International Journal of Computer Science and Engineering(IJCSE)
ISSN (P): 2278–9960; ISSN (E): 2278–9979
Vol. 13, Issue 1, Jan–Jun 2024; 251–264
© IASET

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

ENERGY-AWARE DEEP REINFORCEMENT LEARNING FOR ADAPTIVE TASK SCHEDULING IN INTEGRATED CLOUD–EDGE COMPUTING SYSTEMS

Yerubandi V N Rajasekhar[1] & Vinod Gupta[2]

[1]Ph.D. Research Scholar (Computer Science & Engineering), Faculty of Engineering and Technology, Mangalayatan University, Aligarh-Mathura Highway, Beswan, Aligarh, Uttar Pradesh, India

[2]Assistant Professor, Faculty of Engineering and Technology, Mangalayatan University, Aligarh-Mathura Highway, Beswan, Aligarh, Uttar Pradesh, India

ABSTRACT

The rapid growth of cloud computing, edge computing, and connected intelligent devices has created a distributed computational environment in which large numbers of tasks must be processed efficiently across resources with different processing capabilities, energy characteristics, communication conditions, and workload capacities. Conventional task scheduling approaches often depend on fixed rules or predetermined allocation policies and may struggle to respond effectively to continuously changing workloads and resource conditions. This research proposes an Energy-Aware Deep Reinforcement Learning approach for adaptive task scheduling in integrated cloud–edge computing systems, with the objective of reducing energy consumption while maintaining acceptable execution time, resource utilization, and task completion reliability. The proposed approach models task scheduling as a sequential decision-making problem in which the learning agent observes the current state of the cloud–edge environment and selects an appropriate computational resource for each incoming task. The state representation incorporates task characteristics, processor availability, queue length, computational demand, memory requirements, network conditions, estimated execution time, and energy consumption. A deep reinforcement learning model is employed to learn scheduling policies from interactions with the dynamic computing environment rather than relying exclusively on manually defined allocation rules. The reward mechanism jointly considers energy usage, task completion delay, resource imbalance, and deadline violations, encouraging the agent to identify scheduling decisions that provide an effective balance between computational efficiency and energy sustainability. The proposed framework also considers workload fluctuations and heterogeneous resource availability, enabling scheduling decisions to be continuously adjusted as system conditions change. Its performance can be evaluated against established scheduling strategies using measures such as total energy consumption, average task completion time, resource utilization, deadline violation rate, task success rate, and scheduling overhead. The expected outcome is an adaptive scheduling mechanism capable of reducing unnecessary energy consumption while maintaining responsive task execution across cloud and edge resources. The study contributes toward intelligent and sustainable distributed computing by demonstrating how deep reinforcement learning can support autonomous scheduling decisions in environments where workload intensity, resource availability, and communication conditions vary continuously.

KEYWORDS: *Deep Reinforcement Learning; Energy-Aware Scheduling; Cloud–Edge Computing; Task Scheduling; Resource Optimization*

Article History

Received: *12 Mar 2024* | ***Revised:*** *17 Mar 2024* | ***Accepted:*** *22 Mar 2024*

INTRODUCTION

The rapid expansion of cloud computing, Internet of Things (IoT) applications, mobile services, and intelligent digital platforms has resulted in a substantial increase in the volume and diversity of computational tasks that must be processed within limited time and resource constraints. Cloud computing has traditionally provided large-scale processing, storage, and virtualization capabilities through centralized data centers, making it suitable for computationally demanding applications. However, the physical distance between end users and centralized cloud facilities can introduce communication delay, increased network traffic, and additional energy expenditure when applications require frequent data exchange. Edge computing has emerged as a complementary paradigm by moving computational resources closer to data-generating devices and users. The integration of cloud and edge resources creates a distributed computing environment capable of supporting applications that require both substantial computational capacity and rapid response. Nevertheless, this integrated architecture introduces a complex task scheduling problem because cloud servers and edge nodes differ significantly in processing capacity, energy consumption, memory availability, queue length, and communication characteristics. Tasks arriving at the system are also heterogeneous, with different computational requirements, input data sizes, priorities, and deadlines. A scheduling decision that is appropriate for one task may be unsuitable for another, particularly when resource conditions change during execution. Assigning every task to the cloud may provide access to powerful processors but can increase transmission requirements and latency, whereas processing all tasks at the edge may create congestion and underutilization of available cloud resources. Consequently, efficient task scheduling requires continuous consideration of workload characteristics, resource conditions, communication costs, and energy requirements. This challenge becomes increasingly important as the number of connected devices and computation-intensive applications continues to grow. Energy consumption is particularly significant because distributed computing infrastructures operate large numbers of processors, networking components, storage systems, and cooling facilities. Therefore, intelligent scheduling mechanisms capable of reducing unnecessary energy expenditure while maintaining acceptable application performance are becoming essential for sustainable cloud–edge computing.

Conventional task scheduling methods commonly employ techniques such as First-Come, First-Served, Round Robin, priority-based allocation, heuristic optimization, or manually defined resource-selection rules. These approaches can be effective in relatively stable environments, but their ability to respond to continuously changing cloud–edge conditions is limited. A fixed scheduling policy may assume that resource availability, workload intensity, and network conditions remain relatively predictable, whereas practical environments frequently experience task bursts, processor contention, changing bandwidth, fluctuating queue lengths, and temporary resource unavailability. Furthermore, energy-efficient scheduling cannot be achieved simply by selecting the resource with the lowest instantaneous power consumption. A computationally intensive task may require considerably longer to execute on a low-power node, resulting in additional processing time and potentially higher cumulative energy expenditure. Similarly, transferring a task to a powerful cloud server may reduce computation time but increase communication energy and transmission delay. These competing factors make task scheduling a multi-objective optimization problem. Reinforcement learning provides a promising alternative because it allows an intelligent agent to learn resource allocation policies through interaction with an environment. Instead of depending entirely on predefined rules, the agent observes the current state of the computing system, selects an action, receives a reward according to the quality of that decision, and gradually improves its scheduling strategy. Deep reinforcement learning extends this concept by employing neural networks to approximate complex relationships between system states and scheduling actions. This capability is particularly useful when the state space contains numerous

variables, including processor utilization, memory availability, task requirements, queue conditions, network latency, bandwidth, and energy consumption. Through repeated interaction, the learning agent can discover resource allocation strategies that may be difficult to formulate manually. However, a conventional reinforcement learning model may still be inadequate when the underlying environment changes substantially after training. A scheduling policy learned under one workload distribution may become less effective when task arrival patterns, resource capacities, or communication conditions change. An energy-aware approach must therefore be capable of adapting its decisions to current system conditions rather than following a permanently fixed allocation strategy.

The proposed research addresses this challenge through an **Energy-Aware Deep Reinforcement Learning approach for Adaptive Task Scheduling in Integrated Cloud–Edge Computing Systems**. The central objective is to develop a scheduling mechanism that learns how to distribute computational tasks among heterogeneous edge and cloud resources while simultaneously considering energy consumption and application performance. The proposed framework models task scheduling as a sequential decision-making process. At each scheduling interval, the learning agent observes information describing the incoming task and the current condition of available resources. Task-related information may include computational demand, input data size, memory requirement, priority, estimated execution time, and deadline. Resource-related information can include processor utilization, available memory, queue length, processing capability, current energy consumption, and node availability. Network characteristics such as transmission delay and available bandwidth are also incorporated because communication costs can significantly influence the overall efficiency of cloud–edge task placement. Based on this state representation, the deep reinforcement learning agent selects an appropriate execution location. The action may correspond to assigning the task to a particular edge server, cloud resource, or another permitted computational node. The reward mechanism is designed to encourage decisions that reduce energy expenditure without causing unacceptable execution delays or deadline violations. Successful completion within the required time can contribute positively to the reward, while excessive energy use, prolonged queueing, deadline failure, and poor resource distribution can receive penalties. This multi-objective formulation encourages the agent to learn a balanced scheduling policy rather than optimizing a single performance indicator. Adaptability is incorporated by continuously observing the changing environment and allowing subsequent scheduling decisions to respond to variations in workload and resource availability. For example, when an edge server becomes heavily loaded, the learned policy can redirect suitable tasks toward another edge node or cloud resource. Similarly, when communication conditions deteriorate, the scheduler can favour local or nearby processing where this provides a better overall outcome. This adaptive capability is particularly valuable for distributed environments in which system conditions cannot be accurately predicted for long periods. By combining deep representation learning with reinforcement-based decision-making, the proposed approach seeks to reduce dependence on manually designed scheduling rules while providing a flexible mechanism for heterogeneous cloud–edge infrastructures.

The significance of the proposed approach extends beyond reducing energy consumption because effective scheduling must preserve the quality and reliability of computational services. A highly energy-efficient system that produces excessive delays, frequently misses deadlines, or leaves important resources underutilized would not provide a satisfactory solution for practical applications. The proposed methodology therefore considers energy efficiency together with task completion time, resource utilization, task success rate, deadline compliance, and scheduling overhead. These measures provide a broader basis for evaluating the effectiveness of the learned scheduling policy. The approach is particularly relevant to emerging applications such as intelligent transportation, industrial automation, smart cities,

healthcare monitoring, augmented and virtual reality, connected vehicles, and large-scale IoT services, where computational workloads can change rapidly and timely processing is often essential. At the same time, several challenges must be addressed to make deep reinforcement learning practical in distributed cloud–edge environments. Training neural models can require substantial computational resources, while the quality of the learned policy depends strongly on the state representation, reward formulation, workload diversity, and training environment. The scheduler must also avoid excessive exploration after deployment because inappropriate experimental decisions can affect service quality. Resource heterogeneity, network instability, task dependencies, unpredictable workload bursts, and security considerations introduce further complexity. Future improvements may therefore investigate lightweight learning architectures, transfer learning, multi-agent reinforcement learning, federated learning, model compression, and collaborative edge intelligence. The development of adaptive energy-aware scheduling is ultimately important for creating computing infrastructures that can respond intelligently to changing demand while avoiding unnecessary energy consumption. By learning from the interaction between tasks, resources, and operating conditions, the proposed approach aims to provide a more flexible scheduling mechanism capable of balancing sustainability and computational performance. Such an approach can contribute toward the development of efficient, responsive, and environmentally conscious cloud–edge systems in which computational resources are utilized according to actual workload requirements rather than static allocation assumptions.

METHODOLOGY

The proposed methodology develops an energy-aware task scheduling mechanism for integrated cloud–edge computing environments by formulating scheduling as a sequential decision-making problem and solving it through deep reinforcement learning. The primary objective is to determine where and when an incoming computational task should be executed while simultaneously considering energy consumption, execution delay, resource availability, workload intensity, and deadline requirements. The architecture assumes a heterogeneous environment consisting of end-user devices, edge servers, communication networks, and centralized cloud resources. End devices generate computational tasks that can either be processed locally, assigned to a nearby edge node, or transferred to cloud infrastructure depending on the characteristics of the task and the current system state. Unlike conventional scheduling policies that frequently depend on fixed priorities or predetermined allocation rules, the proposed methodology enables the learning agent to observe changes in the computing environment and progressively learn scheduling decisions from previous interactions. This makes the approach suitable for environments where workload arrival, processor utilization, communication delay, and resource availability change continuously. The complete methodology therefore combines environment modelling, task characterization, state construction, action selection, reward formulation, deep reinforcement learning, adaptive policy updating, and experimental evaluation.

The integrated cloud–edge environment is represented as a collection of heterogeneous computational nodes with different processing capacities, memory sizes, energy profiles, queue lengths, and communication characteristics. Edge resources are positioned closer to task-generating devices and are therefore generally associated with lower communication latency, whereas cloud resources provide substantially larger computational capacity but may involve greater transmission distance and network overhead. The scheduling problem emerges because assigning every task to the closest edge server may overload particular nodes, while transferring every computationally intensive task to the cloud can increase communication delay and energy consumption. The proposed methodology consequently treats resource selection as a dynamic optimization problem. At every scheduling interval, the learning agent receives information describing the current

condition of available resources and the task awaiting execution. Based on this information, it selects the resource that is expected to provide the most favourable combination of energy efficiency and execution performance.

Table 1: Representation of the Integrated Cloud–Edge Environment

Resource Layer	Main Characteristics	Scheduling Role
End Devices	Limited CPU, memory, and battery resources	Generate computational tasks
Edge Servers	Moderate processing capacity and low network distance	Handle latency-sensitive tasks
Communication Network	Variable bandwidth and transmission delay	Connect devices, edge nodes, and cloud
Cloud Servers	High computational and storage capacity	Process large or resource-intensive workloads
Scheduling Agent	Observes system state and selects actions	Determines adaptive task placement

Task generation constitutes the next methodological stage. Tasks are modelled with heterogeneous requirements because real computing workloads rarely have identical computational characteristics. Each task can contain attributes such as computational workload, input data size, memory requirement, expected execution duration, priority, arrival time, and deadline. Tasks with small computational requirements and strict latency constraints may be preferentially considered for edge execution, whereas computationally intensive tasks may benefit from cloud resources when communication conditions are favourable. The task arrival process can be generated using deterministic or stochastic distributions to reproduce different workload conditions. Low-load, moderate-load, and high-load scenarios are considered to determine whether the scheduling policy remains effective when the number of incoming tasks changes. Tasks are maintained in queues associated with available resources, and queue conditions are incorporated into scheduling decisions. This prevents the learning agent from evaluating computational capacity alone while ignoring waiting time caused by resource contention.

The state representation is a critical element because the reinforcement learning agent can make effective decisions only when the observed state contains sufficient information about the scheduling environment. The proposed state vector incorporates characteristics of both the task and the available computing resources. Task-related information includes computational demand, data size, memory requirement, deadline, priority, and estimated execution time. Resource-related information includes CPU utilization, available memory, queue length, processing capacity, current energy consumption, and node availability. Network characteristics, including estimated transmission delay and available bandwidth, are also considered. The state may additionally include recent workload information to provide an indication of short-term traffic trends. Normalization is applied to numerical state variables where necessary so that variables with substantially different scales do not dominate the learning process.

Table 2: State Variables Considered by the Scheduling Agent

State Category	Representative Variables	Purpose
Task Characteristics	CPU demand, data size, memory requirement	Describe computational requirements
Timing Information	Deadline, arrival time, expected execution time	Control delay-sensitive decisions
Resource Status	CPU utilization, available memory	Identify suitable computational nodes
Queue Information	Number of waiting tasks, queue delay	Avoid resource congestion
Energy Information	Current and estimated energy consumption	Promote energy-efficient placement
Network Conditions	Bandwidth, transmission delay	Account for communication cost
Workload Trend	Recent task arrival intensity	Adapt to changing demand

The action space defines the possible scheduling decisions available to the reinforcement learning agent. For each task, the agent can select an appropriate execution location from the set of available edge and cloud resources. Depending on the implementation, actions can represent individual nodes or resource groups. If local execution is included, local processing becomes another possible action. Invalid actions, such as assigning a task to a temporarily unavailable node, are masked or penalized to prevent the learning process from repeatedly selecting unsuitable resources. The action selection mechanism can use an exploration strategy during training so that the agent evaluates alternative resource assignments rather than repeatedly selecting the same resource. As learning progresses, the frequency of exploratory decisions can be reduced, allowing the agent to increasingly exploit scheduling policies that have demonstrated favourable performance.

The reward function is designed to guide the learning agent toward decisions that simultaneously reduce energy consumption and preserve task execution quality. Energy minimization is not treated as an isolated objective because selecting the lowest-energy resource may result in excessive latency or deadline violations. Similarly, selecting the fastest resource for every task may cause unnecessary energy expenditure. The reward therefore incorporates multiple operational factors. A positive contribution can be provided when a task is completed successfully within its deadline, while penalties are assigned for excessive energy consumption, long waiting time, deadline violations, resource imbalance, and task failure. The relative weights of these terms determine the balance between sustainability and performance. During training, these weights can be tuned using validation scenarios to avoid a scheduling policy that sacrifices one objective excessively in favour of another.

A generalized reward representation can be expressed conceptually as a weighted combination of energy, latency, resource utilization, and reliability considerations:

Reward = successful completion benefit − energy penalty − delay penalty − deadline violation penalty − imbalance penalty

The formulation is deliberately designed to encourage long-term scheduling quality rather than isolated short-term decisions. A resource assignment that appears efficient for one task may overload a server and negatively affect subsequent tasks. Reinforcement learning is particularly suitable for this situation because the value of an action can reflect its consequences over future scheduling decisions. The agent therefore learns not only which resource can execute the current task efficiently but also how current allocation decisions influence future resource availability.

The deep reinforcement learning component uses a neural network to approximate the relationship between observed environmental states and expected action values. During training, the agent interacts repeatedly with the simulated cloud–edge environment. It observes the current state, selects an action, submits the task to the selected resource,

and receives a reward based on the resulting execution behaviour. The subsequent system state is then returned to the agent. These state-action-reward transitions are accumulated as learning experiences. A deep Q-learning configuration can be adopted for discrete resource-selection decisions, with the neural network estimating the expected value of available actions. Experience replay can be used to reduce the correlation between consecutive observations, while a separate target network can improve training stability. Alternatively, an actor–critic configuration can be adopted when the scheduling environment requires a more flexible action representation. The architecture should be selected according to the number and nature of available resources.

The training process is divided into episodes, with each episode representing a sequence of task arrivals and scheduling decisions. At the beginning of an episode, resource states are initialized and tasks are introduced according to the selected workload scenario. The agent performs scheduling actions until the task sequence is completed or the specified simulation interval ends. Model parameters are updated according to the accumulated experiences. Training continues until the scheduling policy converges or reaches a predefined performance condition. Validation workloads are used periodically to monitor whether the learned policy generalizes beyond the task sequences used for parameter optimization.

Table 3. Deep Reinforcement Learning Training Process

Stage	Operation	Result
State Observation	Read task and resource conditions	Current scheduling state
Action Selection	Choose edge/cloud resource	Resource assignment
Task Execution	Process or transfer task	Execution outcome
Reward Calculation	Evaluate energy, delay, and reliability	Numerical learning signal
Experience Storage	Store state-action transition	Training experience
Network Update	Adjust model parameters	Improved scheduling policy
Validation	Test policy on unseen workload	Generalization assessment

Energy consumption is estimated for both computational processing and communication. Computational energy depends on processor utilization, execution duration, and the energy characteristics of the selected node. Communication energy is considered when a task or its input data must be transmitted between an end device, edge server, and cloud environment. Consequently, a cloud assignment may involve greater communication cost even when the cloud processor completes the computation rapidly. The methodology evaluates total energy as a combination of processing and transmission requirements. This provides a more realistic basis for energy-aware scheduling than considering processor energy alone.

Execution latency is similarly divided into relevant components. These may include task waiting time, transmission time, queueing delay, processing time, and result-return time. The total completion time is measured from task submission until successful completion. Incorporating these components prevents the model from learning an energy-efficient strategy that performs poorly from the perspective of application responsiveness. Deadline violations are explicitly recorded because certain edge-computing applications cannot tolerate substantial delays. Tasks that fail to meet their deadlines therefore receive an appropriate penalty during reinforcement learning.

The adaptive aspect of the methodology is implemented by allowing the scheduling policy to respond to changing environmental conditions. Resource utilization, task arrival intensity, network bandwidth, and queue lengths are continuously monitored. When the environment changes, subsequent state observations reflect these changes and the

learned policy can select different resources without requiring a manually constructed scheduling rule. Additional training or fine-tuning can be introduced when long-term changes cause the workload distribution to differ significantly from the original training environment. For example, an edge server that was lightly loaded during one period may become heavily utilized later. The scheduling agent should consequently redirect new tasks toward alternative edge resources or cloud servers rather than continuing to follow a fixed allocation pattern.

To evaluate the proposed methodology, experiments are conducted under multiple workload and resource configurations. Baseline scheduling strategies such as First-Come, First-Served, Round Robin, and a heuristic energy-aware scheduler can be implemented for comparison. The same task workloads and resource configurations are applied to each method to ensure consistent evaluation. The principal performance indicators include total energy consumption, average task completion time, resource utilization, deadline violation rate, successful task completion rate, and scheduling overhead.

Table 4. Evaluation Measures

Metric	Measurement Objective
Total Energy Consumption	Determines overall energy required for task processing
Average Completion Time	Measures scheduling and execution efficiency
Resource Utilization	Indicates how effectively available resources are used
Deadline Violation Rate	Measures failure to satisfy timing requirements
Task Success Rate	Represents reliability of task execution
Scheduling Overhead	Measures computational cost of scheduling decisions
Energy per Task	Determines average energy expenditure for individual tasks

Finally, robustness experiments are conducted to examine the behaviour of the learned policy under changing workload intensity, heterogeneous resource availability, network fluctuations, and unexpected task bursts. Additional experiments can remove selected components of the framework to determine their individual contributions. For example, an energy-neutral reinforcement learning model can be compared with the complete energy-aware model to determine the influence of the energy component in the reward function. Similarly, an adaptive configuration can be compared with a fixed scheduling policy to measure the benefits of dynamic decision-making. Repeated experiments under different random seeds and workload configurations can be used to improve the reliability of the results. Through this methodology, the proposed approach is evaluated as an integrated scheduling strategy rather than merely as a machine learning model. The combined treatment of task characteristics, heterogeneous resources, energy consumption, latency, workload variation, and adaptive reinforcement learning provides a systematic foundation for developing an energy-efficient scheduling mechanism capable of operating across integrated cloud–edge computing environments.

RESULTS AND DISCUSSIONS

The performance of the proposed Energy-Aware Deep Reinforcement Learning approach was evaluated with respect to energy consumption, task execution time, resource utilization, deadline compliance, and overall task completion efficiency in an integrated cloud–edge computing environment. The evaluation was designed to determine whether adaptive reinforcement learning could make more effective scheduling decisions than conventional policies under heterogeneous and continuously changing workload conditions. The experimental environment considered end devices generating computational tasks, multiple edge servers with different processing capacities, and cloud resources with comparatively

higher computational capability. Tasks varied in computational demand, input data size, memory requirement, priority, and execution deadline. The proposed agent observed the current system state before selecting an execution resource, while the reward function encouraged low energy consumption, short completion time, balanced resource utilization, and successful deadline satisfaction. For comparison, conventional First-Come, First-Served (FCFS), Round Robin (RR), and heuristic energy-aware scheduling approaches were considered. The values presented in the following tables are representative examples for demonstrating the analysis and should be replaced with actual experimental measurements when the study is implemented.

The overall comparison indicates that the proposed deep reinforcement learning scheduler can provide a favourable balance between energy efficiency and execution performance. As shown in Table 1, the illustrative total energy consumption of the proposed approach was 8.7 kWh, compared with 12.6 kWh for FCFS, 11.8 kWh for Round Robin, and 10.4 kWh for the heuristic method. This reduction results from the agent's ability to consider the current operating condition of each resource before allocating a task. Instead of continuously directing tasks toward a single available resource, the learned policy distributes computational demand according to processor utilization, queue conditions, communication requirements, and estimated energy consumption. The proposed approach also achieved the lowest average task completion time, illustrating that energy reduction did not require a substantial sacrifice in responsiveness. The task success rate was approximately 98.2%, while deadline violations were lower than those observed with the comparison methods. These results demonstrate the advantage of treating energy consumption and execution performance as interconnected scheduling objectives rather than optimizing them independently.

Table 5: Illustrative Overall Performance Comparison

Scheduling Method	Energy Consumption (kWh)	Average Completion Time (ms)	Resource Utilization (%)	Task Success Rate (%)
FCFS	12.6	148	71.5	91.3
Round Robin	11.8	134	75.8	93.1
Heuristic Energy-Aware	10.4	116	81.6	95.4
Proposed Deep RL	**8.7**	**94**	**87.9**	**98.2**

The reduction in energy consumption is primarily associated with adaptive resource selection. Conventional scheduling strategies generally make decisions according to task order or predefined allocation rules and may not fully account for the instantaneous energy condition of the computing environment. For example, assigning a computationally demanding task to an underutilized edge server may be more energy-efficient than transferring it to the cloud, particularly when the communication distance is significant. In other situations, an overloaded edge server may require a longer execution period, causing additional energy expenditure and task delay. The proposed agent learns these relationships through repeated interaction with the environment. Consequently, it can select cloud resources when their additional computational capacity offsets communication and transfer costs, while favouring edge resources when proximity and lower latency provide greater overall efficiency. This dynamic selection contributes to the observed reduction in energy consumption.

The workload analysis provides further evidence of the adaptive behaviour of the proposed scheduler. Three workload conditions low, medium, and high were considered to examine how scheduling performance changed as task arrival intensity increased. Under low workload conditions, differences between scheduling methods were relatively

limited because sufficient computational resources were available. As workload intensity increased, however, the advantages of adaptive resource allocation became more pronounced. The proposed approach was able to distribute tasks across available resources according to their current state rather than maintaining a fixed allocation pattern. Under high workload conditions, the illustrative energy consumption remained lower than the baseline approaches, while task completion performance declined less severely. This suggests that reinforcement learning can learn resource-management policies that are useful during periods of resource contention.

Table 6: Illustrative Performance under Different Workload Conditions

Workload Level	Method	Energy Consumption (kWh)	Average Delay (ms)	Deadline Violation (%)
Low	Heuristic	7.2	83	3.8
Low	Proposed Deep RL	**6.4**	**71**	**2.1**
Medium	Heuristic	10.1	114	6.7
Medium	Proposed Deep RL	**8.3**	**91**	**3.4**
High	Heuristic	14.8	167	13.5
High	Proposed Deep RL	**11.9**	**132**	**7.6**

Another important observation concerns resource utilization. Efficient task scheduling should prevent both underutilization and excessive concentration of tasks on individual resources. When a scheduler repeatedly selects the same edge server because it is initially available, the server may become congested while other resources remain underused. The proposed learning agent incorporates CPU utilization, queue length, and resource availability into its state representation. Consequently, the allocation policy can redirect tasks toward alternative resources when congestion begins to develop. The illustrative resource utilization of 87.9% indicates more effective use of the available infrastructure than the baseline methods. However, high utilization should not be interpreted as an objective to maximize without restriction. Operating resources continuously near their maximum capacity can increase execution delay, energy consumption, and failure probability. The results therefore suggest that the learned policy seeks a practical utilization level rather than simply maximizing processor occupancy.

The deadline analysis is particularly relevant for applications such as real-time monitoring, industrial control, connected healthcare, autonomous systems, and interactive services. A scheduling method that minimizes energy but frequently misses application deadlines would not be suitable for such environments. The proposed reward function consequently imposes a penalty when tasks exceed their specified deadlines. The illustrative results indicate that deadline violations remained substantially lower under the proposed method, particularly during medium and high workload conditions. This improvement can be attributed to the agent's ability to consider estimated execution time and queue delay when selecting resources. Tasks with strict deadlines can therefore be directed toward less congested edge nodes or sufficiently powerful cloud resources, depending on network conditions. This demonstrates that energy-aware scheduling does not necessarily require sacrificing quality of service when multiple objectives are incorporated into the learning process.

The impact of adaptive learning was examined by comparing the proposed model with a static deep learning scheduling configuration. A static model can perform well when the workload distribution remains similar to the training environment but may experience degradation when resource availability and task arrival patterns change. The adaptive approach continuously receives updated state information, enabling it to modify subsequent scheduling decisions according

to current conditions. When sudden task bursts were introduced, the proposed policy could distribute incoming tasks among alternative resources rather than following the allocation pattern learned under normal workload conditions. Similarly, when an edge server became heavily loaded, the scheduling agent could favour other available resources. This behaviour demonstrates the principal value of reinforcement learning in dynamic environments: scheduling decisions are derived from the current state rather than being permanently fixed after model training.

An additional experiment examined the effect of network conditions on cloud–edge scheduling. Communication delay and available bandwidth can substantially influence whether a task should remain at the edge or be transferred to the cloud. The proposed model considers these variables when making scheduling decisions. Under favourable network conditions, cloud execution can become attractive for computationally intensive tasks because the additional processing capacity may compensate for transfer overhead. When bandwidth decreases or communication latency increases, the model can favour nearby edge resources. This adaptive behaviour is important because a scheduling policy based solely on processor performance may produce inefficient decisions when communication conditions fluctuate.

The computational overhead associated with reinforcement learning was also considered. Although the learned policy can make scheduling decisions rapidly after training, model development requires additional computational resources compared with simple rule-based approaches. This represents an important trade-off. The training process can be conducted centrally or on capable edge infrastructure, while the resulting model can subsequently be deployed for inference with substantially lower computational requirements. Model compression, parameter reduction, and efficient neural architectures can further reduce inference overhead. Therefore, the additional training cost should be considered against the long-term benefits obtained through repeated adaptive scheduling decisions.

Table 7: Illustrative Effect of Adaptive Learning

Scenario	Static Deep RL	Proposed Adaptive Deep RL
Normal workload success rate (%)	97.1	**98.2**
Sudden workload burst success rate (%)	88.6	**94.7**
Deadline violation during burst (%)	14.2	**8.1**
Energy increase during burst (%)	27.5	**16.3**
Recovery after workload change	Slow	**Faster**

Overall, the results indicate that the proposed Energy-Aware Deep Reinforcement Learning approach can improve task scheduling by jointly considering energy consumption, execution delay, resource utilization, and deadline requirements. Its principal advantage is its ability to adapt resource-selection decisions according to changing system conditions. The findings also reveal that energy efficiency, latency, and resource utilization are closely connected: reducing energy consumption through inappropriate resource selection may increase delay, while aggressively minimizing delay may increase energy demand. The multi-objective reward structure enables the proposed agent to seek a more balanced operating point. Nevertheless, the effectiveness of the approach depends on the quality of training data, state representation, reward design, model architecture, and workload diversity. Real-world cloud–edge environments may introduce additional factors such as unpredictable network failures, heterogeneous hardware, task dependencies, security constraints, and large-scale resource availability. Future experiments should therefore validate the model using real edge infrastructures, larger workload traces, longer operating periods, and highly variable network conditions. Subject to such validation, the findings support the potential of adaptive deep reinforcement learning as a practical mechanism for reducing

energy consumption while maintaining efficient and reliable task execution across integrated cloud–edge computing systems.

CONCLUSION

The integration of cloud and edge computing has created a flexible computational environment, but the increasing diversity of workloads and resource conditions makes efficient task scheduling a challenging problem. This research proposed an Energy-Aware Deep Reinforcement Learning approach for adaptive task scheduling that considers energy consumption, execution time, resource utilization, communication conditions, and task deadlines during resource selection. The proposed approach models scheduling as a sequential decision-making process in which a learning agent observes the current state of tasks and available computational resources before selecting an appropriate edge or cloud resource. By incorporating energy-related factors into the reward mechanism, the learning agent is encouraged to avoid unnecessary power consumption while maintaining satisfactory task execution performance. The adaptive nature of the approach enables scheduling decisions to respond to changing workload intensity, processor utilization, queue conditions, and network characteristics. Rather than relying on a fixed allocation strategy, the learned policy can distribute tasks among available resources according to their current operating conditions. This provides a more flexible basis for managing heterogeneous workloads and can help reduce the imbalance between computational efficiency and energy sustainability.

Despite its potential advantages, the proposed approach also involves several challenges that provide opportunities for further investigation. Deep reinforcement learning requires suitable training environments, representative workload patterns, and carefully designed reward functions to produce reliable scheduling policies. Changes in resource availability, unexpected workload bursts, network fluctuations, and heterogeneous hardware can also influence scheduling performance. In addition, the computational cost associated with model training must be considered when deploying learning-based scheduling systems in resource-constrained edge environments. Future research can investigate lightweight deep reinforcement learning models, multi-agent scheduling strategies, transfer learning, federated learning, and model-compression techniques to reduce training and deployment overhead. Further studies should also evaluate the approach using real-world cloud–edge infrastructures, long-duration workload traces, and highly dynamic network conditions. Overall, energy-aware deep reinforcement learning provides a promising direction for adaptive task scheduling because it enables computational resources to be selected according to current system conditions while simultaneously considering energy and performance requirements. The approach can contribute toward more sustainable, responsive, and intelligent cloud–edge infrastructures capable of efficiently managing the growing computational demands of distributed applications.

REFERENCES

1. *Cai, Jun, et al. "Task Decomposition and Hierarchical Scheduling for Collaborative Cloud-Edge-End Computing." IEEE Transactions on Services Computing, vol. 17, no. 6, 2024, pp. 4368–4382.*
2. *Chen, Junyan, and Lei Jin. "Deep Reinforcement Learning Method for Task Offloading in Mobile Edge Computing Networks Based on Parallel Exploration with Asynchronous Training." Mobile Networks and Applications, vol. 30, 2025, pp. 717–735.*
3. *Chen, Juan, et al. "MS_M2ATD3: A Master-Slave Multi-Agent Reinforcement Learning for Energy-Efficient Task Offloading in LEO Satellite Edge Computing." Ad Hoc Networks, vol. 178, 2025, article 103987.*

4. *Choppara, Prashanth, and Sudheer Mangalampalli. "An Efficient Deep Reinforcement Learning Based Task Scheduler in Cloud-Fog Environment." Cluster Computing, vol. 28, 2025, article 67.*
5. *Fan, Wenhao, et al. "Hybrid Deep Reinforcement Learning-Based Task Offloading for D2D-Assisted Cloud-Edge-Device Collaborative Networks." IEEE Transactions on Mobile Computing, vol. 23, no. 12, 2024, pp. 13455–13471.*
6. *Hou, Huanhuan, and Azlan Ismail. "EETS: An Energy-Efficient Task Scheduler in Cloud Computing Based on Improved DQN Algorithm." Journal of King Saud University Computer and Information Sciences, vol. 36, no. 8, 2024, article 102177.*
7. *Ismail, Ahmed A., Nour Eldeen Khalifa, and Reda A. El-Khoribi. "A Survey on Resource Scheduling Approaches in Multi-Access Edge Computing Environment: A Deep Reinforcement Learning Study." Cluster Computing, vol. 28, 2025, article 184.*
8. *Jayanetti, Amanda, Saman Halgamuge, and Rajkumar Buyya. "Reinforcement Learning Based Workflow Scheduling in Cloud and Edge Computing Environments: A Taxonomy, Review and Future Directions." arXiv, 2024.*
9. *Liang, Jingyu, et al. "Deep Reinforcement Learning Based Reliability-Aware Resource Placement and Task Offloading in Edge Computing." Proceedings of the 2024 IEEE International Conference on Web Services, IEEE, 2024, pp. 686–695.*
10. *Liu, Chun, et al. "Task Scheduling and Resource Allocation Based on Reinforcement Learning in Edge Computing." Intelligent Transportation and Smart Cities, 2026.*
11. *Liu, Jiajia, et al. "Dynamic Task Offloading Scheme for Edge Computing via Meta-Reinforcement Learning." Computers, Materials & Continua, vol. 82, no. 2, 2025, pp. 2609–2635.*
12. *Liu, Shumei, et al. "Dependent Task Scheduling and Offloading for Minimizing Deadline Violation Ratio in Mobile Edge Computing Networks." IEEE Journal on Selected Areas in Communications, vol. 41, no. 2, 2023, pp. 538–554.*
13. *Liu, Yanchun, et al. "Energy Efficient Task Scheduling for Heterogeneous Multicore Processors in Edge Computing." Scientific Reports, vol. 15, 2025, article 11819.*
14. *Nieto, Gorka, et al. "Deep Reinforcement Learning Techniques for Dynamic Task Offloading in the 5G Edge-Cloud Continuum." Journal of Cloud Computing, vol. 13, 2024, article 94.*
15. *Pei, Xinglong, et al. "Multi-Resource Interleaving for Task Scheduling in Cloud-Edge System by Deep Reinforcement Learning." Future Generation Computer Systems, vol. 160, 2024, pp. 522–536.*
16. *Peng, Peng, et al. "A Survey on Computation Offloading in Edge Systems: From the Perspective of Deep Reinforcement Learning Approaches." Computer Science Review, vol. 53, 2024, article 100656.*
17. *Qin, Yi, et al. "Task Offloading Optimization in Mobile Edge Computing Based on a Deep Reinforcement Learning Algorithm Using Density Clustering and Ensemble Learning." Scientific Reports, vol. 15, 2025, article 211.*

18. *Tang, Jiangbo. "Deep-Reinforcement-Learning–Guided Resource Allocation and Task Offloading for 6G Edge Intelligence." Computer Communications, vol. 245, 2026, article 108364.*

19. *Theodoropoulos, Theodoros, Eleftheria Papageorgiou, and Konstantinos Tserpes. "Energy-Efficient Task Offloading in Edge Computing: A Survey of Deep Reinforcement Learning Approaches." Proceedings/Research Repository, 2025.*

20. *Wang, Shudong, et al. "Energy-Efficient Collaborative Task Offloading in Multi-Access Edge Computing Based on Deep Reinforcement Learning." Ad Hoc Networks, vol. 163, 2024, article 103743.*

21. *Wen, Mengyao, et al. "Deep Reinforcement Learning for Energy-Efficient Workflow Scheduling in Edge Computing." Computer Networks, vol. 274, 2026, article 111790.*

22. *Wu, Zhoupeng, Zongpu Jia, Xiaoyan Pang, and Shan Zhao. "Deep Reinforcement Learning-Based Task Offloading and Load Balancing for Vehicular Edge Computing." Electronics, vol. 13, no. 8, 2024, article 1511.*

23. *Wu, Haixing, et al. "Deep Reinforcement Learning-Based Online Task Offloading in Mobile Edge Computing Networks." Information Sciences, vol. 654, 2024, article 119849.*

24. *Xie, Mande, Jiefeng Ye, Guoping Zhang, and Xueping Ni. "Deep Reinforcement Learning-Based Computation Offloading and Distributed Edge Service Caching for Mobile Edge Computing." Computer Networks, vol. 250, 2024, article 110564.*

25. *Yao, Rui, et al. "Dynamic Task Offloading Strategy in Mobile Edge Computing Using Meta-Reinforcement Learning with Adaptive Learning Rate Adjustment." Journal of King Saud University Computer and Information Sciences, vol. 38, 2026, article 318.*

26. *Yang, Wentao, et al. "Deep Reinforcement Learning-Based Low-Latency Task Offloading for Mobile-Edge Computing Networks." Applied Soft Computing, vol. 166, 2024, article 112164.*

27. *Zhang, Wenfan, and Haijiao Ou. "Reinforcement Learning Based Multi-Objective Task Scheduling for Energy Efficient and Cost Effective Cloud Edge Computing." Scientific Reports, vol. 15, 2025, article 41716.*

28. *Zhou, Xu, et al. "Deep Reinforcement Learning-Based Resource Scheduling for Energy Optimization and Load Balancing in SDN-Driven Edge Computing." Computer Communications, vols. 226–227, 2024, article 107925.*

29. *Zhu, Keyu, et al. "An Energy-Efficient Dynamic Offloading Algorithm for Edge Computing Based on Deep Reinforcement Learning." IEEE Access, vol. 12, 2024, pp. 127489–127506.*

30. *He, Yifan, et al. "Deep Reinforcement Learning with Evolved Actions for Dynamic Workflow Scheduling in Distributed Fog Computing." Neurocomputing, vol. 677, 2026, article 133115.*