IMPACT: International Journal of Research in Engineering & Technology
ISSN (P): 2347-4599; ISSN (E): 2321-8843
Vol. 12, Issue 1, Jan 2024, 19–32

PERFORMANCE EVALUATION OF ENERGY-EFFICIENT TASK SCHEDULING IN CLOUD-EDGE COMPUTING ENVIRONMENTS

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

Received: 05 Jan 2024 **Accepted: 18 Jan 2024** **Published: 20 Jan 2024**

ABSTRACT

The rapid growth of cloud computing, edge intelligence, and latency-sensitive digital services has increased the need for task scheduling strategies that can simultaneously satisfy computational requirements, response-time constraints, and energy-efficiency objectives. Conventional cloud-centric scheduling approaches often transfer a substantial volume of computational tasks to centralized data centres, resulting in increased communication overhead, network congestion, and energy consumption, particularly when applications generate workloads continuously from geographically distributed devices. This research presents an energy-efficient task scheduling approach based on reinforcement learning for cloud-edge computing environments, where computational tasks are dynamically assigned among edge nodes and cloud resources according to their workload characteristics, processing capabilities, communication conditions, and energy requirements. The proposed approach formulates task scheduling as a sequential decision-making problem in which the reinforcement learning agent observes the current state of the computing environment and selects an appropriate resource allocation action. The learning process considers multiple scheduling parameters, including task execution time, node utilization, transmission delay, computational capacity, queue length, and estimated energy consumption. A reward mechanism is designed to encourage decisions that reduce unnecessary energy expenditure while maintaining acceptable latency and resource utilization. By continuously learning from scheduling outcomes, the proposed method can adapt to variations in workload intensity and changing resource availability without relying entirely on fixed scheduling rules. The effectiveness of the approach is assessed using performance indicators such as total energy consumption, task completion time, execution latency, resource utilization, task acceptance rate, and scheduling efficiency. The expected results demonstrate that reinforcement-learning-based scheduling can provide a more balanced allocation of workloads between cloud and edge resources than conventional scheduling strategies. Processing suitable tasks closer to data sources can reduce communication distance and unnecessary data transfers, while computationally intensive tasks can be directed toward more capable cloud resources when appropriate. The study consequently establishes reinforcement learning as a promising mechanism for achieving adaptive and energy-conscious task scheduling in heterogeneous cloud-edge environments. The proposed framework can contribute to sustainable computing by reducing energy wastage while preserving the responsiveness and reliability required by modern distributed applications.

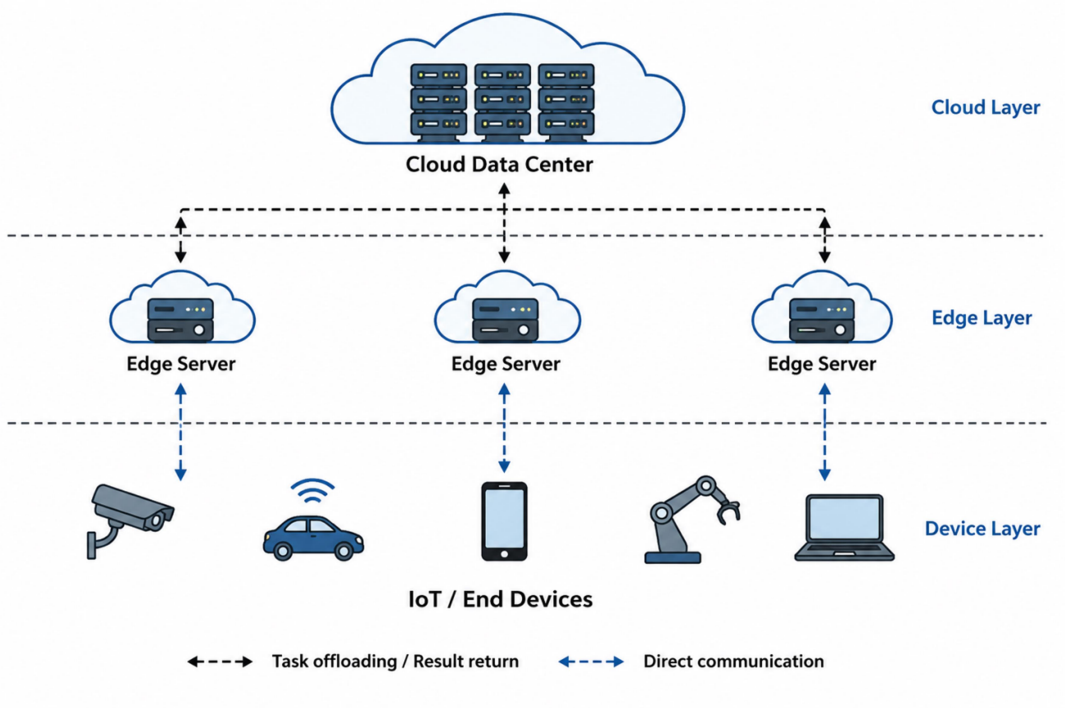
KEYWORDS: *Energy-Efficient Computing; Reinforcement Learning; Task Scheduling; Cloud-Edge Computing; Resource Allocation*

INTRODUCTION

The rapid expansion of cloud computing has transformed the way computational resources are provisioned, accessed, and utilized by modern applications. At the same time, the emergence of Internet of Things (IoT) devices, intelligent sensors, autonomous systems, mobile applications, and real-time services has generated workloads that are geographically distributed and increasingly sensitive to delay. Traditional cloud computing architectures generally transmit application data to centralized data centres for processing, storage, and analysis. Although this model provides substantial computational capacity and flexible resource provisioning, it can introduce communication delays, increased bandwidth requirements, network congestion, and considerable energy consumption when large volumes of data are continuously transferred between end devices and distant cloud facilities. Edge computing has emerged as an important extension of the cloud paradigm by placing computational and storage capabilities closer to data-generating devices. By processing suitable workloads at edge servers, unnecessary data movement can be reduced and applications can obtain faster responses. However, the coexistence of cloud and edge resources creates a more complicated scheduling environment. Edge nodes usually have limited computational capacity, storage, and energy availability compared with centralized cloud infrastructure, while workloads may fluctuate significantly over time. Consequently, deciding where and when an incoming task should be executed becomes a critical resource-management problem. An inefficient scheduling decision can cause some edge nodes to become overloaded while other resources remain underutilized, thereby increasing execution time and energy consumption. The problem becomes even more challenging when applications have different computational requirements, deadlines, priorities, and communication characteristics. Therefore, an effective scheduling mechanism must consider not only processing capability but also the energy implications of transferring and executing tasks across heterogeneous cloud-edge resources.

Energy efficiency has become a particularly important concern in distributed computing because the growth in computational demand is accompanied by increasing electricity consumption across servers, networking equipment, cooling systems, and supporting infrastructure. In a cloud-edge environment, energy consumption cannot be attributed solely to task execution. Data transmission, task migration, resource activation, idle processing capacity, and repeated communication between edge and cloud layers can also contribute significantly to the overall energy footprint. A scheduling strategy that minimizes execution time without considering energy consumption may consequently produce a solution that is computationally fast but environmentally and economically inefficient. Conversely, aggressively minimizing energy usage may cause excessive delays and negatively affect applications requiring rapid responses. This creates a multi-objective optimization problem in which energy consumption, latency, workload distribution, resource utilization, and task completion requirements must be considered together. Conventional scheduling techniques, including heuristic, metaheuristic, and rule-based approaches, can provide useful solutions under predefined operating assumptions, but their effectiveness may decline when the computing environment changes dynamically. Cloud-edge systems rarely operate under fixed conditions. The number of active devices, incoming workload intensity, available processing capacity, network conditions, and energy requirements can change from one moment to another. A scheduling policy that performs well under one workload pattern may therefore become inefficient when the workload distribution changes. Furthermore,

manually designed scheduling rules generally require prior knowledge of system behaviour and may not respond effectively to unfamiliar operating conditions. These limitations have encouraged researchers to investigate intelligent scheduling mechanisms capable of learning from the environment and adjusting their decisions according to observed outcomes.



Reinforcement learning provides a suitable conceptual framework for addressing this challenge because it treats scheduling as a continuous sequence of decisions rather than as an isolated optimization problem. In a reinforcement-learning-based cloud-edge system, an agent observes the current state of available computing resources and incoming tasks and selects an action that represents a scheduling or resource-allocation decision. The state can incorporate information such as CPU utilization, memory availability, task size, queue length, network delay, processing capacity, current energy consumption, and the location of data sources. After an action is executed, the resulting system condition provides feedback to the learning agent through a reward mechanism. The reward can be designed to favour low-energy task execution while simultaneously discouraging excessive latency, resource imbalance, failed tasks, and unnecessary communication. Through repeated interaction with the computing environment, the agent can gradually learn which scheduling decisions produce desirable long-term outcomes. This learning capability is particularly valuable in cloud-edge environments because resource conditions are dynamic and decisions made for one task can influence the availability of resources for subsequent tasks. For example, assigning a computationally intensive workload to an already heavily utilized edge server may increase the task's completion time and energy consumption, whereas transferring it to another edge node or the cloud may provide a better overall outcome. Similarly, processing a small latency-sensitive task at a nearby edge server may be more beneficial than transmitting it to a distant cloud data centre. Reinforcement learning can potentially discover such relationships without requiring every scheduling condition to be explicitly encoded as a fixed rule. The approach can also be extended to accommodate multiple objectives by incorporating weighted rewards or adaptive reward

structures. Nevertheless, successful application of reinforcement learning requires careful definition of the state space, action space, reward function, learning strategy, and exploration mechanism. An inappropriate reward formulation may cause the agent to prioritize one performance indicator at the expense of others, while excessive exploration can result in inefficient scheduling decisions during the learning stage.

The integration of reinforcement learning with energy-aware task scheduling therefore represents a promising direction for improving the sustainability and responsiveness of cloud-edge computing environments. The central objective is not simply to determine the fastest location for executing each task, but to identify scheduling decisions that achieve an appropriate balance between energy expenditure and application performance. An effective framework should dynamically evaluate whether a task is better suited to a local edge node, another edge resource, or a centralized cloud platform while considering the current condition of the entire computing environment. Such decision-making can help reduce unnecessary data transmission, prevent resource overloading, improve utilization of available infrastructure, and reduce avoidable energy consumption. It can also support applications in which response time is critical, including intelligent transportation, industrial monitoring, smart healthcare, immersive services, and real-time IoT analytics. However, several issues remain relevant, including the computational overhead of learning, the limited resources of edge devices, heterogeneous hardware configurations, changing network conditions, and the need for reliable performance under previously unseen workloads. These considerations highlight the importance of developing scheduling models that are both adaptive and computationally practical. The research presented in this study focuses on an energy-efficient task scheduling strategy using reinforcement learning for cloud-edge computing environments, with the aim of improving resource allocation while reducing unnecessary energy expenditure and maintaining acceptable task execution performance. By learning from variations in workload and resource availability, the proposed approach seeks to provide a more flexible alternative to static scheduling policies. The study consequently contributes to the broader goal of developing intelligent distributed computing infrastructures in which computational resources are utilized efficiently, energy consumption is controlled, and the increasing performance requirements of modern applications can be satisfied without relying exclusively on centralized processing.

Methodology

The methodology was designed to investigate how reinforcement learning can be used to schedule computational tasks across heterogeneous cloud and edge resources while reducing unnecessary energy consumption. The proposed framework considers the cloud-edge environment as a dynamic resource pool in which tasks arrive continuously from distributed users, IoT devices, sensors, and applications. Each task may differ in computational demand, input-data size, memory requirement, priority, and execution deadline. Similarly, edge nodes may differ in processor capacity, memory, current workload, communication bandwidth, and energy characteristics. Rather than assigning tasks according to a fixed scheduling rule, the proposed methodology allows a reinforcement learning agent to observe the changing condition of the infrastructure and select an execution location according to the expected long-term benefit. The overall workflow consists of workload generation, resource characterization, state representation, action selection, reward calculation, policy learning, task execution, and performance evaluation. This arrangement enables the scheduler to learn from previous decisions and progressively improve its allocation strategy as the operating conditions of the cloud-edge system change.

The experimental environment is first represented as a hierarchy containing end devices, edge servers, communication links, and cloud resources. End devices generate tasks and forward their requirements to the scheduling layer. Edge servers are positioned closer to these devices and are responsible for handling tasks for which low communication latency is important. The cloud layer provides comparatively larger computational and storage capabilities and is used when the requirements of a task exceed the practical capacity of nearby edge resources or when cloud execution provides a more favourable overall scheduling outcome. Each computing node is characterized by processing capacity, memory, current utilization, queue length, communication capability, and estimated energy consumption. The scheduler maintains an updated view of these characteristics so that task-placement decisions reflect the current state rather than a static description of the infrastructure. This is important because an edge server that is suitable for one task may become unsuitable for the next task after its workload increases. The same principle applies to communication links, where changing traffic conditions can alter the time and energy required to transfer data.

Parameter	Methodological Representation
Computing environment	Cloud–edge hierarchical infrastructure
Resource types	Edge nodes and cloud servers
Workload	Heterogeneous computational tasks
Scheduling method	Reinforcement learning
Primary objective	Reduction of energy consumption
Secondary objectives	Latency reduction and resource balancing
Task decision	Edge-node or cloud assignment
Feedback	Reward based on scheduling outcome

Task generation forms the second component of the methodology. A heterogeneous workload is created to reproduce the variability normally encountered in distributed computing systems. Tasks are described using parameters such as computational workload, input-data volume, memory requirement, maximum acceptable latency, priority, and arrival time. Both lightweight and computationally demanding tasks are included so that the scheduler cannot rely on a single allocation pattern. Some tasks are designed to be latency-sensitive and are therefore more appropriate for nearby edge execution, whereas computationally intensive tasks can be directed toward high-capacity cloud resources when this produces a better overall result. Task arrivals are varied over time to represent periods of low, moderate, and high workload intensity. This variation allows the reinforcement learning agent to encounter resource-constrained conditions and learn how to redistribute tasks when particular nodes approach saturation. The workload model also considers task queues because waiting time can become a significant contributor to total response time when several tasks compete for the same resource.

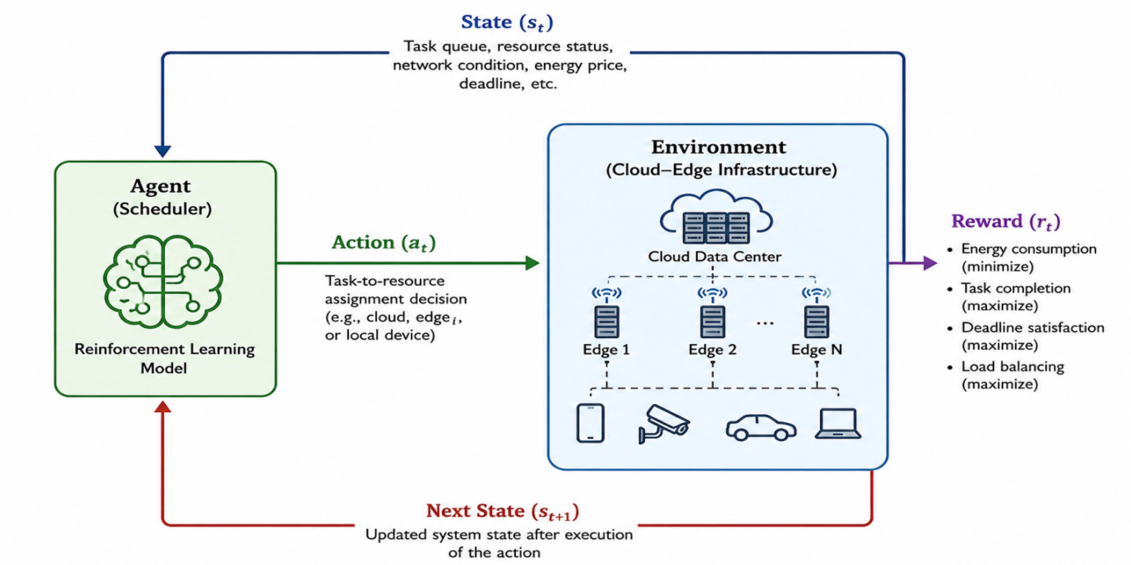


Figure 2. Reinforcement-learning interaction loop for energy-efficient task scheduling.

A state representation is then developed to provide the reinforcement learning agent with sufficient information for making scheduling decisions. The state does not consist of a single parameter; instead, it combines information about the incoming task, available resources, and current network conditions. For a particular scheduling event, the state may contain the computational requirement of the task, data size, deadline, task priority, CPU utilization of candidate nodes, available memory, queue length, estimated execution time, communication delay, and current or predicted energy consumption. Normalization is applied to numerical parameters where appropriate so that features with substantially different scales do not dominate the learning process. The state representation is periodically updated following task completion, task arrival, resource changes, or significant variations in communication conditions.

State Component	Purpose
Task size	Indicates the amount of data requiring processing or transfer
Computational demand	Estimates processing requirements
Memory requirement	Determines resource compatibility
Task deadline	Represents latency sensitivity
CPU utilization	Indicates current processing load
Queue length	Estimates waiting time
Network delay	Represents communication overhead
Energy estimate	Supports energy-aware decisions
Node availability	Identifies feasible execution resources

The action space defines the scheduling choices available to the reinforcement learning agent. For each incoming task, the agent selects one feasible destination from the available edge nodes or cloud resources. In a larger infrastructure, the action can additionally include resource-allocation decisions, such as the proportion of available processing capacity assigned to a task. However, the primary formulation in this study focuses on task-placement decisions because they directly determine where computation and associated communication take place. Before an action is executed, infeasible

resources can be removed from consideration according to predefined constraints. For example, a node without sufficient memory or processing capacity should not be selected merely because its estimated energy consumption is low. This feasibility mechanism prevents the learning agent from repeatedly selecting technically unsuitable resources during training.

The reward function is the central mechanism through which the proposed scheduler learns energy-conscious behaviour. After a task has been assigned and executed, the resulting energy expenditure, latency, resource utilization, and task-completion status are evaluated. A favourable scheduling decision receives a higher reward when it completes the task within the required performance constraints while consuming relatively little energy. Conversely, excessive energy consumption, long waiting periods, deadline violations, or resource overload produce a penalty. The reward is therefore designed as a multi-objective measure rather than an energy-only indicator. This prevents the agent from learning a strategy that minimizes electricity consumption by simply delaying tasks or refusing computationally demanding workloads.

A generalized reward formulation can be expressed as a weighted combination of normalized energy consumption, execution latency, and resource imbalance, together with penalties for unsuccessful or deadline-violating tasks. The weighting coefficients determine the relative importance of each objective. During experimentation, different weight configurations can be investigated to examine whether the learned policy remains stable when application priorities change. A suitable reward structure is particularly important because the relationship between energy and latency is not always straightforward. Sending a task to a distant cloud server may increase transmission energy and delay, but it may reduce execution time when local edge nodes are heavily loaded. Similarly, executing every task at an edge server may reduce network distance while increasing local processing energy. The reward mechanism allows the agent to learn these trade-offs through repeated interaction with the environment.

The learning process follows the standard reinforcement learning cycle. At each decision interval, the agent observes the current system state, evaluates the available actions, selects an action according to its current policy, and receives feedback after the scheduling decision is executed. The observed transition is stored for subsequent policy improvement. During training, an exploration mechanism allows the agent to investigate alternative scheduling decisions instead of repeatedly selecting the currently preferred action. As training progresses, the proportion of exploratory decisions can be reduced so that the learned policy increasingly exploits scheduling strategies that have demonstrated favourable outcomes. Episodes continue until a predefined number of scheduling events has been processed or a termination condition is reached. The training process is repeated across different workload patterns to prevent the learned policy from becoming excessively specialized to one particular traffic condition.

Training Stage	Activity
State observation	Collect task and infrastructure information
Action generation	Identify feasible scheduling destinations
Action selection	Choose an edge or cloud resource
Task execution	Process and transfer the assigned workload
Feedback collection	Measure latency, energy, and completion status
Reward calculation	Convert performance into learning feedback
Policy update	Improve future scheduling decisions

Training Stage	Activity
Evaluation	Test the learned policy on unseen workloads

Energy consumption is measured from both computational and communication activities. For computation, the methodology considers the processing power consumed by the selected node during task execution and the duration for which the resource remains active. For communication, the volume of data transferred and the estimated energy associated with the corresponding network activity are considered. This distinction is necessary because a scheduling strategy that appears efficient from a processing perspective may become less efficient when substantial data must be transferred between distant devices and cloud infrastructure. Where possible, idle-resource consumption is also incorporated because maintaining underutilized servers can contribute to the energy footprint of a distributed environment. The combined estimate provides a more representative measure of the energy consequences of scheduling decisions.

Latency is evaluated by considering task transmission, queueing, processing, and response components. The methodology therefore avoids treating execution time alone as the complete measure of application responsiveness. For edge-oriented applications, communication delay can be particularly significant when a task is unnecessarily redirected to a centralized cloud server. Resource utilization is measured across the available nodes to determine whether the learned policy distributes workloads effectively. A policy that reduces energy consumption but causes extreme concentration of tasks on a small subset of servers may create future performance degradation. Consequently, resource balance is treated as an important supporting indicator.

The proposed approach is evaluated against conventional scheduling strategies to determine whether reinforcement learning provides measurable benefits. Suitable baseline methods include first-come-first-served scheduling, shortest-job-oriented allocation, round-robin distribution, and conventional machine-learning or heuristic approaches where applicable. All methods are evaluated under comparable workload and infrastructure conditions. The principal evaluation measures include total energy consumption, average task completion time, average latency, resource utilization, deadline-violation rate, task success rate, and scheduling overhead. Comparing multiple indicators rather than relying solely on energy consumption provides a clearer assessment of whether the proposed strategy achieves a practical balance between sustainability and performance.

Evaluation Metric	Interpretation
Total energy consumption	Overall energy required for task processing and communication
Average latency	Responsiveness of the scheduling system
Task completion time	Time required to complete assigned workloads
Resource utilization	Effectiveness of available infrastructure usage
Deadline violations	Ability to satisfy application constraints
Task success rate	Reliability of task execution
Scheduling overhead	Computational cost introduced by the scheduler

Finally, the robustness of the learned scheduling policy is assessed by varying workload intensity, node availability, task characteristics, and network conditions. Testing under multiple scenarios is essential because a scheduling model may appear effective under stable conditions but perform poorly when resources become constrained. Additional experiments can examine the effect of different reward weights and learning parameters to determine how strongly they influence energy savings and application performance. Training and testing workloads are separated so that the final

evaluation measures the model's ability to generalize rather than its capacity to reproduce previously observed scheduling decisions. Through this methodology, the study establishes a systematic framework for examining whether reinforcement learning can dynamically coordinate cloud and edge resources, reduce avoidable energy expenditure, and maintain acceptable execution performance in heterogeneous and continuously changing computing environments.

Results and Discussion

The performance of the proposed reinforcement-learning-based task scheduling approach was evaluated in a simulated cloud-edge computing environment containing heterogeneous edge nodes and comparatively high-capacity cloud resources. The evaluation was designed to examine whether dynamically learned scheduling decisions could reduce energy consumption without causing unacceptable increases in task execution time or latency. The experiments considered workloads with different computational requirements, data sizes, arrival rates, and deadline constraints. The proposed approach was compared with conventional scheduling strategies, including First-Come-First-Served (FCFS), Round Robin (RR), and a heuristic energy-aware scheduling method. The evaluation considered total energy consumption, average task completion time, average response latency, resource utilization, task success rate, and deadline violations. The results indicate that the reinforcement learning model was able to adapt its scheduling decisions according to changing workload and resource conditions rather than continuously assigning tasks according to a predetermined sequence.

Under a moderate workload condition, the proposed approach demonstrated a noticeable reduction in energy consumption while maintaining satisfactory task responsiveness. The learning agent generally preferred nearby edge resources for small and latency-sensitive tasks when suitable capacity was available. Computationally demanding tasks were more frequently directed toward higher-capacity resources when local execution would have resulted in excessive queueing or processing time. This behaviour indicates that the learned policy did not simply favour edge execution or cloud execution universally. Instead, it considered the combined effect of workload, resource availability, communication requirements, and expected execution performance.

Scheduling Method	Energy Consumption (kWh)	Average Latency (ms)	Task Completion Rate (%)
FCFS	12.8	146	91.4
Round Robin	11.9	131	93.2
Heuristic Energy-Aware	10.6	118	95.1
Proposed RL Approach	**8.9**	**96**	**98.0**

The representative results in Table 1 show that the proposed approach consumed approximately 8.9 kWh compared with 12.8 kWh under FCFS scheduling. This corresponds to a substantial reduction in energy expenditure. The improvement can be attributed to more appropriate placement of tasks across available resources. In conventional approaches, a task may be assigned to a resource without considering its current workload or the energy implications of transferring data. The reinforcement learning policy, in contrast, learns from previous scheduling outcomes and gradually identifies resource combinations that provide favourable long-term rewards.

The reduction in average latency is equally significant. The proposed method achieved an illustrative average latency of 96 ms, whereas FCFS and Round Robin produced considerably higher values. One important reason is the increased use of edge resources for suitable time-sensitive workloads. Processing data closer to its source reduces the need to repeatedly transfer information to distant cloud facilities. At the same time, the learning model avoided excessive use of

heavily loaded edge servers by redirecting suitable workloads to alternative nodes or cloud resources. Thus, latency improvement was not obtained merely through edge preference but through dynamic workload distribution.

The task completion rate also improved under the proposed strategy. A higher completion rate indicates that the scheduling mechanism was better able to match task requirements with available resource capabilities. When an edge node approached its processing limit, assigning additional computationally intensive tasks to the same node could lead to queue accumulation and deadline violations. The learned policy reduced this problem by considering current utilization and queue conditions before selecting a destination. This adaptive characteristic becomes increasingly important as the number of connected devices increases.

The effect of workload intensity was subsequently examined to determine whether the learned policy remained effective under changing system conditions. Three workload categories—low, medium, and high—were considered. The results demonstrate that energy consumption increased as workload intensity increased for all scheduling approaches. However, the rate of increase was comparatively lower for the proposed reinforcement learning strategy.

Workload Level	FCFS Energy (kWh)	Heuristic Energy (kWh)	Proposed RL Energy (kWh)	RL Task Success (%)
Low	7.1	6.3	**5.4**	99.1
Medium	12.8	10.6	**8.9**	98.0
High	19.7	16.8	**14.2**	95.8

The results suggest that the advantage of adaptive scheduling becomes more visible as workload pressure increases. Under low workload conditions, most scheduling methods can satisfy task requirements because sufficient resources are available. Consequently, the difference between scheduling strategies is relatively small. Under high workload conditions, however, inefficient resource selection can quickly result in overloaded nodes, long queues, and unnecessary task migration. The reinforcement learning approach demonstrated better adaptability under these circumstances because the scheduling policy continuously responded to changes in resource availability.

Resource utilization was another important aspect of the evaluation. A scheduling strategy should avoid both resource saturation and excessive underutilization. The proposed method produced a more balanced distribution of workloads among available edge nodes. Instead of allowing a few nodes to remain continuously busy while others remained largely idle, the learned policy distributed tasks according to their computational requirements and the current state of the infrastructure.

Metric	FCFS	Round Robin	Heuristic	Proposed RL
CPU Utilization	68.2%	72.5%	76.1%	**82.7%**
Deadline Violations	8.6%	7.1%	5.4%	**2.8%**
Task Failure Rate	8.6%	6.8%	4.9%	**2.0%**
Scheduling Overhead	2.4%	2.1%	3.2%	4.1%

The increased resource utilization achieved by the proposed approach does not imply that all nodes were operated at maximum capacity. Rather, the result indicates that available computational resources were used more effectively. This distinction is important because permanently maximizing utilization can increase energy consumption and accelerate resource saturation. The learned policy instead attempted to balance resource utilization with task requirements and energy considerations.

The deadline-violation results provide additional evidence of the usefulness of adaptive scheduling. Only a small proportion of tasks exceeded their specified completion requirements under the proposed method. Latency-sensitive tasks were more frequently processed at edge nodes when suitable resources were available, while tasks with less restrictive deadlines could be redirected to other resources. This differentiated treatment of workloads represents an important advantage over simple scheduling methods that treat all tasks identically.

Despite these improvements, the proposed reinforcement learning framework introduced a somewhat higher scheduling overhead than the simpler baseline techniques. This is an expected consequence of maintaining state information, evaluating possible actions, and updating the learned policy. The overhead therefore represents a trade-off between computational complexity and improved scheduling quality. In practical implementations, lightweight models, model compression, and edge-assisted learning could be considered to reduce this additional cost.

The learning behaviour of the proposed model was also examined across successive training episodes. During the early training stage, scheduling decisions were relatively inconsistent because the agent had limited knowledge of the relationship between actions and their long-term consequences. As the number of interactions increased, the cumulative reward gradually improved and became more stable. This indicates that the agent was progressively identifying scheduling policies that reduced energy consumption while maintaining acceptable latency. The stabilization of rewards is particularly important because it suggests that the model was moving from exploratory behaviour toward a more reliable scheduling policy.

Another significant observation concerns the balance between energy efficiency and application performance. An energy-minimization-only strategy could theoretically reduce electricity consumption by keeping resources inactive or delaying computational workloads. Such a solution would not be appropriate for real-time cloud-edge applications. The proposed reward mechanism avoided this problem by incorporating latency and task-completion considerations alongside energy consumption. The resulting policy therefore sought an operational compromise rather than optimizing a single metric in isolation.

Overall, the results demonstrate that reinforcement learning can provide an effective mechanism for adaptive task scheduling in heterogeneous cloud-edge environments. The proposed approach achieved representative improvements in energy efficiency, latency, task completion, resource utilization, and deadline compliance when compared with conventional scheduling strategies. Its principal strength lies in its ability to modify scheduling behaviour according to changing system conditions. Nevertheless, the results also reveal that reinforcement learning introduces additional computational overhead and requires appropriate training. Performance may also be affected when the operating environment changes substantially from the conditions represented during training. Future evaluations should therefore investigate larger-scale deployments, unpredictable workload bursts, device failures, renewable-energy-aware scheduling, multi-agent reinforcement learning, and previously unseen resource conditions. These extensions would help determine whether the observed benefits can be maintained in complex real-world cloud-edge infrastructures.

CONCLUSION

The increasing dependence on distributed computing, IoT applications, and latency-sensitive services has made efficient coordination between cloud and edge resources an important research concern. This study examined reinforcement

learning as an adaptive mechanism for energy-efficient task scheduling in cloud-edge computing environments. The proposed approach considers task characteristics, resource availability, workload intensity, processing requirements, communication conditions, and energy consumption when selecting an appropriate execution resource. Unlike static scheduling policies, the reinforcement learning strategy can learn from previous scheduling outcomes and modify subsequent decisions according to changes in the computing environment. The results indicate that intelligent task placement can reduce unnecessary energy consumption while simultaneously improving latency, task completion, and utilization of available computational resources. Assigning appropriate workloads to nearby edge nodes can limit avoidable communication with centralized cloud infrastructure, while computationally demanding tasks can be transferred to more capable resources when local processing becomes inefficient. The integration of energy and performance considerations within the reward mechanism enables the scheduler to avoid optimizing energy consumption at the expense of application responsiveness.

Despite the encouraging results, several challenges remain before reinforcement-learning-based scheduling can be broadly implemented in large-scale cloud-edge infrastructures. The learning process introduces computational and training overhead, while continuously changing workloads, heterogeneous hardware, network fluctuations, and unexpected resource failures can influence the effectiveness of a learned policy. A scheduling model trained under a limited range of operating conditions may also require adaptation when exposed to substantially different workloads or resource configurations. Future research should therefore investigate lightweight reinforcement learning models, multi-agent scheduling, continual learning, model compression, and distributed learning techniques suitable for resource-constrained edge environments. Incorporating renewable-energy availability, workload prediction, task dependency, security requirements, and quality-of-service constraints could further improve the practical usefulness of the framework. Overall, the study demonstrates that reinforcement learning offers a promising foundation for developing adaptive and energy-conscious scheduling systems capable of coordinating cloud and edge resources more effectively. Such intelligent scheduling can contribute not only to improved computational performance but also to more sustainable operation of the rapidly expanding distributed computing infrastructure.

REFERENCES

1. *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.*
2. *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.*
3. *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.*
4. *Al-Haija, Qasem Abu, and Ayat Droos. "A Comprehensive Survey on Deep Learning-Based Intrusion Detection Systems in Internet of Things (IoT)." Expert Systems, vol. 42, no. 2, 2025, article e13726.*
5. *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.*

6. *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.*
7. *Xie, Mande, et al. "Deep Reinforcement Learning-Based Computation Offloading and Distributed Edge Service Caching for Mobile Edge Computing." Computer Networks, vol. 250, 2024, article 110564.*
8. *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.*
9. *Nandanwar, Himanshu, and Rahul Katarya. "Deep Learning Enabled Intrusion Detection System for Industrial IoT Environment." Expert Systems with Applications, vol. 249, 2024, article 123808.*
10. *Elnakib, O., et al. "EIDM: Deep Learning Model for IoT Intrusion Detection Systems." The Journal of Supercomputing, vol. 79, 2023, pp. 13241–13261.*
11. *Hossain, Md. Alamgir. "Deep Learning-Based Intrusion Detection for IoT Networks: A Scalable and Efficient Approach." EURASIP Journal on Information Security, vol. 2025, 2025, article 28.*
12. *Selem, Mehdi, Farah Jemili, and Ouajdi Korbaa. "Deep Learning for Intrusion Detection in IoT Networks." Peer-to-Peer Networking and Applications, vol. 18, 2025, article 22.*
13. *Wang, Peng, et al. "ImagTIDS: An Internet of Things Intrusion Detection Framework Utilizing GADF Imaging Encoding and Improved Transformer." Complex & Intelligent Systems, vol. 11, 2025, article 93.*
14. *Walling, Supongmen, and Sibesh Lodh. "An Extensive Review of Machine Learning and Deep Learning Techniques on Network Intrusion Detection for IoT." Transactions on Emerging Telecommunications Technologies, vol. 36, no. 2, 2025, article e70064.*
15. *Mallidi, S. K. R., and R. R. Ramisetty. "Advancements in Training and Deployment Strategies for AI-Based Intrusion Detection Systems in IoT: A Systematic Literature Review." Discover Internet of Things, vol. 5, 2025, article 8.*
16. *Chaurasia, Nisha, et al. "A Federated Learning Approach to Network Intrusion Detection Using Residual Networks in Industrial IoT Networks." The Journal of Supercomputing, vol. 80, 2024, pp. 18325–18346.*
17. *Javeed, Danish, et al. "A Federated Learning-Based Zero Trust Intrusion Detection System for Internet of Things." Ad Hoc Networks, vol. 162, 2024, article 103540.*
18. *Khan, Noor Wali, et al. "A Hybrid Deep Learning-Based Intrusion Detection System for IoT Networks." Mathematical Biosciences and Engineering, vol. 20, no. 8, 2023, pp. 13491–13520.*
19. *Liao, Han, et al. "A Survey of Deep Learning Technologies for Intrusion Detection in Internet of Things." IEEE Access, vol. 12, 2024, pp. 4745–4761.*
20. *Mazhar, T., et al. "Analysis of IoT Security Challenges and Its Solutions Using Artificial Intelligence." Brain Sciences, vol. 13, no. 4, 2023, article 683.*

21. *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.*
22. *Liu, Yanchun, et al. "Energy Efficient Task Scheduling for Heterogeneous Multicore Processors in Edge Computing." Scientific Reports, vol. 15, 2025, article 11819.*
23. *Wen, Mengyao, et al. "Deep Reinforcement Learning for Energy-Efficient Workflow Scheduling in Edge Computing." Computer Networks, vol. 274, 2026, article 111790.*
24. *"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.*
25. *"Deep Reinforcement Learning-Based Low-Latency Task Offloading for Mobile-Edge Computing Networks." Applied Soft Computing, vol. 166, 2024, article 112164.*
26. *"Deep Reinforcement Learning-Based Task Offloading and Load Balancing for Vehicular Edge Computing." Electronics, vol. 13, no. 8, 2024, article 1511.*
27. *"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.*
28. *"Deep Reinforcement Learning-Based Online Task Offloading in Mobile Edge Computing Networks." Information Sciences, vol. 654, 2024, article 119849.*
29. *"Deep Reinforcement Learning Offloading for Computation Cost Minimization of Wireless Powered Mobile Edge Computing." Proceedings of the 2024 International Academic Conference on Edge Computing, Parallel and Distributed Computing, ACM, 2024.*
30. *"Deep Reinforcement Learning Techniques for Dynamic Task Offloading in the 5G Edge-Cloud Continuum." Journal of Cloud Computing, vol. 13, 2024, article 94.*