Journal of Information Systems Engineering and Management
2025, 10(55s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Performance Evaluation and Optimization of Dynamic Resource Allocation and Task Offloading for Mobile Edge Computing using Deep Reinforcement Learning

Shilpi Kaura[1], Nupa Ram Chauhan[2]
[1]*Computer Science and Engineering*
Teerthanker Mahaveer University, Moradabad, UP, India.
[2]*Computer Science and Engineering*
Teerthanker Mahaveer University, Moradabad, UP, India.

ARTICLE INFO

Received: 05 Jan 2025

Revised: 20 Feb 2025

Accepted: 10 Mar 2025

ABSTRACT

Resource-efficient and low-latency applications are made possible by the revolutionary paradigm known as Multi-Access Edge Computing (MEC), which places computational resources closer to the end users. Dynamic resource allocation and task offloading are essential for guaranteeing optimal performance since MEC systems enable a wide range of computationally demanding and time-sensitive applications. However, a number of obstacles, including network congestion, a lack of processing capacity, and fluctuating user demands, make it difficult to manage these resources efficiently in real-time. In this regard, the intricate trade-offs between latency, energy efficiency, computational resources, and quality of service (QoS) can be effectively addressed by multi-objective optimization (MOO) and deep reinforcement learning (DRL). In order to optimize resource allocation and work offloading, this research investigates the use of MOO and DRL in MEC systems. In particular, we suggest a hybrid framework that uses DRL for adaptive, real-time decision-making and multi-objective optimization to balance conflicting objectives. The study offers a thorough model, simulations, and findings that show how well our strategy works to enhance system performance in a variety of scenarios. This study makes two contributions: first, it presents a new method for dynamic resource management and job offloading in MEC systems; second, it shows that combining MOO and DRL in practical applications is feasible and has potential advantages. Improved system performance, energy efficiency, and user happiness are anticipated results, which would represent a major step toward the creation of effective, scalable MEC settings.

Keywords: Multi-Access Edge Computing (MEC), network congestion, quality of service (QoS), by multi-objective optimization (MOO), deep reinforcement learning (DRL).

1. Introduction

1.1 Background

In the context of 5G and beyond, Multi-Access Edge Computing (MEC) is a crucial enabler of the upcoming generation of mobile networks. Low-latency and high-throughput applications like augmented reality, driverless cars, and real-time data analytics are made possible by MEC, which moves data storage and processing power to the network's edge, closer to end users. By minimizing the need for centralized cloud data centers, this edge computing architecture improves performance for applications that are sensitive to latency and eases network congestion. Task offloading and dynamic resource allocation are essential components of MEC systems that guarantee effective system operation. It is crucial to control how and when computational tasks are offloaded to edge nodes or cloud servers because edge devices—like mobile users, IoT sensors, and drones—generate vast amounts of data and need a significant amount of processing power. To maximize performance and preserve Quality of Service (QoS) for consumers, this dynamic process necessitates efficient management of finite resources, such as CPU power, bandwidth, and battery life. Task offloading can improve system efficiency and drastically lower the computing load on mobile devices, but it also makes decision-making more difficult, particularly in real-time systems with changing workloads.

1.2 Problem Statement

Balancing the trade-offs between processing resources, energy efficiency, job offloading, and user quality of service is one of the main issues in MEC systems. The effective management of resources is made more difficult by the edge devices' limited computational capacity, changing network conditions, and different task needs. In order to minimize energy consumption, communication delays, and resource waste, tasks must be offloaded to the proper edge node or cloud server using dynamic resource allocation and task offloading. Furthermore, it is challenging to come up with a globally ideal solution that functions in every situation since real-time decision-making is necessary to adjust to shifting network conditions, device statuses, and user requests.

As MEC environments become more heterogeneous—where several devices with varying capabilities, needs, and network circumstances must cooperate to guarantee optimal system performance—the complexity of these issues rises. Therefore, the success of contemporary mobile and Internet of Things applications depends on an effective solution for dynamic resource allocation and task offloading in MEC systems.

1.3 Research Objectives and Contributions

Through the use of two cutting-edge methodologies, Multi-Objective Optimization (MOO) and Deep Reinforcement Learning (DRL), this study seeks to address the difficulties associated with dynamic resource allocation and task offloading in MEC systems.

This work's main goals are to:

- Provide a novel method for dynamic resource allocation that balances trade-offs between several objectives, such as energy consumption, computing load, latency, and quality of service, by using MOO.
- To use DRL for adaptive task offloading choices, which would allow for real-time learning

and decision-making in response to shifting user needs and system conditions.

This study offers two contributions:

1. **Increased Task Offloading Efficiency:** To increase the effectiveness of task offloading in MEC systems, we suggest a hybrid framework that combines MOO and DRL. This method optimizes workload distribution among edge nodes and cloud servers by dynamically adjusting to changing network circumstances and device capabilities.
2. **Optimised Resource Utilization**: Our framework maximizes resource utilization and minimizes waste by considering several objectives at once, which improves energy efficiency and performance in MEC situations.

2. Related Work

2.1 Multi-Objective Optimization in MEC Systems

Mobile Edge Computing (MEC) systems have made extensive use of Multi-Objective Optimization (MOO) to handle the intricate trade-offs between multiple performance goals, including resource utilization, energy consumption, latency, and throughput. These goals frequently clash, necessitating the simultaneous consideration of several while allocating resources and delegating tasks. When no objective can be enhanced without making another worse, MOO approaches seek to discover Pareto-optimal solutions that achieve the best possible balance between these competing goals.

Some key methods used in MEC systems include:

- **Pareto Efficiency:** In multi-objective optimization, Pareto optimality is frequently used to depict solutions in which enhancing one goal would impair the others. This method aids in determining the optimal resource allocation trade-offs in MEC systems, such as striking a balance between energy conservation and latency reduction. Pareto-based techniques are commonly used to find non-dominated solutions that satisfy a range of user needs.
- **Genetic Algorithms (GAs)**:MEC has made substantial use of genetic algorithms to solve multi-objective optimization problems. GAs employ a population-based search strategy, applying crossover, mutation, and

selection operators to iterate through several generations of solutions. These algorithms' capacity to converge toward Pareto-optimal solutions and explore wide search spaces makes them ideal for MEC systems.

- **Multi-Objective Evolutionary Strategies (MOES):** Complex optimization issues in MEC contexts have been solved using evolutionary techniques, including multi-objective variations. These methods, which draw inspiration from natural evolutionary processes, make it possible to explore solution spaces effectively, particularly in dynamic and diverse situations. Large-scale optimization issues with several competing goals are especially well-suited for MOES.

These MOO techniques help in optimizing resource allocation in MEC, ensuring that both the mobile devices and edge nodes can operate efficiently under varying conditions.

2.2 Task Offloading in MEC

A crucial component of MEC systems is task offloading, which lowers latency and boosts performance by transferring computational activities produced by user devices to neighboring edge nodes or the cloud. To make the best choices, offloading strategies must take into account variables including latency, energy usage, processing power, and network circumstances.

The following are a few of the current task offloading models:

• **Latency-Driven Offloading:** This ensures real-time performance in latency-sensitive applications by offloading jobs to edge nodes with the least amount of delay. By utilizing edge computing capabilities, strategies aim to minimize network latency by choosing the edge node that is nearest to the device. Decisions on offloading depend on the availability of edge nodes, task characteristics, and network conditions.

• **Energy-Aware Offloading:** In mobile edge contexts, energy consumption is a crucial limitation. By shifting computationally demanding tasks to edge nodes or the cloud, energy-efficient task offloading techniques reduce the amount of energy used by mobile devices with limited power. By considering variables like processor power and battery levels, these tactics seek to strike a balance between work offloading and energy consumption.

• **Resource-Aware Offloading:** Decisions about offloading in MEC environments with limited resources are determined by the computing and storage capacity at edge nodes. Resource-aware offloading techniques make sure that tasks are only released when the edge node has enough resources to complete them effectively. These models seek to prevent resource overloads at edge nodes while maximizing system performance overall.

2.3 Deep Reinforcement Learning (DRL) in MEC Systems

In MEC systems, Deep Reinforcement Learning (DRL) has become a potent method for resolving dynamic decision-making issues. Through interactions with their surroundings, agents can learn optimal policies thanks to DRL algorithms, which use incentives and penalties to direct the learning process. DRL is very helpful in MEC systems for responding to real-time variations in user needs, resource availability, and network circumstances.

Several applications of DRL in MEC include:

• **Dynamic Resource Allocation:** By teaching models to make decisions in real time depending on the system's present state, DRL has been utilized to improve resource allocation. By dynamically allocating processing resources to edge nodes, these systems optimize throughput, latency, and energy usage. For instance, when jobs are transferred between edge nodes and cloud servers or offloaded, DRL algorithms can determine the optimal course of action for resource allocation.

• **Task Offloading:** DRL has been used to solve task offloading issues where the system chooses which jobs to offload to the cloud or edge and when to do so. Based on variables such as job size, computing demands, energy consumption, and network circumstances, the agent learns the best offloading strategy. This lowers latency and improves QoS for MEC systems.

• **Adaptive Learning and Optimization:** DRL gives MEC systems the ability to continuously adjust and improve resource management choices in response to shifting circumstances, including user requests, network congestion, and varying workloads. In contrast to conventional optimization methods, DRL can modify its approach in real time, producing more adaptable and effective results.

2.4 Gaps in Existing Research

The use of MOO and DRL for job offloading and resource allocation in MEC systems has advanced significantly, although there are still a number of holes in the body of research:

• **Absence of Integrated Methods**: Few studies have looked into how to combine the advantages of both approaches; most current approaches either concentrate on MOO or DRL separately. DRL is skilled at making decisions in real time and learning from changing conditions, whereas MOO is excellent at identifying the best trade-offs between several goals. Although this field is still unexplored, combining these two approaches could enhance resource management in MEC systems.

• **Scalability and Generalization Problems:** When used in large-scale MEC systems with numerous devices and edge nodes interacting, DRL-based techniques frequently have scalability problems. The real-world applicability of current DRL models may be limited by their poor generalization to various network settings and user behaviors. DRL models that scale well and adjust to various MEC contexts are required.

• **Real-Time Performance and Stability:** A lot of DRL solutions for MEC aren't completely tuned for real-time performance, which makes them unsuitable for applications that need to be completed quickly. Faster, more reliable learning algorithms are required to enable nearly immediate decision-making in MEC systems, as DRL models can likewise have sluggish convergence.

• **Trade-offs between energy and latency:** The majority of current research concentrates on either lowering latency or minimizing energy usage, but it ignores the optimization of both goals at the same time. To create multi-objective models that can concurrently optimize energy, latency, and other pertinent performance measures, further effort is needed.

3. System Model and Problem Formulation

3.1 MEC System Architecture

The MEC system architecture is made up of a number of essential parts that cooperate to give mobile devices (MDs) connectivity, storage, and processing power. The system's primary components are as follows:

- **Mobile Devices (MDs):** These are user devices that produce computational jobs, including smartphones, drones, Internet of Things sensors, and other edge devices. Task offloading to more capable resources at the edge or cloud is necessary because they are usually resource-constrained in terms of processor power, battery capacity, and network connectivity.
- **Edge Nodes (ENs):**The computational resources near the MDs at the network's edge are known as Edge Nodes (ENs). Edge nodes, which offer low-latency compute and storage capabilities, can be a component of a distributed infrastructure, such as base stations, tiny cells, or edge servers. They have more potent computational resources than MDs and can complete the duties that MDs have delegated to them.
- **Cloud Servers:** When computational or storage demands surpass edge nodes' capacity, certain tasks may be transferred to centralized cloud servers in addition to edge nodes. Because of their distance from end users, cloud servers have higher latency even though they have more processing capability.
- **Communication Channels**: Wireless communication channels, which can differ in bandwidth, latency, and dependability, are used for communication between MDs, edge nodes, and cloud servers. Data transfer for feedback mechanisms, control signals, and task offloading is supported by the communication infrastructure.

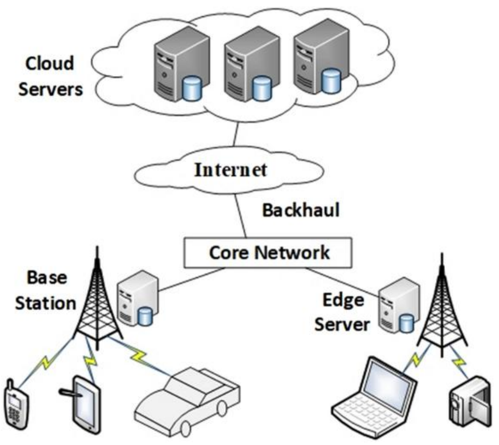

Fig1: The MEC system architecture

Assumptions and Constraints:

• **Bandwidth:** Geographical distance, network congestion, and interference can all affect the available bandwidth between MDs and edge nodes or cloud servers. This has an impact on how long it takes to offload duties and how much energy MDs use while doing so.

• **Latency:** For real-time applications in particular, latency is a crucial performance measure for MEC systems. Decisions about task offloading must take into consideration the processing times and communication lags related to offloading to cloud servers or edge nodes.

• **Energy Consumption**: Since most mobile devices run on batteries, energy efficiency is a key consideration. MDs use less energy when activities are offloaded, but excessive offloading or offloading to distant servers may result in higher energy consumption because of transmission costs.

• **Computational Resources:** To avoid overloading the limited computational resources of edge nodes and cloud servers, job offloading needs to be controlled. The appropriate distribution of resources throughout the system must be guaranteed by the task offloading mechanism.

3.2 Task Offloading and Resource Allocation Model

A multi-objective optimization problem can be used to describe the task offloading and resource allocation issue in MEC systems. The purpose is to maximize resource allocation among MDs, edge nodes, and cloud servers while taking a number of performance measures into account. The main goals are to maximize job completion efficiency while decreasing latency, energy consumption, and resource use.

Let's define the problem in a formal manner:

• **Objective 1 (delay Minimization):** Reduce the overall delay that the offloading procedure incurs. Latency comprises processing time at the offloading destination, task transmission time from MDs to edge nodes/cloud, and any additional network delays.

$$Minimize\ L = \sum_{i=0}^{n} (Transmission\ Time_{\ i} + Processing\ \ Time_{\ i})$$

where N is the number of tasks, L is the latency for offloading task i, and the sum includes both transmission and processing time components.

• **Objective 2 (Energy Consumption Minimization):** Reduce the amount of energy used by mobile devices for communication and task offloading. The distance to the cloud or edge node, the bandwidth needed for transmission, and the energy needed for task processing all affect this.

$$Minimize\ E = \sum_{i=0}^{N} (Energy\ for\ Transmission_i + Energy\ for\ processing\ _i)$$

Where E represents the total energy consumption across all tasks, and each task's energy consumption depends on communication and computational requirements.

• **Objective3 (Resource Utilization Maximization):** Make the most effective use of the computational resources at cloud servers and edge nodes by allocating tasks to nodes according to processing power and resource availability. This guarantees that resources are not overworked or underutilized.

$$Maximize\ R = \sum_{j=1}^{N} \left(Resource\ Utilization_j\right)$$

where R represents the resource utilization across M edge nodes, with each edge node providing computational resources for task processing.

The combined multi-objective problem can be formulated as:

$$Minimize\ \{\ L, E\ \}, Maximize\ R$$

Subject toconstraints such as:

- Latency and bandwidth restrictions for MD-to-edge/cloud node connection.
- Limitations on the computational capabilities of cloud servers and edge nodes' available processing power.
- MDs, which are usually battery-powered, have energy limitations.

Finding a Pareto-optimal solution that strikes a compromise between these goals is the aim in order to make sure that the MEC system manages resource allocation and task offloading effectively and efficiently.

3.3 DRL Framework for Decision Making

We suggest applying Deep Reinforcement Learning (DRL) for adaptive decision-making to tackle the task offloading and resource allocation problem, which is dynamic and complex. DRL models work best in real-time systems where decisions must be made constantly in response to changing user needs and the system's present state.

• **DRL Model Overview:**

- **Q-learning:** By interacting with the environment, an agent can learn optimum policies using traditional Q-learning techniques, which do not require a model. To enhance decision-making over time, Q-learning entails updating the action-value function and assessing the worth of actions performed in particular stages.
- **Deep Q-Networks (DQN):**A neural network is used to approximate the Q-values in deep Q-Networks (DQN), a deep learning-based extension of Q-learning that enables the agent to handle huge state and action spaces that are otherwise challenging to manage with tabular Q-learning. DQN has been effectively used to solve a number of decision-making issues, such as task offloading and MEC resource distribution.

Definition of State, Action, and Reward:

- **State (s):** The system's current configuration, comprising the state of cloud servers, edge nodes, and MDs, is represented by the state ss. Task requirements (e.g., size, computing demands), MDs' current energy levels, available resources at edge nodes, network circumstances, and latency are some of the details it contains.
- **Action (a):** The DRL agent's choice is represented by the action a. This decision may include deciding whether to offload a job to the cloud or a nearby edge node, picking the target edge node or cloud server, and determining when to offload.
- **Reward (r):** The agent receives a scalar value as the reward (r) depending on how well the action was executed. The system's objectives are used to determine the reward, taking latency, energy consumption, and resource use into account. For instance:

$$\mathrm{r} = -\alpha\mathrm{L} - \beta\mathrm{E} + \gamma\mathrm{R}$$

Journal of Information Systems Engineering and Management
2025, 10(55s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

where α, β, and γ are weight factors for latency, energy consumption, and resource utilization, respectively. A positive reward encourages actions that minimize latency and energy consumption while maximizing resource utilization.

4. Multi-Objective Optimization for Dynamic Resource Allocation

4.1 Objective Functions

Task offloading and dynamic resource allocation in MEC systems necessitate the optimization of several competing goals. Among other things, the primary goals usually include optimizing throughput, decreasing latency, and limiting energy use. In order to balance the trade-offs between system performance, resource usage, and user experience, each goal is essential to the MEC system's smooth operation.

1. Energy Consumption Minimization (Objective 1): Because mobile devices (MDs) are frequently battery-powered, MEC systems are particularly concerned with the energy consumption of MDs and edge nodes. The computing power needed for job processing, the communication distance, and network conditions are some of the variables that affect a device's energy consumption. Offloading computational work to the edge or cloud as required while taking processing and communication energy usage into account is the aim in order to reduce energy consumption.

Mathematical Formulation:

$$Minimize\ E_{MD} = \sum_{i=0}^{n} Transmission\ Time_{\ i} + Processing\ \ Time_{\ i}$$

Where:

- E_{MD} is the total energy consumed by MDs,
- Energy for Transmission$_i$ is the energy required for sending task *i* from the MD to an edge node or cloud,
- Energy for Processing$_i$ is the energy required to process task *i* at the offloading destination (edge node or cloud).

2. **Minimization of Latency (Objective 2):** In real-time applications, latency is a crucial component. Improving system performance requires reducing the amount of time needed to offload operations and finish calculations. Transmission time, edge processing time, and network congestion are some of the variables that affect latency.

   Mathematical Formulation:

$$Minimize\ L = \sum_{i=0}^{n} Transmission\ Time_{\ i} + Processing\ \ Time_{\ i}$$

   Where:

   - Lis the total latency incurred during task offloading and processing,
   - Transmission Time$_i$ is the time it takes to transmit task *i* to the offloading node,
   - Processing Time$_i$ is the time required for the offloading node (edge/cloud) to process the task.

3. **Throughput Maximization (Objective 3):** The system's capacity to complete tasks in a specified amount of time is referred to as throughput. Increased throughput suggests that the MEC system is using resources and processing jobs effectively. This goal is to make sure that job offloading is done effectively and that edge nodes and cloud servers are used to their full potential.

   Mathematical Formulation:

$$Maximize\ T = \sum_{j=1}^{M} \left(Resource\ Utilization_j\right)$$

Where:

- T is the total throughput, or the number of tasks processed per unit of time across all edge nodes,
- Resource Utilization$_j$ is the utilization of computational resources at edge node j.

Improvements in one objective (like reducing energy use) may result in deterioration in other goals (like delay). This is because these goals frequently clash. Multi-objective optimization is crucial for dynamic resource allocation and job offloading in MEC systems since it presents a challenge in determining the optimal trade-off between these conflicting goals.

4.2 Optimization Techniques

Different optimization algorithms are employed to address the multi-objective nature of resource allocation and task offloading in MEC systems. When no single goal can be enhanced without sacrificing another, these methods seek to identify a set of Pareto-optimal solutions.

1. Genetic Algorithms (GAs): Multi-objective optimization problems are solved by genetic algorithms, a class of evolutionary algorithms. They work with a population of solutions, exploring the solution space through processes including crossover, mutation, and selection. GAs can be used to investigate different job offloading tactics and resource allocation choices in the context of MEC systems, resulting in a set of solutions that strike a balance between throughput, latency, and energy consumption.

GAs have the following advantages:

- They are adaptable to changing conditions and can manage big, complicated search spaces.
- Non-linear optimization problems, like those in MEC systems, are ideally suited for GAs.

2. Methods Based on Pareto: Finding a collection of non-dominated solutions, or the Pareto front, where no aim can be improved without making another worse is the main goal of Pareto-based approaches. These techniques are applied in multi-objective optimization to produce a collection of Pareto-optimal solutions that provide trade-offs between competing goals. Pareto-based techniques make it possible to find solutions for MEC systems that effectively balance throughput, energy consumption, and latency.

Pareto-based methods have the following benefits:

- They clearly illustrate how objectives are traded off.
- They guarantee that the ultimate solution is the best possible for all goals.
- The approach enables decision-makers to choose a solution according to their inclinations for goal trade-offs.

3. Particle Swarm Optimization (PSO): This evolutionary method was also influenced by the social behavior of fish schools and flocks of birds. By modeling a swarm of particles (solutions) that navigate the solution space according to their own and their neighbors' experiences, it can be applied to multi-objective optimization in MEC systems. The swarm as a whole converges toward the ideal set of solutions, with each particle's position representing a potential solution.

PSO has the following benefits:

- It is fast to converge and computationally efficient.
- It works especially well for issues involving continuous objective functions, like resource allocation in MEC.

4. Hybrid Approaches: Multiple optimization techniques are used in hybrid methods to overcome individual shortcomings and capitalize on their strengths. In dynamic and complicated contexts, for instance, the convergence speed and quality of the Pareto-optimal solutions can be improved by combining genetic algorithms with local search strategies or by employing reinforcement learning to direct the search process.

Advantages of Hybrid Methods:

- The ability to handle both discrete and continuous decision variables is one of the benefits of hybrid methods.
- By combining several strategies, they frequently offer a more complete solution.

4.3 Simulation of Multi-Objective Optimization in MEC

A number of simulations are run in order to assess how well the multi-objective optimization techniques for job offloading and resource allocation in MEC systems work. In the trials, the trade-offs between throughput, energy usage, and delay are evaluated using several optimization strategies.

Experimental Setup:

• Simulation parameters:

- A network of cloud servers, edge nodes, and mobile devices (MDs) with different processing capabilities, energy consumption patterns, and communication bandwidths are included in the simulation.
- Transmission distance and task complexity are used to model latency and energy expenditure. The resource usage of cloud servers and edge nodes determines throughput.

• Optimization Algorithms:

- To address the multi-objective optimization problem, Particle Swarm Optimization (PSO), Pareto-based techniques, and genetic algorithms (GA) are used.
- The quantity of Pareto-optimal solutions discovered and the caliber of trade-offs made are used to compare the performance of each algorithm.

Performance Metrics:

- Trade-off between Energy and Latency: When energy consumption is kept to a minimum, latency tends to rise since longer communication lengths are required. On the other hand, optimizing for low latency may result in increased energy usage.
- Throughput Maximization: Especially in situations with high task volumes, throughput is greatly increased by optimizing resource allocation.

5. Deep Reinforcement Learning for Task Offloading and Resource Allocation

5.1 Deep Reinforcement Learning Overview

A branch of machine learning called deep reinforcement learning (DRL) blends deep learning methods with reinforcement learning (RL). In complicated, high-dimensional situations where the agent must learn to respond in a way that maximizes cumulative rewards over time, it works especially well for dynamic decision-making. DRL enables agents to engage with the environment and learn from the results of their actions through trial and error, in contrast to traditional machine learning models that are usually trained on fixed datasets.

DRL is important in MEC systems because it offers a way to make adaptive, real-time decisions about resource allocation and task offloading. In MEC contexts, where network circumstances, energy limits, and compute resources are ever-changing, this is crucial. DRL is perfect for handling dynamic and unpredictable resource management tasks because of its capacity to discover optimal techniques through feedback loops.

Exploration vs. Exploitation : The agent in DRL must choose between exploration and exploitation.

- Exploration: To learn more about the environment, the agent experiments with various offloading techniques at this phase, even if they are not ideal. This is essential when learning is just getting started or when the system's circumstances change.
- Exploitation: After learning which behaviors result in large rewards, the agent concentrates on using that knowledge to its advantage by consistently selecting behaviors that are known to maximize the benefit.

Comparing DRL and Conventional Machine Learning Methods in MEC Systems: Labeled data is necessary for training models in traditional machine learning approaches like supervised learning. These techniques work well in situations with static data or known patterns, like regression or classification tasks. However, conventional methods are frequently insufficient for real-time optimization in MEC systems, where tasks and network circumstances vary continuously.

DRL, on the other hand, enables ongoing learning and adjusts to shifting conditions. It learns from interactions with the environment rather than pre-labeled information. DRL models can handle real-time changes in computational load, network congestion, and task priority in task offloading and resource allocation. They can also experiment with various approaches and improve their choices over time. DRL is an effective method for handling dynamic, multi-dimensional optimization problems in MEC systems because of its adaptability.

5.2 DRL-Based Task Offloading Framework

Description of DRL-Based Architecture for Task Offloading Decisions: Several essential elements comprise a DRL-based architecture for job offloading in MEC systems: the agent, environment, state, action, and reward.

- **Agent:** The DRL agent is in charge of learning and choosing which tasks to offload. Depending on the condition of the system, it communicates with the environment to decide whether to offload work to cloud servers or edge nodes.
- **Environment:** The MEC system's environment consists of all of its components, including communication channels, cloud servers, edge nodes, and mobile devices. It gives the agent information about how well the system is working and how its choices are affecting things like resource usage, latency, and energy consumption.
- **State (s):** The state is a representation of the system's current status, including task characteristics, network bandwidth, MD computational load, and edge node resource availability.
- **Action (a):** The agent's choice to either complete the task locally or offload it to a particular edge node or cloud server. Decisions regarding when and how much of the job to offload are also included in the action set.
- **Reward (r):** Depending on how successful the action was, the agent receives a numerical value as the reward. Metrics of system performance including decreased latency, energy use, and resource utilization are reflected in rewards.

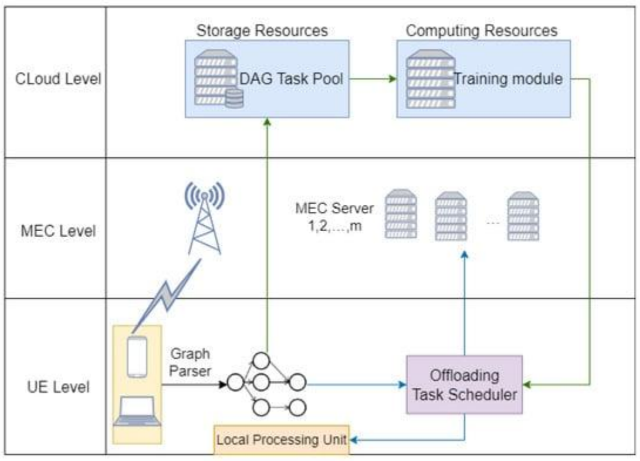

Fig 2: DRL-Based Task Offloading Framework

The agent's goal is to learn policies that result in effective task offloading in order to maximize the cumulative reward over time. As the system develops, the agent improves its policy through repeated interactions with the surroundings.

This entails striking a balance between exploring novel offloading techniques (such as experimenting with different edge nodes or cloud services) and utilizing techniques that have been developed to maximize throughput while minimizing latency and energy consumption.

5.3 DRL for Resource Allocation in MEC

Applying DRL for Dynamic Allocation of Computational and Network Resources: Computational and network resources are usually scarce in MEC systems and must be distributed effectively. By using DRL to dynamically distribute these resources, MDs' tasks can be processed in a way that maximizes energy efficiency, minimizes latency, and maximizes resource utilization.

By modifying its choices in response to real-time data about system conditions, the DRL agent can learn to distribute resources dynamically. For instance, the agent may choose to transfer jobs to a cloud server or less-used edge node when the network is crowded, taking into account variables like task urgency, network bandwidth, and available computing power.

The DRL model can optimize resource allocation strategies to achieve the system's goals, such as reducing delays and upholding Quality of Service (QoS) standards for users, by being trained on historical data and current system performance.

Techniques to Optimize Resource Distribution While Minimizing Delay and Maximizing QoS: To optimize resource allocation, DRL techniques like Q-learning and Deep Q-Networks (DQN) can be applied. By using these techniques, the agent may assess various resource distribution plans and choose the one that ensures high throughput and low energy usage while minimizing delays.

- Q-learning: This method uses anticipated future rewards to teach an agent the value of actions in a particular state. By balancing exploration and exploitation, the agent gradually learns the best resource allocation policy.
- Deep Q-Networks (DQN): DQNs use a deep neural network to approximate the Q-value function, extending Q-learning. DQNs are helpful in managing the extensive state and action spaces that are characteristic of MEC systems, where there may be a large number of options for resource allocation and task offloading.

Through the use of these methods, DRL allows the MEC system to dynamically modify the distribution of resources, taking into consideration variables such as network circumstances, energy limits, computing load, and QoS needs.

Evaluation metrics will include:

- **Latency:** The overall amount of time needed to process and offload jobs.
- **Energy Consumption**: MDs' overall energy usage when offloading tasks.
- **Throughput:** The quantity of jobs completed successfully in a specified amount of time.
- **Resource Utilization:** The efficiency with which cloud servers and edge nodes use their computational resources.

6. Integration of Multi-Objective Optimization and DRL

6.1 Combining Optimization and DRL

Methodology to Integrate Multi-Objective Optimization with DRL to Enhance Resource Allocation and Task Offloading: Combining the advantages of both methods to manage dynamic resource allocation and task offloading in MEC systems is the goal of integrating multi-objective optimization with Deep Reinforcement Learning (DRL). DRL is superior at making adaptive, real-time judgments by learning from the environment, whereas multi-objective optimization offers an organized method to balance conflicting objectives (such latency, energy usage, and throughput).

The proposed methodology integrates these two approaches in the following way:

- **Multi-Objective Optimization Layer:** The Multi-Objective Optimization Layer is in charge of defining and resolving the multi-objective optimization problem while making sure that trade-offs between competing goals are clearly stated. The system can be guided towards solutions that respect several objectives (e.g., decreasing latency while maximizing throughput) by using optimization approaches such as genetic algorithms or Pareto-based methodologies.
- **DRL Layer:** The DRL agent uses the outcomes of the optimization process to direct its learning while operating inside this optimization framework. The multi-objective outcomes are used to inform the agent's reward signal, making sure that its choices support the objectives of the system. The DRL agent continuously improves its approach to resource allocation and job offloading as it engages with the environment, adapting to real-time system conditions.

Hybrid Models for Balancing Multiple Objectives (e.g., Latency, Energy, Throughput):

With DRL fine-tuning judgments for individual real-time conditions and standard optimization techniques providing a high-level solution space, hybrid models that combine multi-objective optimization and DRL are especially good at balancing numerous objectives. For instance:

- **Genetic Algorithm + DRL:** By identifying the most effective tactics in real-world scenarios, the DRL agent may hone in on the many possible offloading and resource allocation solutions that genetic algorithms can investigate.
- **Pareto-based Optimization + DRL:** Pareto-based techniques can produce a collection of non-dominated solutions that reflect various goal trade-offs. Based on the current condition of the system, DRL can then investigate and take advantage of this solution area, enabling more focused decision-making.

The hybrid model ensures that real-time decisions for task offloading and resource allocation are made with a broader understanding of the trade-offs involved, enhancing overall system efficiency and user satisfaction.

6.2 Optimization Process with DRL

How the Exploration and Learning Process of the DRL Model Are Guided by Optimization Criteria:

In the integrated method, the DRL agent's exploration and learning process are guided by the multi-objective optimization criteria, such as minimizing latency, maximizing throughput, and minimizing energy usage. The optimization model offers the DRL agent a wide range of viable offloading tactics that adhere to the system's goals and limitations during the training phase. The exploring phase of the DRL agent is started using these techniques.

- **Exploration:** Within the parameters established by the multi-objective optimization, the DRL agent is first encouraged to investigate various resource allocation and task offloading mechanisms. This enables the agent to get knowledge about how various activities impact the energy consumption, latency, and throughput of the system.
- Learning: By linking actions to incentives determined by the multi-objective optimization criteria, the agent gradually improves its policies. For instance, the agent is rewarded based on the trade-offs between the two competing goals when it decides to offload a task in a way that reduces energy consumption but increases latency. By balancing all goals, the agent modifies its approach to optimize the long-term benefit.

The DRL agent can focus on the most promising regions of the decision space and speed up learning by using the optimization criteria as a baseline. This results in faster convergence and more efficient task offloading techniques.

Adaptation and Real-Time Modifications for Resource Allocation Using Multi-Objective Results: Adjusting in real time depending on ongoing system performance is one of the main advantages of combining multi-objective optimization with DRL. The DRL agent modifies its decision-making in response to changes in the system's network conditions, computing load, or energy availability, while the multi-objective optimization offers a framework for modifying priorities.

For instance, even if it causes a little increase in latency, the DRL agent can modify the offloading decision to lower energy consumption if the energy consumption in a certain edge node gets too high. In contrast, the agent will offload jobs to the closest edge node or cloud server, taking into consideration network conditions and resource availability, if low latency is prioritized because of critical tasks.

System performance in dynamic MEC contexts is enhanced by the constant feedback loop between DRL and multi-objective optimization, which guarantees that resource allocation choices are adaptable and sensitive to real-time changes.

7. Discussion

7.1 Performance Metrics

The performance of the Deep Reinforcement Learning (DRL) model and integrated multi-objective optimization is assessed in this part using a number of key performance indicators (KPIs) that are pertinent to the job offloading and resource allocation issue in MEC systems.

The performance metrics offer a numerical assessment of how successfully the suggested solution accomplishes its goals.

- **Latency:** Because customers anticipate quick response times for time-sensitive applications, latency is one of the most important performance measures in MEC systems. It calculates the time lag between starting and finishing a task, including any communication lags brought on by shifting work to cloud servers or edge nodes. Better decisions on job offloading and network efficiency are shown by lower latency.

- **Energy Efficiency:** Since most gadgets in mobile contexts run on batteries, energy efficiency is essential. This measure measures the overall amount of energy used by edge nodes and mobile devices for processing and offloading tasks. The objective is to balance the trade-offs between energy efficiency and task completion time in order to reduce energy consumption without sacrificing performance.
- **Computational Load:** This is the quantity of processing power needed by cloud servers or edge nodes to manage jobs that have been offloaded. By avoiding overloading any one node and making sure that jobs are distributed effectively based on node capacity, this metric aids in evaluating how well the system divides the computational load among the resources that are available.
- **Task Completion Time:** This includes the time needed for calculation, transmission, and offloading at the edge or cloud server. It is the whole amount of time needed to finish a task from the beginning to the end. Because it has a direct effect on real-time application performance, a shorter task completion time is preferred.

7.2 Discussion of Trade-offs Between Different Optimization Goals:

Crucial factor in the assessment is the trade-offs between the conflicting optimization objectives (latency, energy efficiency, computational load, and task completion time). Although the integrated model seeks to strike a balance between these goals, there are trade-offs that must be considered:

- **Energy Efficiency vs. Latency:** Tasks must frequently be offloaded to the closest available edge node in order to minimize latency, which may use more energy. However, in order to optimize for energy efficiency, jobs may be offloaded to nodes that are farther away, which would increase latency. By skillfully balancing these trade-offs, the integrated model makes sure that neither goal is given undue priority at the expense of the other.
- **Throughput vs. Computational Load:** Increasing throughput by handling more activities at once could put additional computational strain on cloud servers or edge nodes, which could cause resource congestion or task completion delays. This is addressed by the hybrid approach, which dynamically modifies resource allocation to sustain high throughput without putting undue strain on any one resource.

The integrated approach shows how DRL and multi-objective optimization may be used together to make decisions in real time that balance various trade-offs and enhance system performance without sacrificing any one goal at the expense of another.

7.3 Challenges and Future Work

Identifying Challenges in Real-World Implementation of the Proposed Models:

Although the integrated model exhibits encouraging outcomes in the simulated setting, there are still a number of obstacles to overcome before these results can be applied to actual MEC systems:

- **Environmental Change:** Real-world settings are prone to a variety of uncertainties in practice, such as shifting network conditions, different task attributes, and dynamic user behavior. The DRL agent may find it challenging to sustain high performance over time due to this fluctuation. More resilient learning algorithms and system architectures that can manage erratic real-world circumstances are needed to adjust to these uncertainties.
- **Computational Complexity**: Because deep neural networks must be trained and several actions must be assessed at each decision stage, the usage of DRL adds to the computational complexity. The computational load of updating and maintaining the DRL model can be significant in real-world MEC systems with many of devices, edge nodes, and jobs, particularly for devices with limited resources.
- **Data Security and Privacy:** MEC systems frequently handle private user information, which presents security and privacy issues. Careful thought must be given to how to protect user data when job offloading and maintain privacy without compromising system performance when integrating DRL with multi-objective optimization.

- **Implementation and Deployment Costs:** Using an optimization-based and DRL-integrated solution in a real MEC system may result in significant deployment and operating expenses. In addition to continuing expenses for system upkeep and upgrades, these costs may result from the requirement for extra infrastructure, such as strong edge nodes or centralized servers for model training.

Suggestions for Future Research Directions:

To address the challenges mentioned above, several areas of future research can be explored:

- **Improving Learning Algorithms:** Future research can concentrate on enhancing the DRL models' learning algorithms, especially for large-scale MEC settings. In order to increase scalability, robustness, and flexibility, methods including multi-agent reinforcement learning, federated learning, and transfer learning could be investigated.
- **Handling Larger-Scale MEC Environments**: Research on effective scaling strategies for DRL models and multi-objective optimization techniques will be crucial as the number of mobile devices and edge nodes keeps increasing. Creating decentralized or distributed solutions that are better suited to managing large-scale systems is part of this.
- **Including Edge and Cloud Collaboration**: More complex partnerships between edge nodes and cloud servers may be investigated in future studies. Resource allocation and task offloading may be further improved by hybrid models that combine the advantages of edge and cloud computing, such as shifting decision-making to the cloud in response to current network conditions.
- **Privacy-Preserving methods:** In order to protect user data while preserving high performance, future research could look into how privacy-preserving methods can be included into the DRL-based task offloading architecture. To safeguard data privacy, methods like secure multi-party computation and homomorphic encryption could be used.

8. Conclusion

8.1 Summary of Contributions

This work proposes an integrated method for dynamic resource allocation and job offloading in Mobile Edge Computing (MEC) systems that integrates Deep Reinforcement Learning (DRL) and multi-objective optimization. This study's primary contributions are as follows:

- **Hybrid Approach:** We suggest a novel hybrid framework that uses DRL to adaptively make real-time decisions for task offloading and resource allocation, while also utilizing multi-objective optimization techniques (like Pareto-based methods) to balance important objectives like latency, energy consumption, and throughput.
- **Multi-Objective Optimization in MEC:** We show how various objectives can be balanced in MEC systems by offering a mathematical definition of the multi-objective optimization problem. The DRL agent is guided by the optimization criteria, which guarantee that decisions made in real time are in line with the overall performance objectives of the system.
- **Real-Time Adaptability with DRL:** The study emphasizes how DRL helps MEC systems learn the best offloading techniques based on system feedback, improving their capacity to adjust to changing user demands, task requirements, and network constraints.
- **Thorough Evaluation**: We demonstrate through extensive simulations that the integrated multi-objective optimization and DRL framework performs better than baseline models (such as DRL-only and optimization-only models) in a number of performance metrics, such as task completion time, energy efficiency, latency, and computational load.

8.2 Conclusion

The difficulties presented by dynamic resource allocation and job offloading in MEC systems are successfully resolved by the suggested method of combining multi-objective optimization with DRL. The hybrid framework offers the flexibility required to manage the dynamic nature of MEC situations in addition to guaranteeing a well-balanced trade-off between several objectives. The simulation's outcomes show that this integrated strategy works better than conventional techniques, providing increased scalability, efficiency, and real-time flexibility.

The capacity of the hybrid approach to make context-aware decisions, adjust to changing circumstances, and maximize resource use across numerous devices and edge nodes accounts for its efficacy. Because of this, it is especially pertinent to upcoming networks and edge computing applications where energy efficiency, high throughput, and low latency are critical. The combination of multi-objective optimization with DRL holds great potential for guaranteeing the effective, real-time operation of these systems, as MEC remains essential in enabling the next generation of IoT, smart cities, autonomous systems, and 5G applications.

In summary, the study shows that robust and flexible solutions for task offloading and resource allocation in MEC systems may be created by fusing the advantages of DRL with multi-objective optimization. The results offer a solid basis for upcoming developments in edge computing, especially in the creation of resource management systems that are more responsive, scalable, and energy-efficient.

References

[1] Dastjerdi, A. V., & Buyya, R. (2016). Fog computing: Promises and challenges of distributed processing at the edge. ACM Computing Surveys.

[2] Bao-Shan Sun, Hao Huang, Zheng-Yi Chai, Ying-Jie Zhao, Hong-Shen Kang.(2024) Multi-objective optimization algorithm for multi-workflow computation offloading in resource-limited IIoT.

[3] Liu, Y., & Xu, M. (2019). Task offloading and resource allocation in MEC: A survey. Journal of Network and Computer Applications.

[4] S. E. Bouzid, Y. Seresstou, K. Raoof, M. N. Omri, M. Mbarki, And C. Dridi (May 14, 2020) MOONGA: Multi-Objective Optimization of Wireless Network Approach Based on Genetic Algorithm. Digital Object Identifier 10.1109/ACCESS.2020.2999157

[5] Do-Young Lee, Se-Yeon Jeong, Kyung-Chan Ko, Jae-Hyoung Yoo, James Won-Ki Hong(27 June 2021)Deep Q-network-based auto scaling for service in a multi-access edge computing environment. https://doi.org/10.1002/nem.2176

[6] panelHaifeng Lu, Chunhua Gu, Fei Luo, Weichao Ding, Xinping Liu(January 2020) a Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning. https://doi.org/10.1016/j.future.2019.07.019

[7] Chen, Y., et al. (2019). Energy-efficient task offloading in multi-access edge computing. IEEE Journal on Selected Areas in Communications. https://doi.org/10.1109/TMC.2023.3346431

[8] Sihua Wang, Xuanlin Liu, et al. (2019). Task offloading and resource allocation in mobile edge computing: A reinforcement learning approach. Journal of Communications and Networks. http://dx.doi.org/10.1109/GCWkshps45667.2019.9024603

[9] Zhang, L., et al. (2017).Multi-objective optimization for computation offloading in mobile-edge computing. http://dx.doi.org/10.1109/ISCC.2017.8024630

[10] Annisa Sarah, Gianfranco Nencioni, Md. Muhidul I. Khan Resource Allocation in Multi-access Edge Computing for 5G-and-beyond networks.Volume 227, Issue C https://doi.org/10.1016/j.comnet.2023.109720.

[11] Mushu Li. Xuemin Shen (12-2020) Deep Reinforcement Learning for Collaborative Edge Computing in Vehicular Networks.

[12] Fu, X., et al. (2021). Performance analysis of task offloading in MEC systems with QoS constraints. IEEE Transactions on Cloud Computing.

[13] Wang, L., et al. (2020). Design of an energy-efficient MEC system based on multi-objective optimization. Journal of Communications and Networks.

[14] Soni, S., et al. (2019). Task offloading in mobile edge computing: A survey of algorithms and models. Journal of Network and Computer Applications.

[15] Qiao, Y., et al. (2020). Optimization of task offloading in MEC using deep reinforcement learning. International Journal of Communication Systems.

[16] J. Liu, Y. Mao, J. Zhang and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems", Proc. IEEE Int. Symp. Inf. Theory, pp. 1451-1455, 2016.

[17] Y. Mao, J. Zhang and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices", IEEE J. Sel. Area Commun., vol. 34, no. 12, pp. 3590-3605, Dec. 2016.

[18] S. Sardellitti, G. Scutari and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing", IEEE Trans. Signal Inf. Process., vol. 1, no. 2, pp. 89-103, Jun. 2015.

[19] X. Chen, L. Jiao, W. Li and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing", IEEE/ACM Trans. Netw., vol. 24, no. 5, pp. 2795-2808, Oct. 2015.
[20] Y. Xu et al., "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues", Inf. Sci., vol. 270, pp. 255-287, 2014.
[21] S. Yu, W. Xin and R. Langar, "Computation offloading for mobile edge computing: A deep learning approach", Proc. IEEE 28th Annu. Int. Symp. Personal Indoor Mobile Radio Commun., pp. 1-6, 2017.
[22] Y. Mao, J. Zhang and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems", Proc. IEEE Wireless Commun. Netw. Conf., pp. 1-6, 2017.
[23] T. Q. Dinh, J. Tang, Q. D. La and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling", IEEE Trans. Commun., vol. 65, no. 8, pp. 3571-3584, Aug. 2017.
[24] Z. Meng, H. Xu, L. Huang, P. Xi and S. Yang, "Achieving energy efficiency through dynamic computing offloading in mobile edge-clouds", Proc. IEEE 15th Int. Conf. Mobile Ad Hoc Sensor Syst., pp. 175-183, 2018.
[25] Y. Wang, K. Wang, H. Huang, T. Miyazaki and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications", IEEE Trans. Ind. Inform., vol. 15, no. 2, pp. 976-986, Feb. 2019.
[26] Z. Ning, P. Dong, X. Wang and J. J. Rodrigues, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system", ACM Trans. Intell. Syst. Technol., vol. 10, no. 6, pp. 1-24, 2019.
[27] L. Huang, X. Feng, L. Qian and Y. Wu, "Deep reinforcement learning-based task offloading and resource allocation for mobile edge computing", Proc. Int. Conf. Mach. Learn. Intell. Commun., pp. 33-42, 2018.
[28] J. Wang, J. Hu, G. Min, A. Y. Zomaya and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning", IEEE Trans. Parallel Distrib., vol. 32, no. 1, pp. 242-253, Jan. 2021.
[29] Z. Cao, P. Zhou, R. Li, S. Huang and D. Wu, "Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0", IEEE Internet Things, vol. 7, no. 7, pp. 6201-6213, Jul. 2020.
[30] M. Keshavarznejad, M. H. Rezvaniand and S. Adabi, "Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms", Cluster Comput., vol. 24, no. 3, pp. 1825-1853, 2021.
[31] M. K. Hussein and M. H. Mousa, "Efficient task offloading for IoT-based applications in fog computing using ant colony optimization", IEEE Access, vol. 8, pp. 37191-37201, 2020.
[32] A. Bozorgchenani, F. Mashhadi, D. Tarchi and S. S. Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments", IEEE Trans. Mobile Comput., vol. 20, no. 10, pp. 2992-3005, Oct. 2021.
[33] H. Jiang, X. Dai, Z. Xiao and A. K. Iyengar, "Joint task offloading and resource allocation for energy-constrained mobile edge computing", IEEE Trans. Mobile Comput., 2022.
[34] H. Lu, C. Gu, F. Luo, W. Ding and X. Liu, "Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning", Future Gener. Comput. Syst., vol. 102, pp. 847-861, 2020.
[35] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan and Y. Zhang, "Deep learning empowered task offloading for mobile edge computing in urban informatics", IEEE Internet Things, vol. 6, no. 5, pp. 7635-7647, Oct. 2019.
[36] G. Qu, H. Wu, R. Li and P. Jiao, "DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing", IEEE Trans. Netw. Service Manage., vol. 18, no. 3, pp. 3448-3459, Sep. 2021.
[37] J. Baek and G. Kaddoum, "Heterogeneous task offloading and resource allocations via deep recurrent reinforcement learning in partial observable multifog networks", IEEE Internet Things, vol. 8, no. 2, pp. 1041-1056, Jan. 2021.
[38] K. Li, X. Tang and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems", IEEE Trans. Parallel Distrib., vol. 25, no. 11, pp. 2867-2876, Nov. 2014.
[39] F. Mashhadi, S. A. S. Monroy, A. Bozorgchenani and D. Tarchi, "Optimal auction for delay and energy constrained task offloading in mobile edge computing", Comput. Netw., vol. 183, 2020.
[40] P. Kaelo and M. M. Ali, "Some variants of the controlled random search algorithm for global optimization", J. Optim. Theory Appl., vol. 130, no. 2, pp. 253-264, 2006.
[41] F. Saidak, "A new proof of Euclid's theorem", Amer. Math. Monthly, vol. 113, no. 10, pp. 937-938, 2006.

Journal of Information Systems Engineering and Management
2025, 10(55s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

[42] C. Liu, K. Li and K. Li, "A game approach to multi-servers load balancing with load-dependent server availability consideration", IEEE Trans. Cloud Comput., vol. 9, no. 1, pp. 1-13, Jan.–Mar. 2021.

[43] J. Chen et al., "A parallel random forest algorithm for big data in a spark cloud computing environment", IEEE Trans. Parallel Distrib., vol. 28, no. 4, pp. 919-933, Apr. 2017.