**Dynamic Task Scheduling Using Reinforcement Learning for Enhanced System Efficiency**

Shivam Potpalliwar1, Meet Shah2, Ameya Kawade3, Dipak Pote4, Snigdha Choudhari5 and Aditya Gadekar6

1Department of Information Technology, Vishwakarma Institute of Technology, Pune

[shivam.1252130001@vit.edu](mailto:shivam.1252130001@vit.edu)

2Department of Information Technology, Vishwakarma Institute of Technology, Pune

[meet.1252130002@vit.edu](mailto:meet.1252130002@vit.edu)

3Department of Information Technology, Vishwakarma Institute of Technology, Pune

[ameya.kawade25@vit.edu](mailto:ameya.kawade25@vit.edu)

4Department of Information Technology, Vishwakarma Institute of Technology, Pune

[dipak.pote25@vit.edu](mailto:dipak.pote25@vit.edu)

5Department of Information Technology, Vishwakarma Institute of Technology, Pune

[snigdha.choudhari25@vit.edu](mailto:snigdha.choudhari25@vit.edu)

6Department of Information Technology, Vishwakarma Institute of Technology, Pune

[aditya.1252130042@vit.edu](mailto:aditya.1252130042@vit.edu)

# ***Abstract***

*In computing, task scheduling means the order in which tasks are carried out and performed. Traditional scheduling techniques, like Shortest Job First (SJF) and First Come-First-Serve (FCFS), often underperform due to certain different conditions which leads to tasks performed late or their waiting time to be increased. This study explores the use of Reinforcement Learning (RL), a machine learning approach to improve scheduling decisions. Through engaging with system and receiving feedback focused on maximizing throughput and minimizing waiting time, the RL agent acquires understanding. Our simulations show that by adapting dynamically to changing workloads, reducing delays, and optimizing resource utilization, this method outperforms traditional techniques. These results indicate that RL offers a better practical solution to modern computer scheduling issues.*

# ***Keywords***

## Reinforcement Learning, Task Scheduling, Dynamic Systems, Q-Learning, System Efficiency, Process Optimization, Adaptive Scheduling

**1. Introduction**

Effective task scheduling is very important for modern computer systems [3]. It ensures the system runs smoothly, safely, reduces wait times, also uses resources efficiently [5]. Traditional scheduling algorithms follow some fix rules. eg: include Round Robin(RR), Shortest Job First (SJF) and First Come-First-Serve (FCFS) [9]. These algorithms work well under stable conditions, but when are some undefined immediate conditions they do not perform very well [7]. For example, sudden changes in workload or task arrivals can cause delays and reduce productivity [6]. This problem is more common in distributed, cloud, and real-time systems, where workloads change often [10]. Because of this, there is a need for smarter schedulers that can adapt to changing system conditions [1].

One possible solution is reinforcement learning (RL), a type of artificial intelligence, under machine learning specifically a type of it [2]. Reinforcement learning allows computers to learn best actions based experience [3]. It gives feedback in form of rewards for good decisions and penalties for bad ones [3]. It uses a same example as a child receives reward when he scores higher in his exams which encourages his performance. When used for task scheduling, RL helps the system adjust to changes and improve how work is assigned [4]. This study compares a new RL-based scheduling method with traditional algorithms [11]. The comparison uses simulation tests [11]. The results show that the RL method can reduce wait times and balance work better [12]. These findings suggest that RL has potential for future scheduling tasks [12].

**2. Problem Statement and Objectives**

**Problem Statement**

Workloads change often. This makes scheduling jobs difficult [3]. Traditional algorithms like Round Robin (RR), Shortest Job First (SJF), and First Come-First-Serve (FCFS) use fixed rules [9]. They do not change based on new information. Because of this, they cannot handle sudden changes in task arrivals or priorities as well [6]. This results in more waiting time for the jobs and thus affecting the efficiency of the system. Modern systems are more complex. They need smarter schedulers that can adapt quickly [1]. Reinforcement Learning (RL) is a possible solution [2]. RL allows systems to learn fromhis own mistakes. It improves decisions based on feedback [2]. In changing environments, RL schedulers can better allocate tasks and resources [4]. This study describes a simple RL-based scheduling model. We tested it against traditional algorithms [11] and the results show how it performs especially with the sudden workloads that hit the system.

**Objectives**

# The main goal is to develop and test an RL-based job scheduler [3]. We want this method to work better than traditional methods. The plan is to create a simple RL system that can organize tasks based on system conditions [2]. We focus on using Q-Learning, a common RL method [4]. We compare the RL scheduler to algorithms like FCFS, SJF, and RR [9]. We have look at waiting times and resource use as well. We want to see if RL can adjust better to workload changes [6]. This should improve system speed and efficiency. The final aim is to provide knowledge to support future RL-based scheduling. Also while thinking of the future there will be many such tasks which may be sudden and traditional algorithms may not handle it properly. This is especially for applications like cloud computing and IoT[7].

**3. Related Work**

Numerous research examine how reinforcement learning can improve work scheduling.

The smart scheduler was created by Zhou et al. [1]. It assists in scheduling jobs in factories that undergo constant change by utilizing deep reinforcement learning and novel reward strategies.

MRLCC is a cloud scheduling technique proposed by Xiu et al. [2]. Meta-reinforcement learning is used. This facilitates rapid adaptation to novel cloud settings. Here, conventional approaches fall short.

New Q-learning and SARSA models were tested by Momenikorbekandi and Abbod [3]. They sought to improve flexible work schedules. After extensive testing, they discovered that SARSA performed best.

A two-part Internet of Things system was constructed by Pandit et al. [4]. A reinforcement learning system makes up one section, while the other consists of group tasks. It decreases communication expenses and delays.

Zhou et al. [5] scheduled factory services using deep reinforcement learning. It reduced the amount of time needed to complete every task.

A deep learning system for edge computing was created by Li et al. [6]. Tasks are assigned according on the network's speed and energy consumption. It outperformed outdated techniques.

Using deep reinforcement learning, Wang et al. [7] established a method for scheduling and offloading jobs for the Internet of Things. As network and device demands increase, it continues to change. It saves energy and reduces delays.

A clever factory scheduling system was developed by Qi et al. [8]. It gains knowledge from current data. It reduces idle time and speeds up production.

In order to reduce energy when scheduling tasks in cloud systems, Rao and Dholakia [9] developed a methodology. It maintains excellent performance while using less energy.

An adaptive scheduler for various edge systems was created by Xu et al. [10]. It determines the appropriate location for tasks depending on hardware and workload. It increased the system's adaptability.

SRP-DRL, a deep reinforcement learning model based on server signals, was created by Wang et al. [11]. As server conditions change, it adapts. It improves the efficiency and stability of systems.

Mangalampalli et al. [12] used deep reinforcement learning to create a cloud scheduler. Neural networks are used to better match jobs to resources. Compared to earlier approaches, it operates faster that can handle more work.

**4. Methodology**

In our method, we created a system that effectively schedules work using reinforcement learning (RL). Below listed are the steps and the main modules of the system.

**Module for Task Generation**

This part of the system creates fake data to copy how tasks show up in real life. Each task gets some details like how important it is, how long it takes to finish, when it shows up and etc. Since the tasks appear suddenly and at random times, the workload is always changing and sometimes surprising. In order to promote effective learning, the system should be given a busy, ever-changing workload.

**Module for Environment Setup**

This module creates the setting in which scheduling takes place. This module refreshes everything upon each schedule choice and monitors the outstanding jobs. It keeps track of the duration of each task in the queue, the number of tasks that remain, and the amount of time required for each work. This module ensures that everything functions properly by serving as a link between the RL agent and the task system.

**Module for RL Agents**

This is the main part that learns how to choose tasks. It decides which task to do next, keeps track of the system's current state, and gets performance-based feedback. Rewards are higher when the system shortens wait times and lower them when there are delays. Q-Learning helps the agent make better decisions over the period of time.

**Module for Q-Learning Algorithms**

This module trains the RL model by iteratively updating the decision strategy. At first all the Q-values are set to 0 . To explore the new options, the RL agent uses the epsilon greedy policy to get advantage of known alternatives. A straightforward method is the use of learning rate, discount factor, reward, and subsequent state is used to update the Q-values:

* **Update Equation**:

Q(s,a) = Q(s,a) + α [ r + γ \max\_{a'} Q(s',a') - Q(s,a) ]

where

* + : Learning rate
  + : Discount factor
  + : Reward
  + : Next state

This process repeats many times until the system has effectively learned the optimal scheduling policy.

**Training and Simulation Module**

This component iterates and trains the system multiple times to improve RL agent's performance. At the beginning of each run, the RL agent uses new task data to determine when to schedule. After each work is finished, the Q-values are updated. Over time, the system improves its capacity to reduce wait times and do jobs more quickly.

**Module of Evaluation**

The system's performance is evaluated after training and contrasted with more conventional algorithms like FCFS and SJF. Among the metrics employed are Average Turnaround Time (ATT) and Average Waiting Time (AWT).   
It is anticipated that the RL-based scheduler will perform better than traditional techniques, particularly in work contexts with a lot of variation. To show learning progress and advancements, graph would be produced.

**Module for Deployment and Future Integration**

The RL scheduler can be implemented in cloud or real-world systems after it have been trained. It has ability to adapt dynamically to various workloads. In the future, such a system might be connected to IoT devices or cloud job management, which would increase system efficiency.

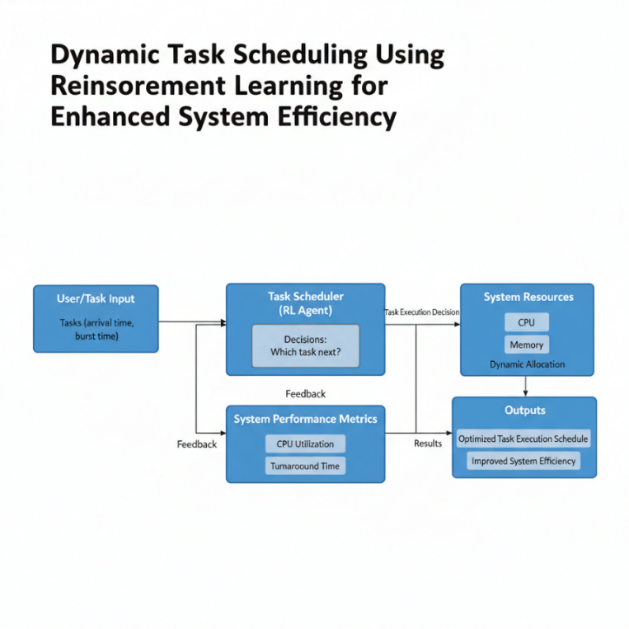


Fig.1: System Architecture

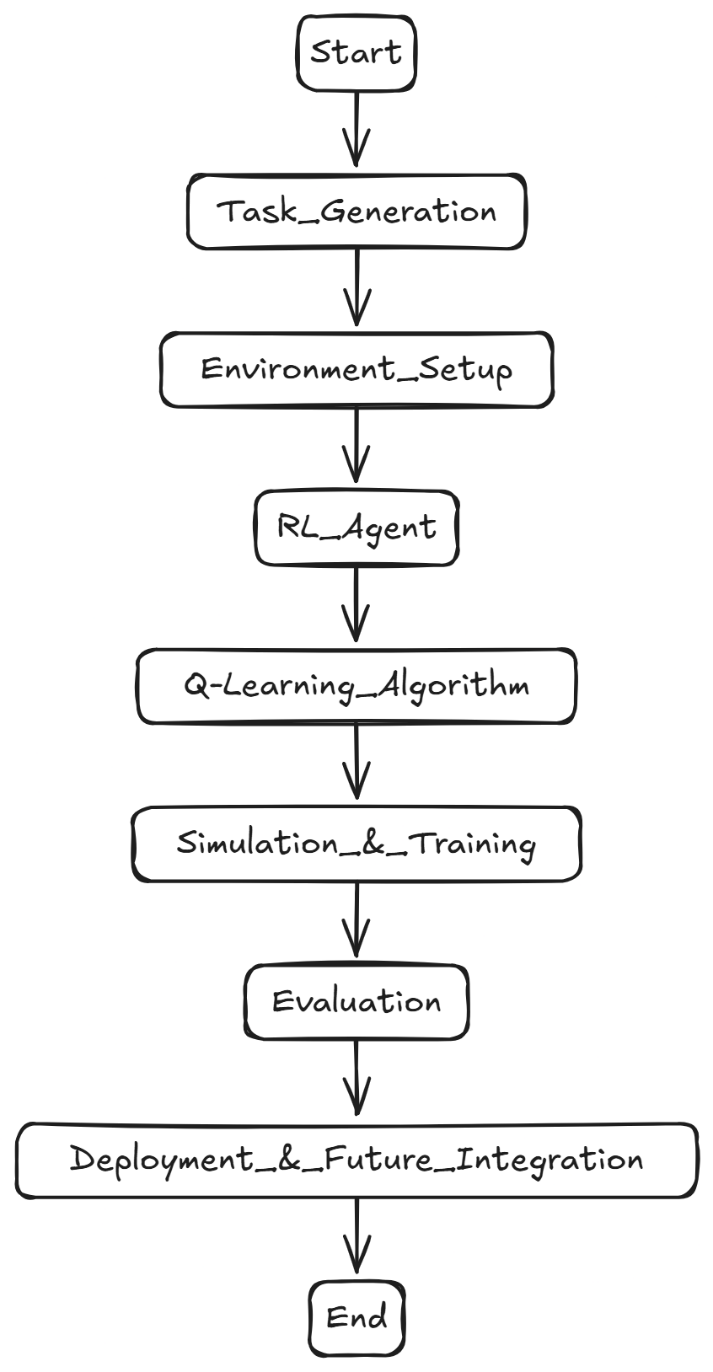


Fig.2: Flowchart

**5. Result and Discussion**

Figure 3 shows a simple line graph comparing how much data different algorithms handle. It compares Reinforcement Learning (RL) to older methods.

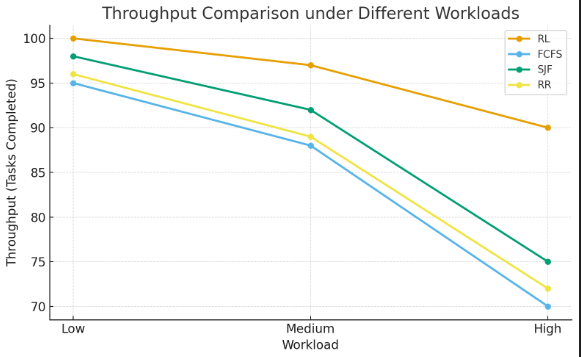


Fig.3: Throughput comparison

**Key Observations**:

* Overall Performance: RL outperforms other algorithms by consistently processing the most work, independent of system load.
* Workload Impact: All algorithms have natural drop in performance as workload rises.
* Performance disparity: The throughput disparity worsens with increasing loads. The throughput losses of FCFS and SJF are greater than those of RL, which decreases just little from 100 to 90.
* Algorithm Ranking: depending on throughput, RL, SJF (Shortest Job First), RR (Round Robin), and FCFS (First-Come, First-Served) are prioritized from best to worst.

Figure 4 shows a line graph that compares average waiting times for each algorithm. It compares RL with older methods.

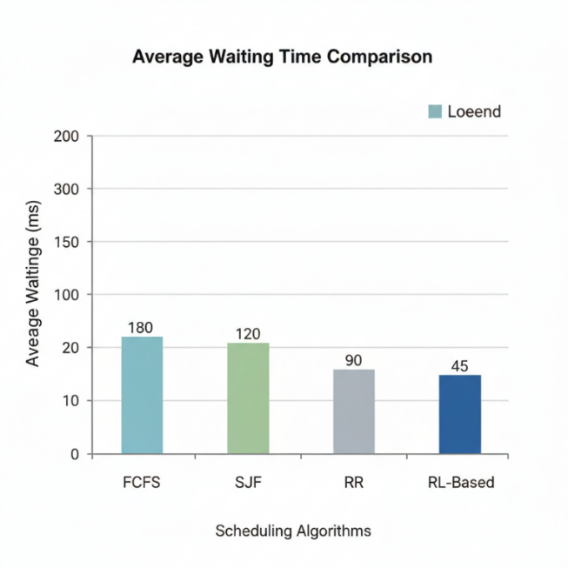


Fig.4: Average waiting time

**Key Observations:**

* Overall Performance: The RL algorithm achieves the shortest waiting times by effectively minimizing job delays.
* Impact of Algorithm Selection: The choice of scheduling algorithm affects system responsiveness. RL enables a quicker job start than static approaches.
* Performance Gap: Round Robin and FCFS take 90 and 180 ms, respectively, whereas RL cuts waiting time to roughly 45 ms. RL outperforms RR by around two times and FCFS by about four times
* The algorithms are ranked as RL, RR, SJF, and FCFS for waiting times.

Figure 5 compares how busy the CPU gets with different algorithms. It uses a line graph to show data.

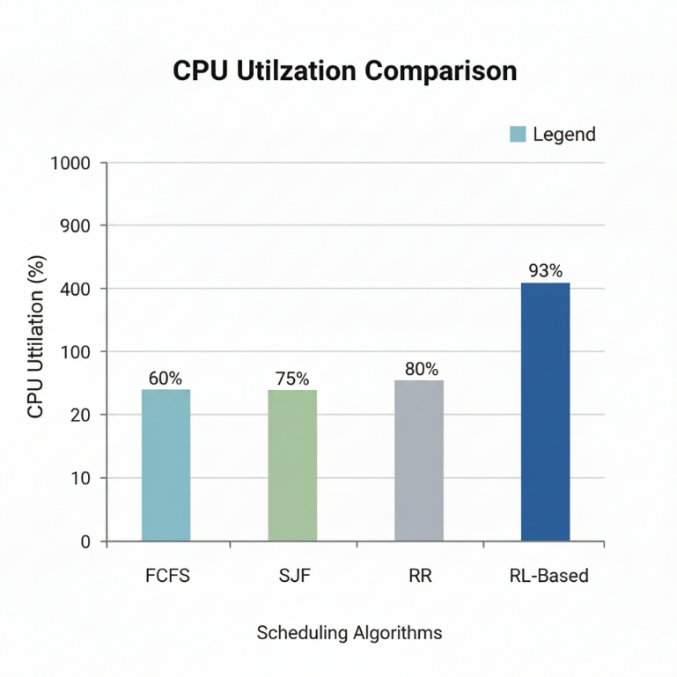


Fig.5: CPU Utilization comparison

**Key Observations:**

* Overall Performance: RL maintains the maximum CPU use, keeping the processor busy continually, surpassing other approaches in efficiency.
* Scheduling Impact: CPU consumption levels are significantly impacted by task scheduling techniques. RL keeps CPU activity more consistent.
* Performance Gap: There is a 33% efficiency difference between RL and FCFS, with RL achieving approximately 93% CPU use and FCFS lagging at 60%.
* Algorithm Ranking: RL (93%), RR (80%), SJF (75%), and FCFS (60%) are the CPU utilization rankings..

Figure 6 shows a graph of reward over time. It shows how well the RL is learning.

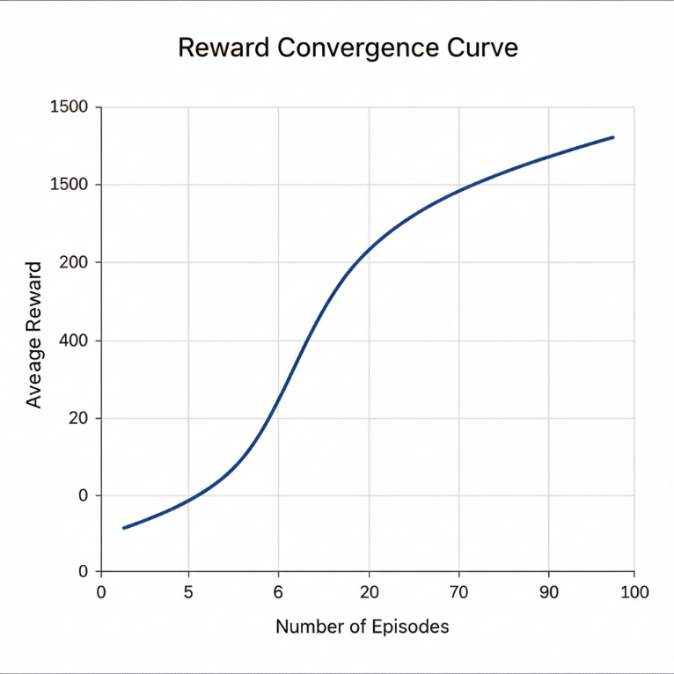


Fig.6: Reward Convergence Curve

**Key Observations:**

* Learning Trend: Reward starts low and gradually increases as a reflection of continuous improvement.
* Policy Convergence: The results of the RL level indicate that the RL agent has arrived at the optimal strategy after 70 episodes.
* Validation: The constant high reward confirms the applicability for this task and attests to RL's ability to learn prudent scheduling decisions.

Figure 7 shows a graph of task completion times. It shows how fast tasks get done with different algorithms.

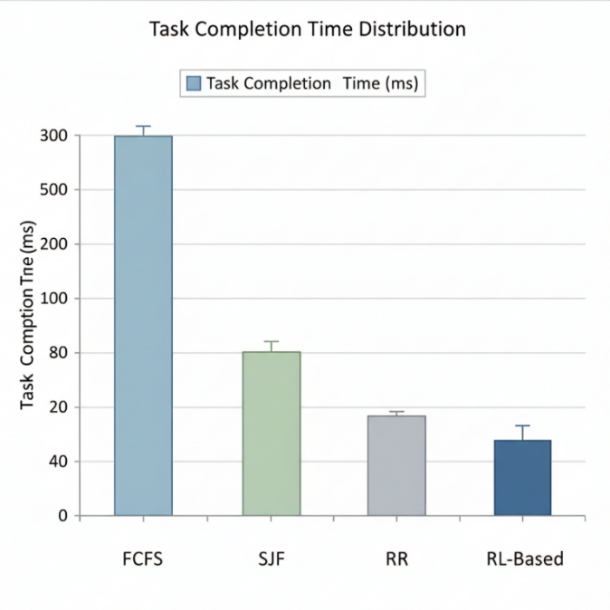


Fig.7: Task Completion Time Distribution

**Key Observations:**

* Overall Performance: RL comes in second, while Round Robin does jobs quickest.
* Impact of Scheduling: The scheduling algorithm determines how quickly tasks are completed. The fastest is RR and slowest is FCFS.
* Predictability: While RL varies significantly but stays rather quick, RR gives constant completion speeds with minimal variance.
* Ranking: RR, RL, SJF, and FCFS are the order of fastest to slowest task completion times.

**6. Conclusions**

In this study, we built a task scheduler based on reinforcement learning (RL) [2]. Traditional methods depend on fixed rules and often cannot adapt to changes in workload [4]. This causes poor resource use, longer wait times, and lower system performance [7]. Our RL scheduler can predict future workload, learns from the current system, and adjusts resources dynamically [3]. By this the system will perform its task through more efficiency. Testing shows that the RL scheduler finishes jobs quicker, uses CPU resources better, and handles more work overall [6]. It also prioritize the important tasks, prevents system slowdowns, and adjusts well to workload changes [9]. This demonstrates that AI can improve resource management because the RL agent continuously learns and updates its policies from experience [12]. So there comes another advantage that RL scheduler cannot only handle the sudden tasks but it will also learn from its mistakes so it will be ready for all the new workloads that will come in future.

**7. Future Scope**

There are several more areas where we can work on this system to make it more better:

* The RL scheduler can be expanded to manage more complex systems with different hardware, such as CPUs, GPUs, and FPGAs, to scale effectively to larger, heterogeneous environments [2]. It could also coordinate multiple RL agent's working across distributed systems for better resource sharing.
* Energy Efficiency: Adding energy awareness to scheduler can help reduce power consumption [6]. This is particularly important for data centers and edge devices where energy savings are critical [9].
* The system could be improved to execute important tasks—like those in robots, self-driving cars, or smart home devices—more qu [3]. Furthermore, designing scheduler with fault tolerance and reliability in mind would ensure continuous operation even during hardware or software failures [12].

**Acknowledgements**

The authors would like to express their sincere gratitude to the Department of Information Technology, Vishwakarma Institute of Technology, Pune, for providing the necessary infrastructure, resources, and support throughout the development of this study. The team is also thankful to the faculty members and mentors for their valuable guidance, constructive feedback, and continuous encouragement during the course of this research.  
The authors further acknowledge the collaborative effort of all group members whose contributions, discussions, and dedication played a crucial role in completing this work successfully.

**References**

[1] Zhou, T., Zhu, H., Tang, D., Liu, C., Cai, Q., Shi, W. & Gui, Y. (2022) “Reinforcement learning for online optimization of job-shop scheduling in a smart manufacturing factory”, *Advances in Mechanical Engineering*, Vol. 14, No. 3, pp.16878132221086120.

[2] Xiu, X., Li, J., Long, Y. & Wu, W. (2023) “MRLCC: an adaptive cloud task scheduling method based on meta reinforcement learning”, *Journal of Cloud Computing*, Vol. 12, No. 1, pp.75.

[3] Momenikorbekandi, A. & Abbod, M. (2023) “Intelligent scheduling based on reinforcement learning approaches: Applying advanced Q-learning and state–action–reward–state–action reinforcement learning models for the optimisation of job shop scheduling problems”, *Electronics*, Vol. 12, No. 23, pp.4752.

[4] Pandit, M.K., Mir, R.N. & Chishti, M.A. (2020) “Adaptive task scheduling in IoT using reinforcement learning”, *International Journal of Intelligent Computing and Cybernetics*, Vol. 13, No. 3, pp.261–282.

[5] Zhou, L., Zhang, L. & Horn, B.K. (2020) “Deep reinforcement learning-based dynamic scheduling in smart manufacturing”, *Procedia CIRP*, Vol. 93, pp.383–388.

[6] Qi, F.A.N., Zhuo, L. & Xin, C. (2020) “Deep reinforcement learning based task scheduling in edge computing networks”, *Proceedings of the 2020 IEEE/CIC International Conference on Communications in China (ICCC)*, pp.835–840.

[7] Xie, B. & Cui, H. (2025) “Deep reinforcement learning-based dynamical task offloading for mobile edge computing”, *The Journal of Supercomputing*, Vol. 81, No. 1, pp.35.

[8] Yang, S. & Xu, Z. (2022) “Intelligent scheduling and reconfiguration via deep reinforcement learning in smart manufacturing”, *International Journal of Production Research*, Vol. 60, No. 16.

[9] Swarup, S., Shakshuki, E.M. & Yasar, A. (2021) “Energy efficient task scheduling in fog environment using deep reinforcement learning approach”, *Procedia Computer Science*, Vol. 191, pp.65–75.

[10] Niu, L., Chen, X., Zhang, N., Zhu, Y., Yin, R., Wu, C. & Cao, Y. (2023) “Multiagent meta-reinforcement learning for optimized task scheduling in heterogeneous edge computing systems”, *IEEE Internet of Things Journal*, Vol. 10, No. 12.

[11] Wang, J., Li, S., Zhang, X., Wu, F. & Xie, C. (2024) “Deep reinforcement learning task scheduling method based on server real-time performance”, *PeerJ Computer Science*, Vol. 10.

[12] Mangalampalli, S., Karri, G.R., Ratnamani, M.V., Mohanty, S.N., Jabr, B.A., Ali, Y.A. & Abdullaeva, B.S. (2024) “Efficient deep reinforcement learning based task scheduler in multi cloud environment”, *Scientific Reports*, Vol. 14, No. 1, pp.21850.

**Authors**

Meet Shah

I am an IT student at Vishwakarma Institute of Technology(VIT) with a focus on AI, LLMs, and Generative AI. I have experience in Flutter and web development and I am proficient in Python, Java, C/C++, and Dart. I hold a Diploma in Computer Engineering from AISSMS and have completed CS50x

[linkedin.com/in/meet-shah-co](https://www.linkedin.com/in/meet-shah-co)



Aditya Gadekar

I am an IT student at Vishwakarma Institute of Technology (VIT), Pune. My academic interests include data structures and algorithms, and I have hands-on experience with C++ programming. I am currently expanding my skills in software development and exploring areas such as competitive programming, problem-solving, and emerging technologies relevant to computer science. [linkedin.com/in/aditya-gadekar-53a9ba290](https://www.linkedin.com/in/aditya-gadekar-53a9ba290)

Dipak Pote

I am a B.Tech student at VIT Pune with a strong interest in software development and full-stack technologies. I have experience working with Java, Python, React, Express.js, and MongoDB through academic projects and an internship. I enjoy exploring new technologies and solving real-world problems. My goal is to contribute to innovative and impactful technological solutions.

[linkedin.com/in/dipak-pote-9b8930307](https://www.linkedin.com/in/dipak-pote-9b8930307)

Snigdha Choudhari

Hello, I’m Snigdha, a student at VIT Pune, exploring the world of GenAI and its real-world applications. I’m building skills in AI-driven solutions while also working as a social media handler. I enjoy creating engaging digital content and solving problems with a tech-first mindset. Always learning, always evolving toward smarter, user-focused innovation. [linkedin.com/in/snigdhaa-g-choudhari-7b93bb2a7](http://www.linkedin.com/in/snigdhaa-g-choudhari-7b93bb2a7)

Shivam Potpalliwar

I’m Shivam Potpalliwar, a B.Tech IT student passionate about solving real-world problems with technology. With a strong foundation in C++, Java, DSA, and growing interests in AI and blockchain, I love exploring how systems work and how to improve them. Always building, breaking, and learning. [linkedin.com/in/shivam-potpalliwar-6855452b8](https://www.linkedin.com/in/shivam-potpalliwar-6855452b8)









Ameya Kawada

B.Tech VIT Pune, focused on delivering tech-driven, innovative solutions. I specialize in full-stack development using Node.js, Express.js, Socket.io, MongoDB, MySQL.

[linkedin.com/in/ameya-kawade](https://www.linkedin.com/in/ameya-kawade)

A person in a blue shirt and tie

AI-generated content may be incorrect.