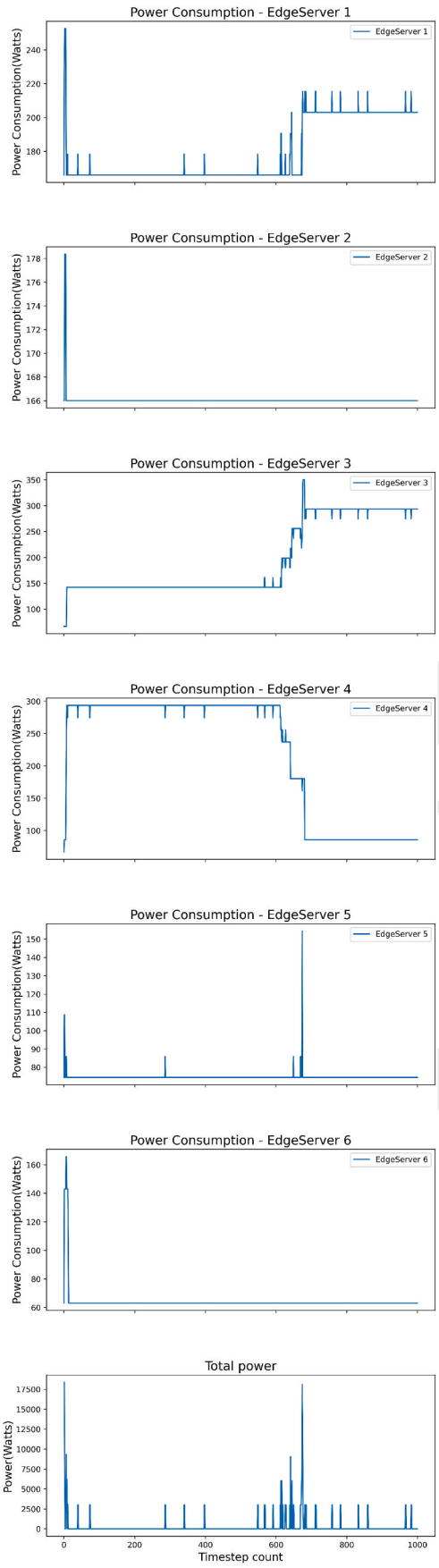

Fig. 8. Power consumption for 6 servers, with total power consumption for Deep Q-Networks with Graphical Neural Network (GNN).

with GNN to optimize power usage through task migration. The incorporation of GNN enables the algorithm to effectively capture and exploit the underlying relationships between edge servers, facilitating more informed and strategic task allocation decisions. By leveraging the power of GNN, the algorithm can better identify the most energy-efficient servers for task offloading, resulting in significant reductions in overall power consumption. Similar to the previous experiment, power consumption by Deep Q-Networks with GNN is initially high due to a large allocation of 17500 W; however, it drops dramatically to roughly 2500 W. When compared to a baseline output of 6000 W, this is an increase in efficiency of almost 58 %.

6.3.5. *Proposed model: actor-critic network*

Fig. 9 shows the power consumption for 6 servers, with total power consumption for a probabilistic Actor-Critic Network. The observed stagnation at a high power consumption level is a concerning outcome for the Actor-Critic network algorithm designed to minimize power consumption through task migration. Despite its potential for continuous action space exploration, the algorithm's inability to effectively learn indicates challenges in capturing the complex relationships between tasks and edge servers. As a result, it struggles to make informed decisions on task allocation, leading to sub-optimal power consumption levels. Similar to the MAB with UCB, this stagnation and wild fluctuation show a 0 or negligible percentage of improvement using the Actor-Critic Network.

6.4. *Performance comparison of EdgeAISim with baseline model*

Table 5 shows the comparison of AI models of EdgeAISim with baseline [30,31] in terms energy consumed on different servers. Notably, two AI models within EdgeAISim, Q-Learning and Q-Learning with GNN, demonstrate reduced overall power consumption in contrast to the baseline worst fit migration algorithm. This positive result can be credited to the integration of algorithmic improvements and the adoption of efficient application strategies embedded within these two models, which leverage reinforcement learning and graph-based machine learning methodologies. As a result, EdgeAISim has effectively presented substantial reductions in power usage within the realm of edge computing, thereby providing valuable insights and serving as a source of inspiration for the implementation of sustainable edge computing solutions on a broader scale.

7. Conclusions and future work

The surge in IoT-based applications has led to an unprecedented volume of data, posing significant challenges for traditional cloud computing due to issues such as latency, network traffic limitations, and security concerns. Edge computing offers a promising solution by bringing processing power and storage closer to the network's edge, reducing latency and alleviating network traffic issues. However, edge devices possess limited resources, necessitating efficient resource management with a focus on optimizing power consumption. In this paper, we proposed EdgeAISim, a novel framework that leverages AI models (especially reinforcement learning algorithms) and task migration strategies to address this critical concern in edge computing. By integrating advanced AI models such as Multi-Armed Bandit with Upper Confidence Bound, Deep Q-Networks, Deep Q-Networks with Graphical Neural Network, and Actor-Critic Network into the existing EdgeSimPy framework, we have successfully demonstrated the capability of EdgeAISim to significantly reduce power consumption in edge computing environments. In summary, our work showcases the potential of EdgeAISim as a powerful tool for enhancing the sustainability and efficiency of edge computing, offering tangible solutions to the pressing challenges posed by the ever-expanding IoT landscape.

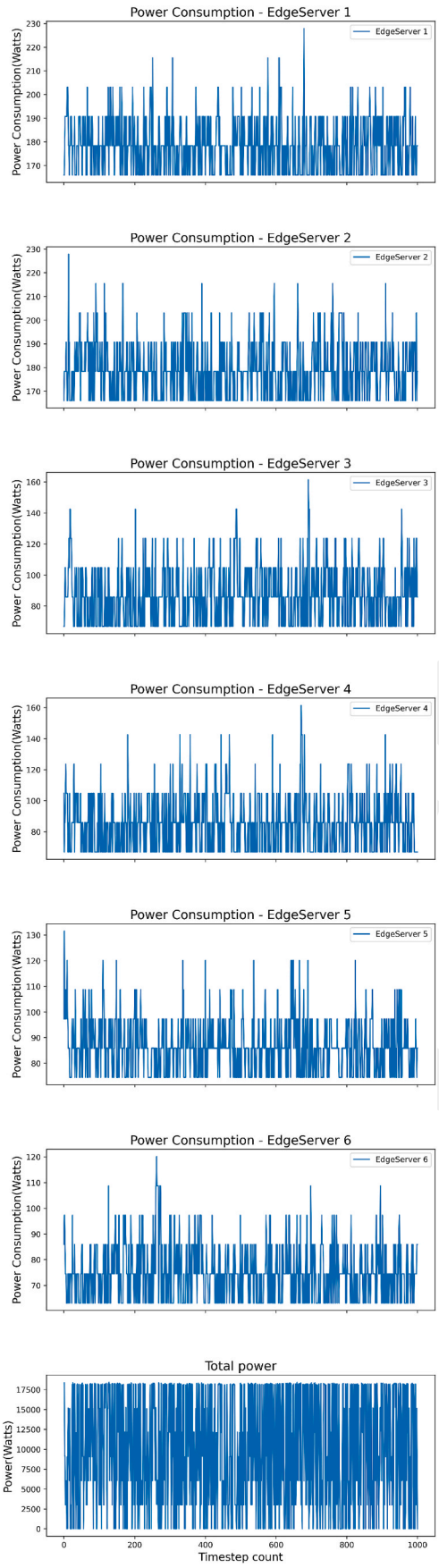

Fig. 9. Power consumption for 6 servers, with total power consumption for a probabilistic actor-critic network.

7.1. Promising future directions

While this work primarily focuses on mitigating energy consumption challenges in edge computing by proposing the basic version of EdgeAISim, it provides a solid foundation for several promising avenues of research and development. These future directions include.

7.1.1. IoT applications

Exploring how EdgeAISim can be tailored to specific IoT use cases will be valuable. Different IoT applications, from smart cities to healthcare, have unique demands, and adapting the framework to these requirements will be essential [5].

7.1.2. 6G networks and beyond

As next-generation networks like 6G networks become more prevalent, investigating how EdgeAISim can leverage the increased connectivity and bandwidth to further improve edge computing performance and efficiency is vital [10].

7.1.3. Security and privacy

Expanding the EdgeAISim framework to address security and privacy concerns in edge computing environments is essential. Developing mechanisms to safeguard sensitive data and ensure secure task migration will be crucial in future deployments [10].

7.1.4. Multi-objective optimization

Moving beyond energy consumption, researchers can explore multi-objective optimization using EdgeAISim that considers a balance between energy efficiency, QoS, and security, among other factors, to provide a holistic solution [5].

7.1.5. Edge device Heterogeneity

Investigating how EdgeAISim can adapt to diverse edge device capabilities and resource constraints will be necessary to accommodate the increasing variety of devices in edge environments [6].

7.1.6. Real-world deployment and validation

Finally, conducting real-world deployments and validations of EdgeAISim in various edge computing scenarios and industries will be a significant step toward practical implementation [6].

Incorporating these aspects into the development and enhancement of EdgeAISim will ensure its relevance and effectiveness in addressing the evolving challenges and opportunities in the dynamic field of edge computing.

Software availability

We released EdgeAISim available for free as open source. All code, datasets, and result reproducibility scripts are publicly available and can be accessed from GitHub: https://github.com/MuhammedGolec/EdgeAISim.

CRediT authorship contribution statement

Aadharsh Roshan Nandhakumar Conceptualization, Data curation, Investigation, Methodology, Software, Visualization, Validation, Formal analysis, Writing - original draft. Ayush Baranwal: Visualization, Validation, Formal analysis, Writing - original draft. Priyanshukumar Choudhary: Conceptualization, Data curation, Investigation, Methodology, Writing - original draft. Muhammed Golec: Conceptualization, Visualization, Validation, Formal analysis, Writing - original draft and Supervision. Sukhpal Singh Gill: Conceptualization, Data curation, Investigation, Methodology, Validation, Formal analysis, Writing - original draft and Supervision.

Table 5
Comparison of EdgeAISim with baseline in terms of Energy Consumption for different servers.

Works	Models	Edge Server 1 (W)	Edge Server 2 (W)	Edge Server 3 (W)	Edge Server 4 (W)	Edge Server 5 (W)	Edge Server 6 (W)	Total Power (W)
EdgeAISim	Q-Learning with GNN	181.4	166.24	184.5	200	81	63	1000
	Q-Learning	307	180	85.9	91.25	90.5	82.4	4875
	Actor-Critic	236.5	188.5	94.4	96	81.9	79.6	12850
	MAB-UCB	217.5	227	163	161.5	128.6	129	5600
Baseline [30, 32]]	Worst fit	220	192.57	77.85	80.85	118.5	125	5431

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.

Data availability

The data that support the findings of this study are openly available at https://github.com/MuhammedGolec/EdgeAISIM.

Acknowledgements

The research investigation was carried out during the internships undertaken by Aadharsh Roshan Nandhakumar, Ayush Baranwal, and Priyanshukumar Choudhary at Queen Mary University of London, UK. Muhammed Golec would express his thanks to the Ministry of Education of the Turkish Republic, for their support and funding.

References

[1] P. Cruz, N. Achir, A.C. Viana, On the edge of the deployment: a survey on multi-access edge computing, ACM Comput. Surv. 55 (5) (2022) 1–34.
[2] S.S. Nabavi, et al., Tractor: traffic-aware and power-efficient virtual machine placement in edge-cloud data centers using artificial bee colony optimization, Int. J. Commun. Syst. 35 (1) (2022) e4747.
[3] M.S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, F. Hussain, Machine learning at the network edge: a survey, ACM Comput. Surv. 54 (8) (2021) 1–37.
[4] H. Hua, Y. Li, T. Wang, N. Dong, W. Li, J. Cao, Edge computing with artificial intelligence: a machine learning perspective, ACM Comput. Surv. 55 (9) (2023) 1–35.
[5] R. Singh, S.S. Gill, Edge ai: a survey, Internet of Things and Cyber-Physical Systems 3 (2023) 71–92.
[6] S. Iftikhar, et al., Ai-based Fog and Edge Computing: A Systematic Review, Taxonomy and Future Directions, Internet of Things, 2022, 100674.
[7] J. Du, et al., Computation energy efficiency maximization for intelligent reflective surface-aided wireless powered mobile edge computing, IEEE Transactions on Sustainable Computing (1) (2023) 1–15.
[8] H. Jiang, X. Dai, Z. Xiao, A.K. Iyengar, Joint Task Offloading and Resource Allocation for Energy-Constrained Mobile Edge Computing, IEEE Transactions on Mobile Computing, 2022.
[9] S. Nabavi, et al., Seagull optimization algorithm based multi-objective vm placement in edge-cloud data centers, Internet of Things and Cyber-Physical Systems 3 (2023) 28–36.
[10] S.S. Gill, M. Xu, C. Ottaviani, P. Patros, R. Bahsoon, A. Shaghaghi, M. Golec, V. Stankovski, H. Wu, A. Abraham, et al., Ai for Next Generation Computing: Emerging Trends and Future Directions, vol. 19, Internet of Things, 2022, 100514.
[11] M.S. Aslanpour, et al., Serverless edge computing: vision and challenges, in: Proceedings of the 2021 Australasian Computer Science Week Multiconference, 2021, pp. 1–10.
[12] S. Ghafouri, et al., Mobile-kube: mobility-aware and energy-efficient service orchestration on kubernetes edge servers, in: 2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC), IEEE, 2022, pp. 82–91.
[13] M.S. Aslanpour, et al., Performance Evaluation Metrics for Cloud, Fog and Edge Computing: A Review, Taxonomy, Benchmarks and Standards for Future Research, vol. 12, Internet of Things, 2020, 100273.
[14] R. Mahmud, S. Pallewatta, M. Goudarzi, R. Buyya, ifogsim2: an extended ifogsim simulator for mobility, clustering, and microservice management in edge and fog computing environments, J. Syst. Software 190 (2022), 111351.
[15] C. Sonmez, A. Ozgovde, C. Ersoy, Edgecloudsim: an Environment for Performance Evaluation of Edge Computing Systems, 2017, pp. 39–44.
[16] H. Gupta, A. Vahid Dastjerdi, S.K. Ghosh, R. Buyya, ifogsim: a toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments, Software Pract. Ex. 47 (9) (2017) 1275–1296.
[17] P.S. Souza, T. Ferreto, R.N. Calheiros, Edgesimpy: Python-based modeling and simulation of edge computing resource management policies, Future Generat. Comput. Syst. 148 (2023) 446–459.
[18] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software Pract. Ex. 41 (1) (2011) 23–50.
[19] X. Zeng, S.K. Garg, P. Strazdins, P.P. Jayaraman, D. Georgakopoulos, R. Ranjan, Iotsim: a simulator for analysing iot applications, J. Syst. Architect. 72 (2017) 93–107. Design Automation for Embedded Ubiquitous Computing Systems.
[20] D.N. Jha, K. Alwasel, A. Alshoshan, X. Huang, R.K. Naha, S.K. Battula, S. Garg, D. Puthal, P. James, A.Y. Zomaya, S. Dustdar, R. Ranjan, Iotsim-edge: A Simulation Framework for Modeling the Behaviour of Iot and Edge Computing Environments, 2019.
[21] K. Alwasel, D.N. Jha, F. Habeeb, U. Demirbaga, O. Rana, T. Baker, S. Dustdar, M. Villari, P. James, E. Solaiman, et al., Iotsim-osmosis: a framework for modeling and simulating iot applications over an edge-cloud continuum, J. Syst. Architect. 116 (2021), 101956.
[22] C. Wang, R. Li, W. Li, C. Qiu, X. Wang, Simedgeintel: a open-source simulation platform for resource management in edge intelligence, J. Syst. Architect. 115 (2021), 102016.
[23] T. Qayyum, A.W. Malik, M.A. Khan Khattak, O. Khalid, S.U. Khan, Fognetsim++: a toolkit for modeling and simulation of distributed fog environment, IEEE Access 6 (2018) 63570–63583.
[24] I. Lera, C. Guerrero, C. Juiz, Yafs: a simulator for iot scenarios in fog computing, IEEE Access 7 (2019) 91745–91758.
[25] A. Saleh, P. Joshi, R.S. Rathore, S.S. Sengar, Trust-aware routing mechanism through an edge node for iot-enabled sensor networks, Sensors 22 (20) (2022) 7820.
[26] C. Zhang, Z. Zheng, Task migration for mobile edge computing using deep reinforcement learning, Future Generat. Comput. Syst. 96 (2019) 111–118.
[27] Z. Liang, Y. Liu, T.-M. Lok, K. Huang, Multi-cell mobile edge computing: joint service migration and resource allocation, IEEE Trans. Wireless Commun. 20 (9) (2021) 5898–5912.
[28] F. Tang, C. Liu, K. Li, Z. Tang, K. Li, Task migration optimization for guaranteeing delay deadline with mobility consideration in mobile edge computing, J. Syst. Architect. 112 (2021), 101849.
[29] M. Hosaagrahara, H. Sethu, Max-min fair scheduling in input-queued switches, IEEE Trans. Parallel Distr. Syst. 19 (4) (2008) 462–475.
[30] P. Lai, Q. He, J. Grundy, F. Chen, M. Abdelrazek, J. Hosking, Y. Yang, Cost-effective app user allocation in an edge computing environment, IEEE Transactions on Cloud Computing 10 (3) (2020) 1701–1713.
[31] X. Xu, H. Cao, Q. Geng, X. Liu, F. Dai, C. Wang, Dynamic resource provisioning for workflow scheduling under uncertainty in edge computing environment, Concurrency Comput. Pract. Ex. 34 (14) (2022) e5674.
[32] A. Slivkins, Introduction to Multi-Armed Bandits, 2022.
[33] A. Carpentier, A. Lazaric, M. Ghavamzadeh, R. Munos, P. Auer, A. Antos, Upper-confidence-bound Algorithms for Active Learning in Multi-Armed Bandits, 2015.
[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing Atari with Deep Reinforcement Learning, 2013.
[35] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: a review of methods and applications, AI Open 1 (2020) 57–81.
[36] I. Grondman, L. Busoniu, G.A.D. Lopes, R. Babuska, A survey of actor-critic reinforcement learning: standard and natural policy gradients, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42 (6) (2012) 1291–1307.
[37] A. Abel, J. Reineke, uops. info: characterizing latency, throughput, and port usage of instructions on intel microarchitectures, in: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, 2019, pp. 673–686.
[38] F. Lombardi, A. Muti, L. Aniello, R. Baldoni, S. Bonomi, L. Querzoni, Pascal: an architecture for proactive auto-scaling of distributed services, Future Generat. Comput. Syst. 98 (2019) 342–361.