SACJ 37(2) December 2025
Research Article

Reinforcement Learning algorithms for adaptive load-balancing for Web applications

Rana Zuhair Al-Shaikh[a], Muna M. Jawad Al-Nayar[a], Ahmed M. Hasan[a]

Control and Systems Engineering Department, University of Technology, Baghdad, Iraq

ABSTRACT
This research investigates the application of reinforcement learning (RL) to optimise load balancing in Nginx web applications. We developed a simulation environment on AWS to evaluate three enhanced RL algorithms: Epsilon-greedy, Upper Confidence Bound, and Proximal Policy Optimization (PPO) against classic methods (round-robin and Least Connections) under diverse load conditions, including normal loads, burst loads, server failures, and heterogeneous server instances. Our results demonstrate that RL, particularly PPO, significantly outperforms classic methods. Notably, PPO achieved up to a 30% increase in throughput, a 20% reduction in latency, and a 5% improvement in the successful message rate compared to the best-performing classic algorithm. These improvements were most pronounced under challenging conditions such as burst loads and server failures, highlighting the adaptability and resilience of RL-based load balancing.

Keywords Load-balancing, Reinforcement Learning, Proximal Policy Optimization, Epsilon Greedy, Upper Confidence Bound

Categories • **Computer systems organisation** ∼ **Architectures, Distributed architectures, Cloud computing**

Email
Rana Zuhair Al-Shaikh – cse.22.11@grad.uotechnology.edu.iq (CORRESPONDING)
Muna M. Jawad Al-Nayar – Muna.m.jawad@uotechnology.edu.iq
Ahmed M. Hasan – 60163@uotechnology.edu.iq

Article history
Received: 07 December 2024
Accepted: 10 July 2025
Online: 22 December 2025

1 INTRODUCTION

In recent years, due to the rapid expansion of the Internet, and the increasing complexity of web applications, the pressure on servers has increased significantly. Efficient road balancing ensures that an optimal level of performance, reliability, and scalability is achieved. Load balancing is the idea of distributing traffic between several servers. This reduces delays and helps the response to be faster, and on top of that, makes the service more available because it reduces service interruption (Wibowo et al., 2023).

Load balancing techniques can be divided into two types in general: classic and dynamic techniques.Classic techniques such as Round Robin (RR), least connections (LC), and IP hash,

Al-Shaikh, R.Z., Jawad Al-Nayar, M.M, and Hassan, A.M. (2025). Reinforcement Learning algorithms for adaptive load-balancing for Web applications . *South African Computer Journal* 37(2), 121–145. https://doi.org/10.18489/sacj.v37i2.20753

SACJ is a publication of *SAICSIT*. ISSN 1015-7999 (print) ISSN 2313-7835 (online)

depend on predefined rules. These techniques are easy to implement, but their problem is that they cannot easily adapt to the change of load or the difference in the capacity of servers over time. This can negatively affect the user experience or resource efficiency. On the other hand, dynamic techniques rely on more advanced mechanisms and make their decisions based on real state in real time, such as network status or server processor consumption. For this reason, these techniques are better when loads are unexpected or complex (Kopparapu, 2002).

Despite their usefulness, dynamic techniques are sometimes complex to implement and require more sophisticated algorithms and infrastructures. Some of the traditional dynamic techniques so far have been problematic when dealing with rapidly changing environments or when trying to achieve long-term improvement goals. For this reason, artificial intelligence or machine learning is used for better performance (Ghomi et al., 2017).

This study focuses on evaluating the usefulness of reinforcement learning (RL) algorithms in load balancing using web applications, especially three RL algorithms: Enhanced Epsilon-Greedy, Enhanced UCB, and PPO. We compare these algorithms with classic techniques.

When we say "adaptive" in research, we mean that the load balancer can dynamically alter its request-routing strategy in real-time, depending on the state of the environment. This is done by an agent who learns from feedback of the environment, such as CPU usage or response time.

The main contributions of this paper are:

RL Framework for Nginx: We developed and implemented a reinforcement learning framework specifically for adaptive load balancing with web applications, incorporating a tailored reward function and an adaptive exploration strategy.

Algorithm Comparison: We conducted a comparative evaluation of three enhanced RL algorithms – Enhanced Epsilon-Greedy (EEG), Enhanced Upper Confidence Bound (EUCB), and Proximal Policy Optimization (PPO) against classic load balancers (Round-Robin and Least Connections).

Scenario-Based Evaluation: We assessed the performance of these algorithms under diverse and realistic conditions using a simulated AWS environment, including normal load, burst load, server failure, and heterogeneous server instances.

Performance Gains: We demonstrated quantitatively that RL algorithms, particularly PPO, significantly outperform traditional methods, achieving up to 30% greater throughput, 20% lower latency, and a 5% higher successful message rate, especially under challenging conditions like burst loads and server failures.

The remainder of this paper is organised as follows: Section 2 reviews related work in load balancing. Section 3 describes the proposed methodology, including the RL framework and experimental setup. Section 4 presents and discusses the experimental results. Finally, Section 5 concludes the paper and suggests future research directions.

https://doi.org/10.18489/sacj.v37i2.20753

2 LITERATURE REVIEW

To conduct a focused and comprehensive review of the state of the art, a systematic search was performed using the IEEE Xplore digital library. The search was constrained to recent peer-reviewed publications covering the period from 2023 to 2025. To ensure high relevance and identify papers where our topics are the central theme, a specific query was constructed requiring the exact phrase "*load balance*" in the document title and the term "*reinforcement* in the Abstract. This targeted search strategy yielded a total of 68 articles. The following review synthesizes the key trends, methodologies, and application domains identified from this body of literature to establish a clear and defensible research gap for our work.

A review of the literature reveals that Reinforcement Learning (RL) is being applied to a wide and diverse range of load balancing problems. A significant portion of research focuses on the network infrastructure layer, particularly in Software-Defined Networks (SDN) (Coelho & Schaeffer-Filho, 2023; Jeong et al., 2024; Kumari et al., 2024; Shaikh, 2023; Y. Shi et al., 2023; H. Sun et al., 2025; Vaishnavi et al., 2023; X. Xu & Zou, 2024). Another major area of focus is general cloud computing environments (Adarsh & Bhargavi, 2023; Ghosh et al., 2024; Krishna & Khasim Vali, 2025; Nimmala et al., 2024), with many studies addressing task scheduling (Feng et al., 2024; Li et al., 2024), resource allocation (Lahande et al., 2023; Shahakar et al., 2024), and VM migration (Brahmam & Vijay Anand, 2024). Furthermore, RL-based load balancing is being actively explored in emerging and specialized domains, including Fog (Baek & Kaddoum, 2023; Choppara & Lokesh, 2025; C. Wu & Yang, 2024) and Multi-Access Edge Computing (MEC) (Chen et al., 2025; Hartman et al., 2023; D. Jiang et al., 2024; L. Liu et al., 2024; B. Shi & Chen, 2023; G. Wu et al., 2024; Y. Wu et al., 2025), Non-Terrestrial Networks such as LEO satellites (Badini et al., 2023; Q. Liu et al., 2023; Wang et al., 2025; Zhao et al., 2025), and Vehicular Networks (IoV/VEC) (He et al., 2025; Talpur & Gurusamy, 2023; Zhu et al., 2023). More recently, research has begun to address application-level contexts such as virtualized container environments and Kubernetes-based microservices (Santos et al., 2024; Yucel, 2023). Other niche applications include data center networks (Das et al., 2025; G. Jiang et al., 2023; Ye et al., 2023), cellular (Abouamasha et al., 2025) and mobile networks (Chang et al., 2023; Gupta et al., 2024; Nakagawa & Mizutani, 2023; G. Wu et al., 2024), and even electrical grids (J. Liu et al., 2023; Mosalli et al., 2025; Xie et al., 2024; Zhao et al., 2025), also, specialized scientific computing for tasks like parallel particle tracing (J. Xu et al., 2023). This broad application demonstrates the versatility of RL but also highlights that most research is directed at the network or infrastructure level rather than the application server level.

The methodologies employed are as varied as the application domains. While many studies refer to a general Deep Reinforcement Learning (DRL) approach (Johann et al., 2024; Kołakowski et al., 2024; D. Wu et al., 2023), several works specify particular algorithms. Value-based methods like Q-Learning (Zervopoulos et al., 2023) and its deep variants like DDQN (A. Sun et al., 2023) are present, as are modern policy-gradient (S. Tian et al., 2025) and actor-critic methods such as DDPG (Dinh et al., 2024; Han et al., 2023), TD3 (L. Liu et al., 2024), and SAC (Huang et al., 2024). Furthermore, researchers are exploring more ad-

vanced learning architectures, such as Graph Convolutional Networks (Fawaz et al., 2023), to better model the topological relationships between network nodes and enable more effective collaboration between agents. Notably, some research also explores the use of PPO (Ma et al., 2025; Y. Tian et al., 2023). A growing trend is the use of Multi-Agent Reinforcement Learning (MARL) to handle decentralized environments (Houidi et al., 2023), with some using specific algorithms like MADDPG (Han et al., 2023). Other advanced paradigms being investigated include Federated RL (J. Liu et al., 2025; Shang et al., 2025) for privacy-preserving learning and Inverse RL (Konar et al., 2023) to tackle reward function design.

Furthermore, many studies propose hybrid approaches that combine RL with other techniques (Attia et al., 2024). This includes integrating RL with traditional heuristics like K-shortest path (Takahashi & Shinomiya, 2024), metaheuristics like Ant Colony Optimization (Nimmala et al., 2024), or Generative AI models (He et al., 2025; Khoramnejad & Hossain, 2025). In contrast, some research tackles similar problems without using RL at all, opting instead for other machine learning techniques like supervised learning (Das et al., 2025), model predictive control (B. Tian et al., 2025), or non-ML data analytics (Jurše & Kuhar, 2024), highlighting that RL is one of several advanced solutions being considered.

While the application of RL to load balancing is clearly an active and fruitful area of research, our comprehensive review indicates that the majority of studies focus on optimizing network-level traffic or resource allocation in large-scale infrastructure like SDN, cellular networks, and general cloud platforms. There is a lack of research that applies and evaluates modern RL algorithms specifically for application-level load balancing of web servers. The challenges at this layer, which are more closely tied to application response times and server processing loads, are distinct from network-level metrics. Many of the reviewed studies also do not test the resilience of their proposed methods under crucial real-world operational scenarios such as sudden burst loads or unexpected server failures. Therefore, our work aims to address this specific gap by implementing and comparing multiple RL algorithms directly within an Nginx web server environment and evaluating their performance and resilience under a variety of realistic and challenging load conditions.

3 BACKGROUND

Load balancing for web servers is the process of distributing incoming network traffic (HTTP requests) across a pool of multiple web servers. Instead of one server dealing with all requests, the workload is shared among multiple servers (Hong et al., 2006). Load balancing strategies are fundamentally classified into two types: static and dynamic.

3.1 Classical Load Balancing Algorithms

Static Load Balancing: In this method, tasks are allocated to processing units based on a fixed set of rules, and these tasks do not change during running. A familiar example is the Round-Robin (RR) algorithm (Barlas, 2023).

https://doi.org/10.18489/sacj.v37i2.20753

Dynamic Load Balancing: These algorithms make their decisions based on the current state of the system, adapting to real-time changes in workload and resource availability. A well-known example is the "*least connections*" (LC) algorithm, which sends new requests to the server with the least active connections (Li et al., 2024).

3.2 Limitations of Classical Approaches and the Need for Reinforcement Learning

Although easy to implement, the classic load balancing methods often fight to deal with the dynamic nature of web flow (Barlas, 2023). For example, the static nature of the RR algorithm means that it does not take into account server capacity or current processing loads, which can lead to resource utilization imbalances. Although the LC algorithm is an improvement, it may not be optimal when there is significant difference in server capacity or request processing time (Li et al., 2024).

In general, these traditional techniques have little ability to adapt to changing flow patterns and system conditions over time, leading to suboptimal resource utilization and performance degradation (Feng et al., 2024). These drawbacks require more intelligent and dynamic load balancing methods. Reinforcement learning (RL) represents a promising path to achieving this ambition. The next subsection describes the concept of reinforcement learning in detail.

3.3 Reinforcement Learning

RL is a subfield of machine learning where agents learn optimal approaches by interacting with their environment (Ris-Ala, 2023). Agents make decisions over time in a setting to maximise total rewards. Unlike supervised or unsupervised learning, RL agents learn from their actions through trial and error, receiving either rewards or penalties as feedback. This learning approach enables RL agents to adapt their behaviours and discover the most effective strategies over time, even in complex and dynamic environments.Figure 1 shows the relations between the components of an RL system. The main components of the RL system are (Sanghi, 2021):

Agent: The learner and decision-maker interacting with the environment.

Environment: The outside world where the agent functions.

State: Information about the current state of environment.

Action: Selection action taken by the agent.

Reward: The calculation which impacts the action on the environment.

The agent performs an action in the environment and calculates the current state and reward which explain the impact of this action on it (Sanghi, 2021).

This work discusses the efficacy of three RL algorithms for achieving load balancing within web applications: Epsilon-greedy (EG) (Bulut, 2022), Upper Confidence Bound (UCB) (Auer,

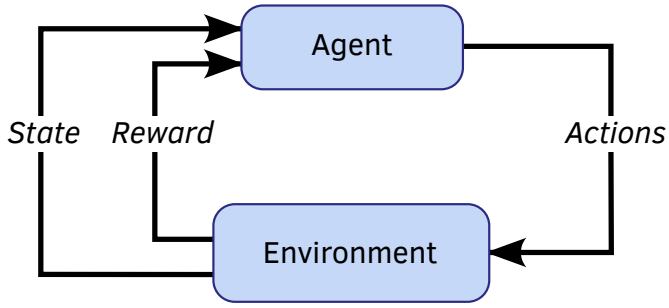


Figure 1: The main components of the Reinforcement Learning system[a]

[a] From (Sanghi, 2021)

2000), and Proximal Policy Optimisation (PPO) (Gu et al., 2022). EG includes two counting objectives: exploration, which means locating new routing policies, and exploitation, which means selecting previous successful policies. UCB uses less frequent actions that have demonstrated high returns. PPO represents a policy gradient method that iteratively improves an agent's policy based on its current estimate (Sanghi, 2021). This research aims to establish the effectiveness of these RL algorithms in optimising Nginx load balancing and provide insight into how this could lead to a fertile area for future studies, with the hope of creating a faster and more robust ecosystem of web applications.

Other research has effectively applied reinforcement learning (RL) to various domains, including adaptive load balancing in publish/subscribe systems, where it significantly improved performance under dynamic conditions (Al Shaikh et al., 2025). Other works have leveraged deep RL for novel feature selection in domain adaptation and utilized ANFIS-based RL strategies for controlling complex non-linear systems like coupled tanks (Mary et al., 2022; Naman & Ameen, 2022).

4 PROPOSED FRAMEWORK

This section considers the proposed framework for implementing RL algorithms for web applications load balancing. We start by reviewing the three main algorithms discussed in this study: the Epsilon-Greedy (EG) algorithm, the Upper Confidence Bound (UCB) algorithm, and the Proximity Policy Optimization (PPO) algorithm. We then present our implementation approach. Finally, we present the practical application of the proposed system on Amazon Web Services (AWS), which formed the basis for our experimental evaluation. The three algorithms are detailed below within the proposed system:

Epsilon-Greedy (EG): The Epsilon-Greedy algorithm (Bulut, 2022) is one of the basic methods in the field of reinforcement learning, as it aims to achieve a balance between exploration and exploitation during the decision-making process. This algorithm is classified as model-free, value-based, and on-policy.

The idea is that an agent seeks to take advantage of high Q-value strategies that they have already tested but also tries new possibilities that may be more effective. Only exploitation can lead to local solutions without reaching the best. Exploration adds randomness, giving the agent a chance to discover alternative strategies.

This algorithm balances these two paths by determining a specific probability ε used to choose a random action from among the available actions. If we consider that the current state s, and that the set of possible actions is $A(s)$, the agent will choose, with probability $(1-\varepsilon)$, the action with the highest Q-value in that case, which is the exploitation aspect. As for the probability ε, an action randomly selected from group $A(s)$ for exploration (Ris-Ala, 2023).

Upper Confidence Bound (UCB): The UCB algorithm (Bulut, 2022) is another way to balance exploration and exploitation, but it uses a more sophisticated approach than Epsilon-Greedy, as it is based on what is known as optimism under uncertainty. As in EG, UCB is classified as modeless, values-based algorithms and operates within the on-policy framework. The basic idea of UCB is to maintain a positive outlook on the potential returns of each action, prompting the agent to try procedures that may be better than the current estimate even if they have not yet been tested often. The UCB method creates an upper confidence bound (ucb_score) for each action's Q-value. The agent selects the action with the highest UCB, balancing between actions with high estimated values and those with high uncertainty.

Proximal Policy Optimization (PPO): PPO (Gu et al., 2022) was introduced in 2017 as a deep reinforcement learning algorithm that directly optimises the agent's policy. It is particularly well-suited for tasks involving continuous action spaces and is recognised for its focus on stable learning and efficient use of interaction data. As a model-free method, PPO learns through direct experience without needing an internal model of the environment. It operates as a policy-based algorithm, meaning it explicitly learns and refines the probability distribution defining which actions to take in given states, aiming to maximise cumulative rewards.

PPO builds upon the foundation of policy gradient methods, which generally adjust the policy to increase the likelihood of actions leading to better-than-expected outcomes (positive advantage) and decrease the likelihood of actions leading to worse-than-expected results. PPO's key innovation is the introduction of a "*clipped*" objective function. This mechanism carefully limits the extent to which the policy can change during each update step by comparing the action probabilities under the new policy to those under the previous one (Brahmam & Vijay Anand, 2024).

This clipping is a critical safeguard, preventing very large updates that could destabilize learning or lead to performance collapses. By restricting policy changes, the PPO algorithm promotes more reliable and stable learning progress than some previous policy gradient techniques (C. Wu & Yang, 2024).

https://doi.org/10.18489/sacj.v37i2.20753

4.1 RL algorithms implementation and enhancements

The methodology of implementing RL for load balancing is presented in this section. In this context, an intelligent agent learns to make optimal routing decisions allowing it to dynamically adapt to changing traffic patterns and server conditions. The following components characterise the RL framework:

States: indicate the current status of the system. States include information about each backend server, including utilisation of CPU, response time, and the number of active connections. They also include system-wide metrics such as average response time and overall resource utilisation.

Actions: selecting which backend server to route the incoming web request to.

Rewards: Rewards provide feedback to the agent, indicating the effectiveness of its actions. Our reward equation encourages fast response times and efficient resource utilisation across all servers. The reward is calculated in Equation (1):

$$reward = 1.0 + \left(\frac{1}{Res}\right) - CPU \times 0.02 \tag{1}$$

where:

Res presents the response time from backend servers in milliseconds.

CPU presents the percentage of backend server usage. The weight of 0.02 was chosen empirically to balance the influence of CPU utilisation relative to response time, ensuring that the agent prioritises both factors effectively.

Base reward: The equation starts with a base reward of 1.0. This ensures the agent receives a positive reward even for moderately performing actions. A positive baseline is crucial in RL to encourage exploration and learning and prevent the agent from getting stuck in a cycle of negative rewards.

Response time component: Response time is a critical metric for user experience, and the goal is to minimise it. The inverse relationship ensures that the reward increases as the response time decreases.

The reward Equation (1) is designed to guide the RL agent toward optimal load-balancing decisions by incentivizing low response times and balanced resource utilization.

4.1.1 Adaptive Exploration Strategy

A constant exploration rate introduces a fundamental inefficiency that harms performance in a dynamic load-balancing environment. In the initial learning phase, a low, fixed exploration rate would slow down the discovery of optimal server routing paths. Conversely, in the later

https://doi.org/10.18489/sacj.v37i2.20753

stages, after the agent has identified effective policies, a non-zero fixed rate would force the system to continue making random, suboptimal decisions. This would needlessly increase latency and reduce overall throughput by persistently routing some requests to servers already known to be performing poorly.

To address this, we introduce an adaptive exploration strategy. The core motivation is to enable the agent to be highly inquisitive when its knowledge is limited (at the beginning) and become progressively more exploitative as it gains confidence in its decisions. This approach allows the agent to learn optimal routes quickly and then converge on a stable, high-performance policy, thereby maximizing system efficiency by fully leveraging its learned insights. This strategy dynamically adjusts the exploration rate ε using Equation (2):

$$\varepsilon = 0.01 + \sqrt{\frac{2 \times \log\left(\textit{total\_count}\right)}{n_a + 1}} \tag{2}$$

Where *total_count* represents the total number of iterations, and n_a represents the number of times the specific action a has been taken. Equation (2) is designed to dynamically balance the exploration-exploitation trade-off in reinforcement learning. The exploration rate ε is a crucial parameter that controls the extent to which the agent explores new actions versus exploiting known, high-reward actions. Algorithms 1 and 2 represent the pseudo code for implementing Enhanced EG and Enhanced UCB.

4.2 EEG and EUCB Algorithm Implementation

To translate the concepts of adaptive exploration and reward calculation into a practical implementation, we designed two algorithms: Algorithm 1 for the Enhanced Epsilon-Greedy (EEG) approach and Algorithm 2 for the Enhanced Upper Confidence Bound (EUCB) approach. These algorithms detail the step-by-step logic the load balancer uses for each incoming request.

As shown in Algorithm 1, the EEG process begins by initializing key parameters. For each incoming request, the agent decides whether to explore or exploit based on the current adaptive exploration rate, ε (lines 8-14). If exploring, it chooses a server randomly; if exploiting, it selects the server with the highest Q-value for the current state. After the request is routed (line 15), the agent observes the new system state and calculates a reward using Equation (1) (lines 16-17). This state-action-reward information is then used to update the Q-table, and the exploration rate ε is updated for the next iteration using Equation (2) (lines 18-19).

Similarly, Algorithm 2 outlines the EUCB strategy. For each request, the agent calculates a UCB score for all available servers (line 8), which balances known performance with uncertainty. It then selects the server with the maximum UCB score (lines 8-12). The subsequent process of observing the new state, calculating the reward (Equation (1)), updating the Q-table, and updating the exploration parameter (using the logic from Equation (2)) follows (lines 14-17).

https://doi.org/10.18489/sacj.v37i2.20753

Algorithm 1 Load balancing based on EEG

Input: Incoming request stream
Data: List of web server IPs, Q-table
Initialize $\varepsilon \leftarrow 0.2$
Initialize Reward $\leftarrow 1$
while system is running **do**
    Wait for New Request
    Generate random number $R \sim \mathcal{U}(0, 1)$
    if $R < \varepsilon$ **then**
        /* **Exploration** */
        Select server randomly
    else
        /* **Exploitation** */
        Select server s with maximum $Q(\text{state}, s)$
    end if
    Action: send Request to server s
    state = get current system state
    Reward $\leftarrow$ Equation 1
    Update Q-table (state, server, Reward)
    Update $\varepsilon \leftarrow$ Equation 2
end while

Algorithm 2 Load balancing based on EUCB

Input: Incoming request stream
Data: List of web server IPs, Q-table
Initialize $c \leftarrow 0.2$
Initialize Reward $\leftarrow 1$
Initialize MAX_UCB_score $\leftarrow 0$
while system is running **do**
    Wait for New Request
    Calculate UCB_score
    if UCB_score $>$ MAX_UCB_score **then**
        MAX_UCB_score $\leftarrow$ UCB_score
    end if
    Select server s based on MAX_UCB_score
    Action: send Request to server s
    state = get current system state
    Reward $\leftarrow$ Equation 1
    Update Q-table (state, server, Reward)
    Update $c \leftarrow$ Equation 2
end while

4.3 Proximal Policy Optimisation Implementation

The PPO agent consists of a four-layered perceptron (MLP) in the implementation. This MLP takes the system's state as input, including metrics such as CPU utilisation, response time, the number of active connections for each backend server, and overall system averages. Its hidden layer has two layers of ReLU-activating functions, which allow learning non-trivial patterns while keeping the structure simple. Its output layer provides a probability distribution over possible actions, guiding the selection of the most appropriate server for each incoming request.

The same dynamic reward equation used in previous algorithms guides the learning of the PPO agent. Reward Equation (1), on the other hand, encourages fast response times and efficient resource usage on all servers. The PPO agent interacts with the Nginx environment by choosing an action – routing requests – and receives a reward based on the system's performance. Figure 2 presents the PPO load balancer flowchart.

As shown in Figure 2, the process begins with the initialization of the neural network that represents the agent's policy. The agent first observes the initial state of the backend servers. Upon the arrival of a new request, the agent selects an action (i.e., chooses a server) based on this policy. After the request is dispatched, the agent observes the new system state

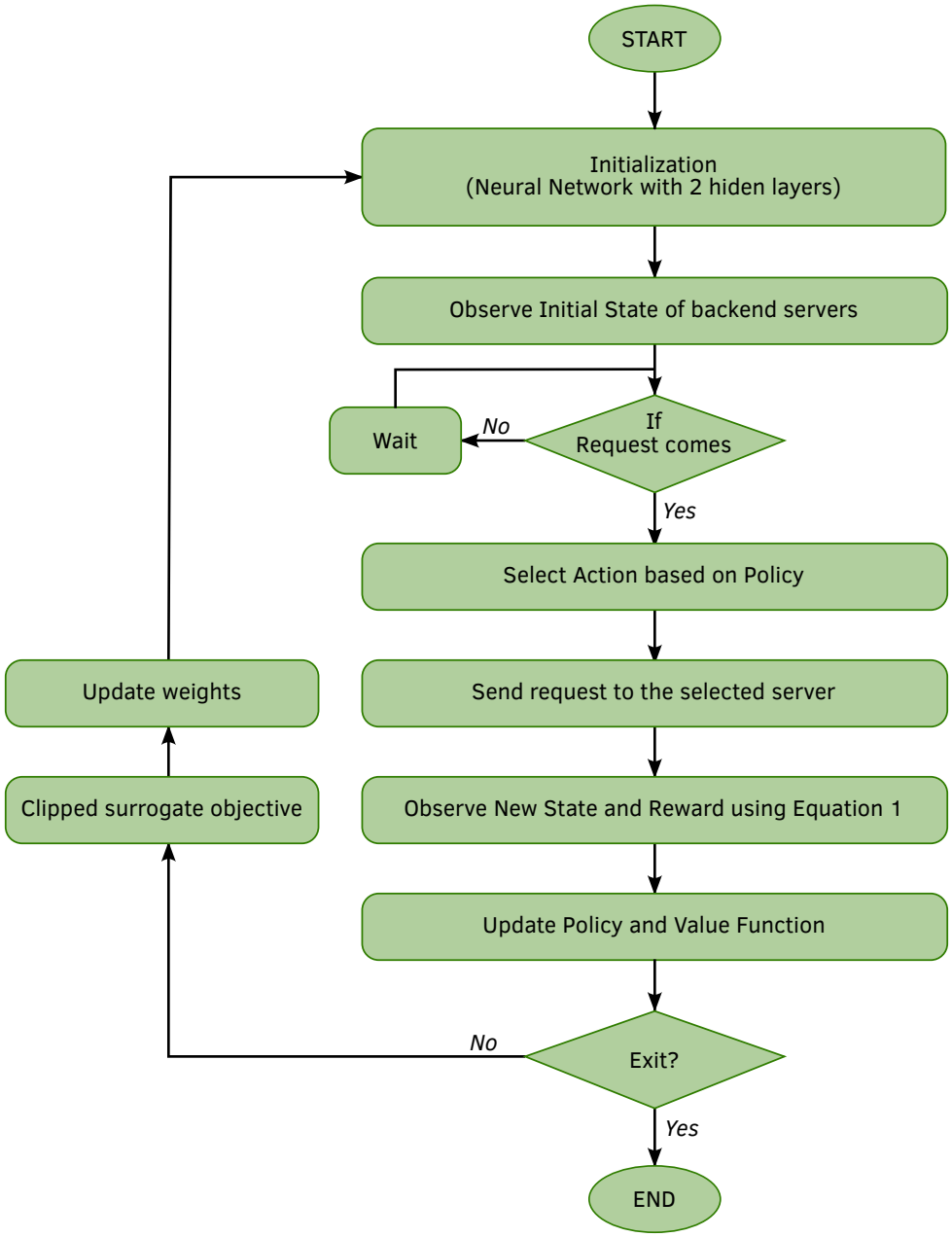


Figure 2: Flowchart of the implementation of the PPO Algorithm

and receives a reward calculated using Equation (1). These calculation will affect the policy which in turn updated by "*clipped surrogate objective*", a key feature of PPO that ensures stable learning. This iterative process of observation, action, and policy refinement allows the agent to continuously improve its load-balancing strategy over time.

https://doi.org/10.18489/sacj.v37i2.20753

4.4 The Proposed System Implementation

The experimental environment was developed to apply different scenarios of flow patterns to examine the ability of developed algorithms. Amazon Web Services (AWS) was used as a cloud platform for hosting web application instances in addition to the load balancer itself. The overall architecture of our system is depicted in Figure 3.

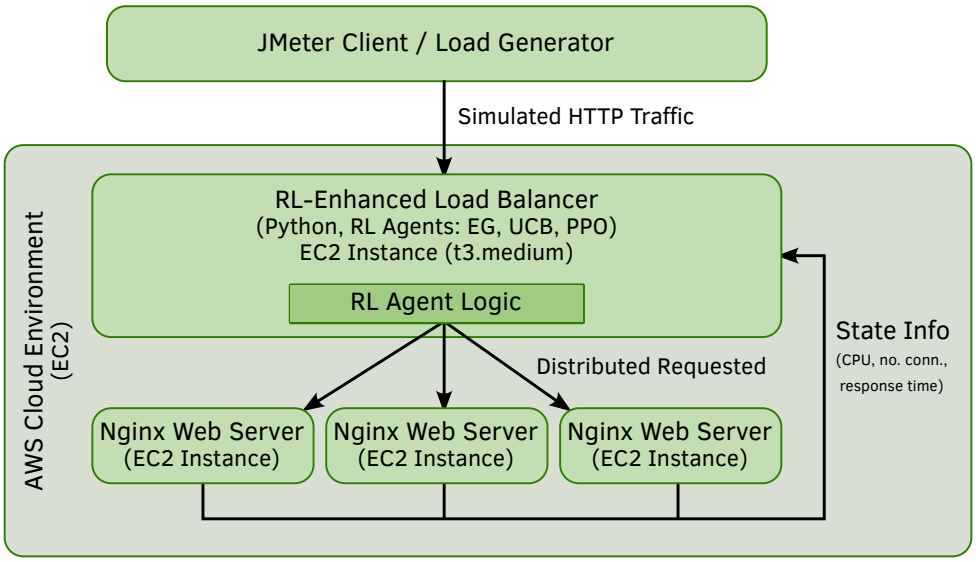


Figure 3: Architectural Overview of the Reinforcement Learning-based Load Balancing

The experimental environment consists of the following components:

Load Generation: By using Apache JMeter tool, the different patterns of HTTP traffic are generated in the local machine.

Request Handling: On AWS, the load balance developed algorithm is hosted on a t3.medium EC2 instance.

Decision and Distribution: The RL agent makes a decision as to where to forward the incoming request to depending on its strategy. The request is then distributed to the selected server.

Backend Processing: On the backend of the system, there are three Nginx web servers to handle the incoming requests, each running on a separate EC2 instance.

Feedback Loop: Each web server periodically sends its state to the load balancer. The state consists of information like CPU utilization, number of active connections, and response time as "*state-info*". This information is essential to enable the RL agent to calculate the reward, which affects its future decisions.

https://doi.org/10.18489/sacj.v37i2.20753

To comprehensively evaluate the load balancer performance across different scenarios, we generate traffic from a local machine using JMeter, which simulates different user behaviours and request patterns.

This enables us to evaluate the performance of different RL algorithms for a realistic web application environment. Analysing metrics such as response time, throughput, and latency will allow us to compare the power of our RL-based approach to traditional load-balancing methods. We used JMeter to simulate four load scenarios:

1. **Normal Load:** Moderate traffic (50 concurrent users, 1000 total requests).
2. **Burst Load:** Sudden traffic surge (100 users, 6000 requests per minute).
3. **Server Failure:** Normal load with one server failing during the test.
4. **Heterogeneous Instances:** Normal load with one server having reduced resources (t3.tiny).

5 RESULT AND DISCUSSIONS

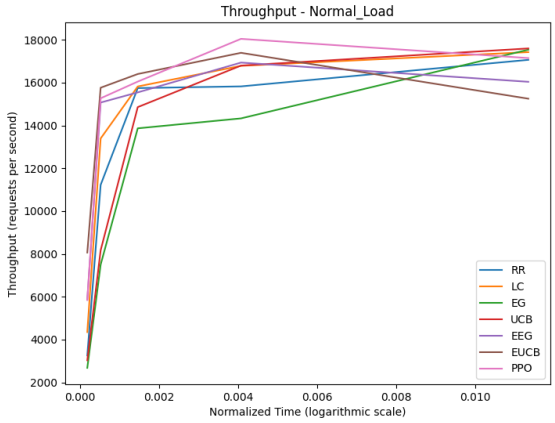
We evaluated classic load balancing algorithms (RR, LC) and enhanced RL algorithms (EEG, EUCB, PPO) in a simulated Nginx environment on AWS. Performance was measured across four scenarios (average load, burst load, server failure, and heterogeneous instances) using metrics such as throughput, latency, successful response rate, efficiency, and QoS. Results are presented as mean values with 95% confidence intervals from 10 runs per scenario.

5.1 System throughput

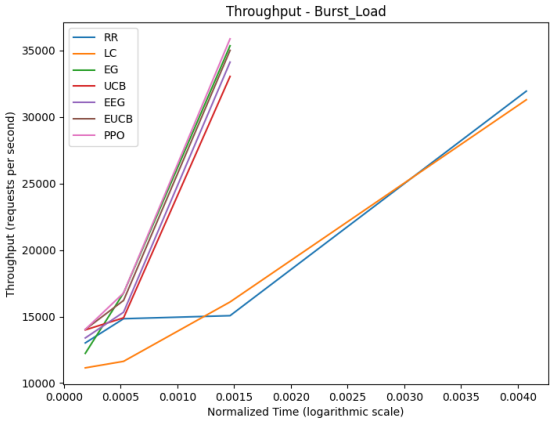
Figures 4(a) to 4(d) illustrate the throughput of the different load-balancing algorithms across the four experimental scenarios. Under normal load conditions (Figure 4(a)), all algorithms exhibited comparable performance. In the burst load scenario in Figure 4(b), characterised by a sudden surge in traffic, the RL algorithms (EEG, EUCB, and PPO) maintained significantly higher throughput than the classic RR and LC algorithms. This indicates their ability to adapt to dynamic load changes and efficiently distribute requests across the available servers. Furthermore, the RL algorithms exhibited greater resilience to server failures, as shown in Figure 4(c). When one server intentionally failed during the test, the RL algorithms quickly adjusted their routing strategies to maintain high throughput, while the performance of RR and LC noticeably degraded.

Similarly, in the heterogeneous instances scenario Figure 4(d), where servers had varying resource capacities, the RL algorithms effectively utilised the available resources and achieved higher throughput than the classic methods. Across all scenarios, PPO consistently demonstrated either the highest or near-highest throughput, suggesting its effectiveness in learning and adapting to diverse load conditions.

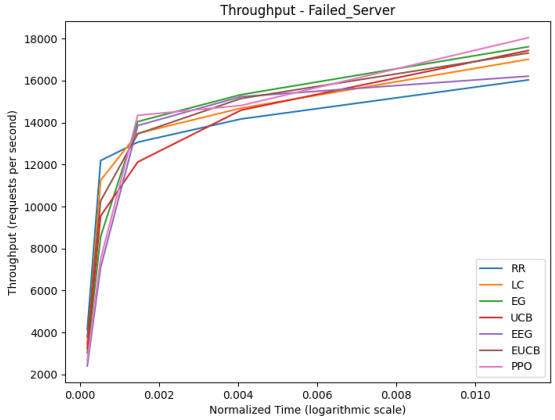
https://doi.org/10.18489/sacj.v37i2.20753



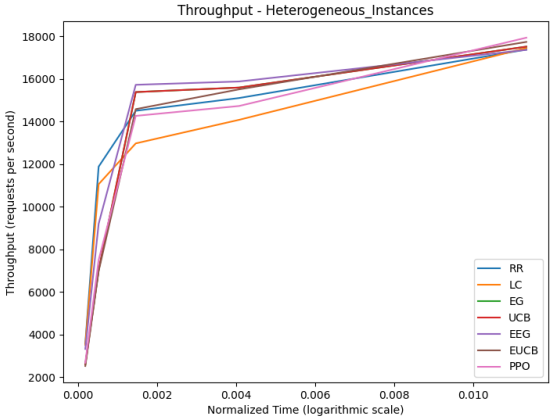
(a) The system throughput under normal load



(b) System throughput under burst load



(c) System throughput under the failed scenario



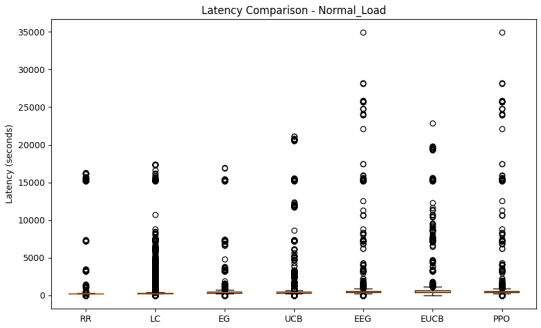
(d) System throughput under the heterogeneous instances scenario

Figure 4: System throughput under various load scenarios

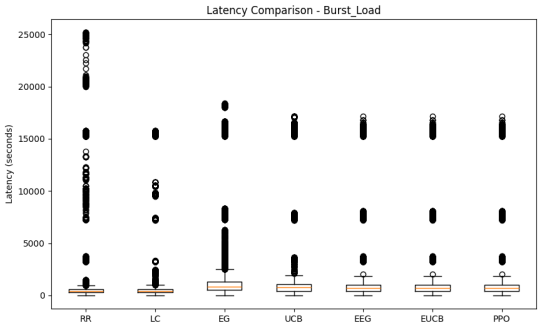
5.2 System Latency

Figures 5(a) to 5(d) illustrate the latency performance of the load-balancing algorithms across the four scenarios, presented as box plots. The box plots provide a detailed view of the latency distributions, including the median (represented by the line within each box), the interquartile range (IQR, represented by the box's height), and potential outliers (indicated by individual circles). Overall, the RL algorithms (EEG, EUCB, and PPO) demonstrate notably lower median latency and a reduced interquartile range (IQR), indicating more consistent and predictable performance, particularly under the more challenging conditions of burst loads and server failures, compared to the classic RR and LC methods. Specifically, PPO consistently exhibits competitive or superior latency performance across all scenarios, with a lower median and

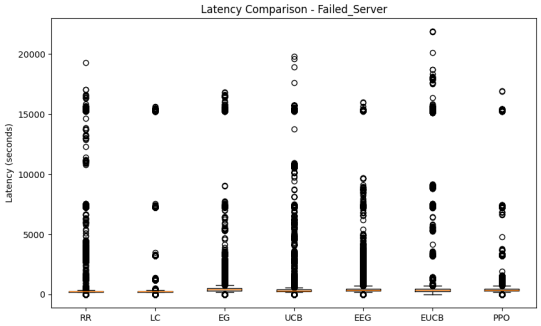
a tighter interquartile range (IQR), suggesting that it maintains responsiveness and stability under varying load conditions. Notably, the classic algorithms, RR and LC, exhibit a greater number of outliers, especially in burst-load and failed-server scenarios, indicating less stable and more unpredictable latency.



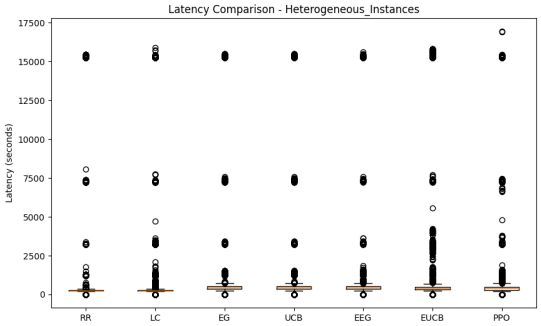
(a) System latency under normal load



(b) System latency under burst load



(c) System latency under the failed scenario



(d) System latency under the heterogeneous instances scenario

Figure 5: System latency under various load scenarios

While the RL algorithms generally outperform the classic methods, all algorithms exhibit relatively similar latency distributions in the heterogeneous instances scenario, with comparable median values and interquartile ranges (IQR), suggesting that differences in server capacity have a less pronounced impact on latency variability compared to load intensity or server availability disruptions. However, even in this scenario, the RL algorithms maintain a lower number of outliers. In summary, the box plots highlight the superior latency performance and stability of the RL algorithms, particularly PPO, with lower medians, tighter interquartile ranges (IQRs), and fewer outliers compared to the classic methods.

https://doi.org/10.18489/sacj.v37i2.20753

5.3 System Success Rate

Table 1 shows effective message success rates of different load-balancing algorithms under various scenarios. The state-of-the-art RL algorithms (EEG, EUCB, PPO) consistently show higher message success rates than the classic methods (RR, LC), especially during burst loads and when server failures occur. PPO typically achieves the best results across all tested scenarios, confirming its robustness and adaptability in maintaining communication reliability under dynamic conditions. However, under common load conditions and heterogeneous server settings, the differences between the algorithms diminish, indicating that these conditions pose a reduced challenge to effective message delivery.

Table 1: The system's successful request rate

	Algorithms (%)						
Scenarios	**RR**	**LC**	**EG**	**UCB**	**EEG**	**EUCB**	**PPO**
Normal Load	88.5	92.0	91.1	92.2	91.6	91.9	97.1
Burst Load	86.9	86.2	93.0	87.6	90.3	92.5	94.7
Failed Server	84.4	88.5	92.1	91.3	83.9	91.8	93.6
Heterogeneous Instances	90.4	90.9	91.0	91.0	91.8	92.0	93.1

5.4 Quality of Service (QoS)

Figure 6 shows the Latency-throughput trade-off in various scenarios. In general, recent reinforcement learning-based algorithms, such as EEG, EUCB, and PPO, achieve lower latency at the cost of higher throughput compared to classical approaches, RR and LC, under burst loads and server failures. PPO consistently demonstrates a good balance between low latency and high throughput in all scenarios, showcasing its effectiveness in enhancing Quality of Service (QoS). However, considering heterogeneous server settings, the discrimination between algorithms appears to decrease, suggesting that variability across servers may have a relatively less significant impact on the aggregate quality of service equilibrium.

5.5 The System Efficiency

Figure 7 depicts the efficiency of the different load-balancing algorithms under varying scenarios. The enhanced RL algorithms – EEG, EUCB, and PPO – show higher efficiency than classical methods, namely RR and LC, especially for burst loads. This suggests that the RL-based load balancer has great potential in optimising resource utilisation and dealing with dynamic traffic conditions.

https://doi.org/10.18489/sacj.v37i2.20753

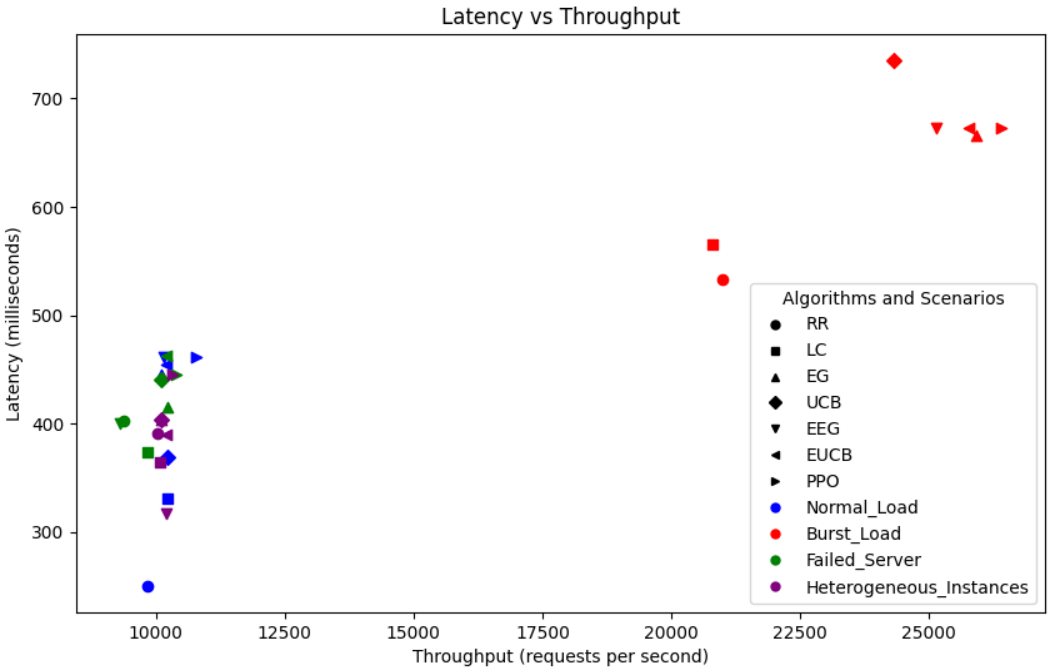


Figure 6: The System QoS

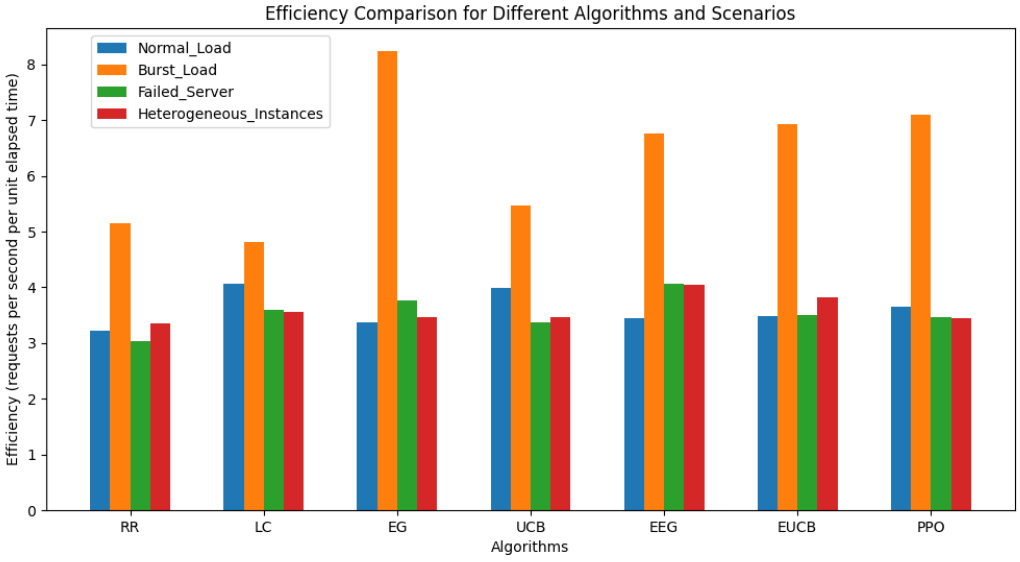


Figure 7: Efficiency of the system

https://doi.org/10.18489/sacj.v37i2.20753

6 CONCLUSION

This research examines the effectiveness of reinforcement learning (RL) algorithms in improving load balance within web applications. A simulator was created on the AWS platform to evaluate the performance of three state-of-the-art algorithms in the industry: Enhanced Epsilon-Greedy, Upper Confidence Bound, and Proximal Policy Optimisation, compared with two traditional methods used to balance loads, Round Robin, and Least Connections. Test scenarios included normal loads, burst loads, in addition to failures of one server, and heterogeneous server instances.

The results showed that RL algorithms, especially PPO, clearly outperformed traditional methods in terms of key performance metrics. PPO increased throughput by up to 30%, reduced latency by 20%, and improved successful messaging rate by 5% compared to the best traditional methods. These gains are clear in situations of high system pressure or when one of the failures occurs, indicating the ability of these algorithms to adapt and be stable.

These findings also open the door to promising future research prospects. In future research, more sophisticated learning algorithms, such as Deep Deterministic Policy Gradient (DDPG) or Twin Delayed DDPG (TD3), could be explored that are designed to operate within continuous decision-making spaces, potentially allowing more precise load control over load distribution.

Reinforcement agent performance can also be enhanced if additional variables are included in the system state representation, such as load change rate, overall server performance trend, or network conditions. By entering this dynamic data, the agent will be able to predict potential changes and proactively adjust the distribution strategy, enhancing performance efficiency and reducing waste.

References

Abouamasha, S. R., Aboelwafa, M., & Seddik, K. G. (2025). Load balancing and energy efficiency in cellular networks with a scenario-aware reinforcement learning agent. *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, 1–6. https://doi.org/10.1109/WCNC61545.2025.10978457

Adarsh, M. G., & Bhargavi, K. (2023). Reinforcement learning-based Bonobo optimizer for efficient load balancing in cloud computing. *2023 3rd International Conference on Intelligent Technologies (CONIT)*, 1–5. https://doi.org/10.1109/CONIT59222.2023.10205866

Al Shaikh, R. Z., Al-Nayar, M. M. J., & Hasan, A. M. (2025). Reinforcement learning algorithms for adaptive load balancing in publish/subscribe systems: PPO, UCB, and epsilon-greedy approaches. *Informatica*, *49*, 77–88. https://doi.org/10.31449/INF.V49I7.6895

Attia, B. S., Elgharably, A., Mariam, A. N., Alsuhli, G., Banawan, K., & Seddik, K. G. (2024). Self-optimized agent for load balancing and energy efficiency: A reinforcement learning framework with hybrid action space. *IEEE Open Journal of the Communications Society*, *5*, 4902–4919. https://doi.org/10.1109/OJCOMS.2024.3429284

Auer, P. (2000). Using upper confidence bounds for online learning. *Proceedings 41st Annual Symposium on Foundations of Computer Science*, 270–279. https://doi.org/10.1109/SFCS.2000.892116

Badini, N., Jaber, M., Marchese, M., & Patrone, F. (2023). Reinforcement learning-based load balancing satellite handover using NS-3. *ICC 2023 - IEEE International Conference on Communications*, 2595–2600. https://doi.org/10.1109/ICC45041.2023.10279521

Baek, J., & Kaddoum, G. (2023). FLoadNet: Load balancing in fog networks with cooperative multiagent using actor–critic method. *IEEE Transactions on Network and Service Management*, *20*(1), 400–414. https://doi.org/10.1109/TNSM.2022.3210827

Barlas, G. (2023). Load balancing. In *Multicore and gpu programming* (2nd ed., pp. 887–941). Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-814120-5.00022-6

Brahmam, M. G., & Vijay Anand, R. (2024). VMMISD: An efficient load balancing model for virtual machine migrations via fused metaheuristics with iterative security measures and deep learning optimizations. *IEEE Access*, *12*, 39351–39374. https://doi.org/10.1109/ACCESS.2024.3373465

Bulut, V. (2022). Optimal path planning method based on epsilon-greedy Q-learning algorithm. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, *44*(106). https://doi.org/10.1007/s40430-022-03399-w

Chang, H.-H., Chen, H., Zhang, J., & Liu, L. (2023). Decentralized deep reinforcement learning meets mobility load balancing. *IEEE/ACM Transactions on Networking*, *31*(2), 473–484. https://doi.org/10.1109/TNET.2022.3176528

Chen, Q., Song, X., Song, T., & Yang, Y. (2025). Vehicular edge computing networks optimization via DRL-based communication resource allocation and load balancing. *IEEE Transactions on Mobile Computing*, *24*(9), 9222–9237. https://doi.org/10.1109/TMC.2025.3559707

Choppara, P., & Lokesh, B. (2025). Efficient task scheduling and load balancing in fog computing for crucial healthcare through deep reinforcement learning. *IEEE Access*, *13*, 26542–26563. https://doi.org/10.1109/ACCESS.2025.3539336

Coelho, B. L., & Schaeffer-Filho, A. E. (2023). CrossBal: Data and control plane cooperation for efficient and scalable network load balancing. *2023 19th International Conference on Network and Service Management (CNSM)*, 1–9. https://doi.org/10.23919/CNSM59352.2023.10327790

Das, S., White, A., Lyn, M., & Kalafatis, S. (2025). Smartly managing traffic and latency in a content-based data center using modified packet headers, machine learning, load balancing, and network telemetry. *2025 International Conference on Computing, Networking and Communications (ICNC)*, 162–167. https://doi.org/10.1109/ICNC64010.2025.10994157

Dinh, L., Quang, P. T. A., & Leguay, J. (2024). Towards safe load balancing based on control barrier functions and deep reinforcement learning. *NOMS 2024 – IEEE Network Operations and Management Symposium*, 1–9. https://doi.org/10.1109/NOMS59830.2024.10575399

Fawaz, H., Houidi, O., Zeghlache, D., Lesca, J., Quang, P. T. A., Leguay, J., & Medagliani, P. (2023). Graph convolutional reinforcement learning for load balancing and smart queuing. *2023 IFIP Networking Conference (IFIP Networking)*, 1–9. https://doi.org/10.23919/IFIPNetworking57963.2023.10186430

Feng, L., Xie, R., Tang, Q., & Huang, T. (2024). Delay-prioritized task scheduling with load balancing in computing power networks. *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, 1–6. https://doi.org/10.1109/WCNC57260.2024.10570622

Ghomi, E. J., Rahmani, A. M., & Qader, N. N. (2017). Load-balancing algorithms in cloud computing: A survey. *Journal of Network and Computer Applications*, *88*, 50–71. https://doi.org/10.1016/J.JNCA.2017.04.007

Ghosh, S., Seshadri, K., & Kollengode, C. (2024). Design and evaluation of reward functions for load balancing in cloud datacenters. *2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT)*, 1–6. https://doi.org/10.1109/ICITIIT61487.2024.10580611

Gu, Y., Cheng, Y., Chen, C. L. P., & Wang, X. (2022). Proximal policy optimization with policy feedback. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *52*(7), 4600–4610. https://doi.org/10.1109/TSMC.2021.3098451

Gupta, M., Chinchali, S., Varkey, P. P., & Andrews, J. G. (2024). Forecaster-aided user association and load balancing in multi-band mobile networks. *IEEE Transactions on Wireless Communications*, *23*(5), 5157–5171. https://doi.org/10.1109/TWC.2023.3324685

Han, S., Lee, E., & Pack, S. (2023). Fine-grained load balancing with multi-agent reinforcement learning for self-organizing networks. *2023 IEEE Globecom Workshops (GC Wkshps)*, 578–583. https://doi.org/10.1109/GCWkshps58843.2023.10464981

Hartman, L., Gomez-Rosero, S., Adjei, P., & Capretz, M. A. M. (2023). Multi-factor edge-weighting with reinforcement learning for load balancing of electric vehicle charging stations. *2023 International Conference on Machine Learning and Applications (ICMLA)*, 580–587. https://doi.org/10.1109/ICMLA58977.2023.00086

He, C., Jiang, W., Wang, X., Wang, W., & Xie, X. (2025). Generative AI-enhanced task offloading strategy for the IoV: An RSU-RSU load-balancing perspective. *IEEE Networking Letters*, *7*(3), 161–165. https://doi.org/10.1109/LNET.2025.3542094

Hong, Y., No, J., & Kim, S. (2006). DNS-based load balancing in distributed web-server systems. *The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06)*. https://doi.org/10.1109/SEUS-WCCIA.2006.23

Houidi, O., Bakri, S., Zeghlache, D., Lesca, J., Quang, P. T. A., Leguay, J., & Medagliani, P. (2023). AMAC: Attention-based multi-agent cooperation for smart load balancing. *NOMS 2023 – IEEE/IFIP Network Operations and Management Symposium*, 1–7. https://doi.org/10.1109/NOMS56928.2023.10154214

Huang, C., Wang, F., & Xu, W. (2024). Joint UAV movement control and load balancing based on indirect control in air-ground-integrated networks. *IEEE Wireless Communications Letters*, *13*(6), 1616–1620. https://doi.org/10.1109/LWC.2024.3383858

Jeong, Y., Lim, J., Choi, G., & Roh, B. (2024). Deep deterministic policy gradient-based load balancing method in SDN environments. *2024 International Conference on Smart Applications, Communications and Networking (SmartNets)*, 1–6. https://doi.org/10.1109/SmartNets61466.2024.10577651

Jiang, D., Zhu, B., Liu, X., & Mumtaz, S. (2024). ALCoD: An adaptive load-aware approach to load balancing for containers in IoT edge computing. *IEEE Internet of Things Journal*, *11*(23), 37480–37492. https://doi.org/10.1109/JIOT.2024.3427642

Jiang, G., Wei, W., Wang, K., Pang, C., & Liu, Y. (2023). DRL-TAL: Deep reinforcement learning-based traffic-aware load balancing in data center networks. *GLOBECOM 2023 – IEEE Global Communications Conference*, 928–933. https://doi.org/10.1109/GLOBECOM54140.2023.10437481

Johann, T., Kuehl, S., & Pachnicke, S. (2024). Deep reinforcement learning based decentralized routing and load-balancing in meshed QKD-networks. *ECOC 2024; 50th European Conference on Optical Communication*, 1385–1388. https://ieeexplore.ieee.org/document/10926232

Jurše, J., & Kuhar, U. (2024). Enhancing low voltage network capacity through phase load balance optimization using advanced smart meter data analytics and static reconnections. *CIRED 2024 Vienna Workshop*, *2024*, 409–412. https://doi.org/10.1049/icp.2024.2062

Khoramnejad, F., & Hossain, E. (2025). Carrier aggregation, load balancing, and backhauling in non-terrestrial networks: Generative diffusion model-based optimization. *IEEE Transactions on Wireless Communications*, *24*(5), 4483–4499. https://doi.org/10.1109/TWC.2025.3542126

Kołakowski, R., Kukliński, S., & Tomaszewski, L. (2024). Hierarchical deep reinforcement learning-based load balancing algorithm for multi-domain software-defined networks. *2024 IFIP Networking Conference (IFIP Networking)*, 607–612. https://doi.org/10.23919/IFIPNetworking62109.2024.10619899

Konar, A., Wu, D., Xu, Y. T., Jang, S., Liu, S., & Dudek, G. (2023). Communication load balancing via efficient inverse reinforcement learning. *ICC 2023 – IEEE International Conference on Communications*, 472–478. https://doi.org/10.1109/ICC45041.2023.10279136

Kopparapu, C. (2002). *Load balancing servers, firewalls, and caches*. Wiley.

Krishna, M. S. R., & Khasim Vali, D. (2025). Meta-RHDC: Meta reinforcement learning driven hybrid Lyrebird Falcon optimization for dynamic load balancing in cloud computing. *IEEE Access*, *13*, 36550–36574. https://doi.org/10.1109/ACCESS.2025.3544775

Kumari, A., Roy, A., & Singh Sairam, A. (2024). Optimizing SDN controller load balancing using online reinforcement learning. *IEEE Access*, *12*, 131591–131604. https://doi.org/10.1109/ACCESS.2024.3459952

Lahande, P. V., Kaveri, P. R., Saini, J. R., Kotecha, K., & Alfarhood, S. (2023). Reinforcement learning approach for optimizing cloud resource utilization with load balancing. *IEEE Access*, *11*, 127567–127577. https://doi.org/10.1109/ACCESS.2023.3329557

Li, T., Ying, S., Zhao, Y., & Shang, J. (2024). Batch jobs load balancing scheduling in cloud computing using distributional reinforcement learning. *IEEE Transactions on Parallel and Distributed Systems*, *35*(1), 169–185. https://doi.org/10.1109/TPDS.2023.3334519

Liu, J., Guo, Y., Leng, X., & Tang, X. (2025). Personalized federated management and load balancing for multiple charging stations. *IEEE Transactions on Industrial Informatics*, *21*(8), 6262–6273. https://doi.org/10.1109/TII.2025.3563534

Liu, J., Liu, Z., & Tang, X. (2023). A deep reinforcement learning method for charging station management and load balancing. *2023 IEEE Transportation Electrification Conference and Expo, Asia-Pacific (ITEC Asia-Pacific)*, 1–5. https://doi.org/10.1109/ITECAsia-Pacific59272.2023.10372308

Liu, L., Chen, Y., Shao, X., Wang, K., & Huang, Y. (2024). Adaptive load balancing strategy for VR microservices with edge AI. *2024 7th International Conference on Electronics Technology (ICET)*, 1064–1069. https://doi.org/10.1109/ICET61945.2024.10672636

Liu, Q., Li, X., Ji, H., & Zhang, H. (2023). User grouping-based beam handover scheme with load-balancing for LEO satellite networks. *GLOBECOM 2023 – IEEE Global Communications Conference*, 3965–3970. https://doi.org/10.1109/GLOBECOM54140.2023.10436950

Ma, J., Liua, X., Hu, H. X. D., Shi, G., & Ma, L. (2025). HydraChain: A cooperative MAPPO architecture for load balancing in IoT sharding blockchain. *IEEE Internet of Things Journal*, *12*(15). https://doi.org/10.1109/JIOT.2025.3567146

Mary, A. H., Miry, A. H., & Miry, M. H. (2022). ANFIS based reinforcement learning strategy for control a nonlinear coupled tanks system. *Journal of Electrical Engineering and Technology*, *17*(3), 1921–1929. https://doi.org/10.1007/S42835-021-00753-1/METRICS

Mosalli, H., Sanami, S., Yang, Y., Yeh, H., & Aghdam, A. G. (2025). Dynamic load balancing for EV charging stations using reinforcement learning and demand prediction. *2025 IEEE International systems Conference (SysCon)*, 1–7. https://doi.org/10.1109/SysCon64521.2025.11014850

Nakagawa, S., & Mizutani, K. (2023). An implementation of intelligent image analysis load balancing for virtualized environment. *2023 Fourteenth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, 1–2. https://doi.org/10.23919/ICMU58504.2023.10412162

Naman, H. A., & Ameen, Z. J. M. (2022). A new method in feature selection based on deep reinforcement learning in domain adaptation. *Iraqi Journal of Science*, *63*(2), 817–829. https://doi.org/10.24996/IJS.2022.63.2.35

Nimmala, S., Ramchander, M., Mahendar, M., Manasa, P., Bhavani, D. D., & Raghavendar, K. (2024). Dynamic RL-ACO: Reinforcement learning-based ant colony optimization for load balancing in cloud networks. *2024 5th International Conference on Smart Electronics*

and Communication (ICOSEC), 475–480. https://doi.org/10.1109/ICOSEC61587.2024.10722410

Ris-Ala, R. (2023). *Fundamentals of reinforcement learning*. Springer Nature. https://link.springer.com/book/10.1007/978-3-031-37345-9

Sanghi, N. (2021). *Deep reinforcement learning with Python: With PyTorch, TensorFlow and OpenAI Gym*. Springer Nature. https://doi.org/10.1007/978-1-4842-6809-4

Santos, J., Wauters, T., Turck, F. D., & Steenkiste, P. (2024). Towards optimal load balancing in multi-zone Kubernetes clusters via reinforcement learning. *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*, 1–9. https://doi.org/10.1109/ICCCN61486.2024.10637606

Shahakar, M., Mahajan, S. A., & Patil, L. (2024). ONU: A reinforcement load balancing approach for resource-aware task allocation in distributed systems. *2024 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 1–6. https://doi.org/10.1109/ESCI59607.2024.10497200

Shaikh, M. R. R. (2023). Bayesian network based optimal load balancing in software defined networks. *2023 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 1–5. https://doi.org/10.1109/ESCI56872.2023.10099730

Shang, X., Liu, Z., Gao, D., Yang, D., Zhang, W., Foh, C. H., & Zhang, H. (2025). Computing and network load balancing for decentralized deep federated learning in industrial cyber-physical systems: A multi-task approach. *IEEE Journal on Selected Areas in Communications*, *43*(9), 2997–3013. https://doi.org/10.1109/JSAC.2025.3574618

Shi, B., & Chen, F. (2023). A dynamic pricing strategy for load balancing across multiple edge servers. *2023 IEEE International Conference on Web Services (ICWS)*, 329–339. https://doi.org/10.1109/ICWS60048.2023.00052

Shi, Y., Yang, Q., Huang, X., Li, D., & Huang, X. (2023). An SDN-enabled framework for a load-balanced and QoS-aware internet of underwater things. *IEEE Internet of Things Journal*, *10*(9), 7824–7834. https://doi.org/10.1109/JIOT.2022.3231329

Sun, A., Yang, F., Wu, W., Wang, T., & Sun, Y. (2023). Deep reinforcement learning-based load balancing algorithm for sliced ultra-dense network. *2023 Eleventh International Conference on Advanced Cloud and Big Data (CBD)*, 27–32. https://doi.org/10.1109/CBD63341.2023.00014

Sun, H., Zhang, H., Ma, H., & Leung, V. C. M. (2025). Joint scheduling, computing, and load balancing for time sensitive traffic in SDN-enabled space-air-ground integrated 6G networks: A federated reinforcement learning approach. *IEEE Transactions on Mobile Computing*, *24*(10), 9995–10008. https://doi.org/10.1109/TMC.2025.3567289

Takahashi, H., & Shinomiya, N. (2024). An approximate solution using K-shortest path and reinforcement learning for a load balancing problem in communication networks. *2024 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*, 659–660. https://doi.org/10.1109/ICCE-Taiwan62264.2024.10674659

Talpur, A., & Gurusamy, M. (2023). Optimizing vehicle-to-edge mapping with load balancing for attack-resilience in IoV. *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, 341–347. https://doi.org/10.1109/CCNC51644.2023.10060632

Tian, B., Pan, X., Wang, F., Hu, S., Zhu, L., Xin, X., & Yu, J. (2025). O-RAN-enabled collaborative multi-access edge computing over optical metro and access networks for enhanced load balancing. *Journal of Lightwave Technology*, *43*(14), 6515–6532. https://doi.org/10.1109/JLT.2025.3566334

Tian, S., Xiang, S., Zhou, Z., Dai, H., Yu, E., & Deng, Q. (2025). Task offloading and resource allocation based on reinforcement learning and load balancing in vehicular networking. *IEEE Transactions on Consumer Electronics*, *71*(1), 2217–2230. https://doi.org/10.1109/TCE.2025.3542133

Tian, Y., Li, J., Wu, D., Jenkin, M., Jang, S., Liu, X., & Dudek, G. (2023). Policy reuse for communication load balancing in unseen traffic scenarios. *ICC 2023 - IEEE International Conference on Communications*, 309–315. https://doi.org/10.1109/ICC45041.2023.10278806

Vaishnavi, S., Maruti, M. S., & Bhargavi, K. (2023). Forward-backward inverse reinforcement learning for load balancing in SDN. *2023 International Conference on Network, Multimedia and Information Technology (NMITCON)*, 01–07. https://doi.org/10.1109/NMITCON58196.2023.10276272

Wang, G., Yang, F., Song, J., & Han, Z. (2025). Resource allocation and load balancing for beam hopping scheduling in satellite-terrestrial communications: A cooperative satellite approach. *IEEE Transactions on Wireless Communications*, *24*(2), 1339–1354. https://doi.org/10.1109/TWC.2024.3508741

Wibowo, B., Haq, A., & Zein, M. T. A. A. (2023). A comparative study of load balancing methods in cloud-based village information systems. *2023 IEEE 7th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, 285–289. https://doi.org/10.1109/ICITISEE58992.2023.10404194

Wu, C., & Yang, T. (2024). Load balancing in fog networks using digital twins. *IET International Conference on Engineering Technologies and Applications (ICETA 2024)*, *2024*, 98–99. https://doi.org/10.1049/icp.2024.4194

Wu, D., Xu, Y. T., Li, J., Jenkin, M., Hossain, E., Jang, S., Xin, Y., Zhang, C., Liu, X., & Dudek, G. (2023). Learning to adapt: Communication load balancing via adaptive deep reinforcement learning. *GLOBECOM 2023 – IEEE Global Communications Conference*, 2973–2978. https://doi.org/10.1109/GLOBECOM54140.2023.10437528

Wu, G., Wang, H., Zhang, H., Shen, Y., Shen, S., & Yu, S. (2024). Mean-field game-based task-offloaded load balance for industrial mobile edge computing systems using software-defined networking. *IEEE Transactions on Mobile Computing*, *23*(12), 13773–13786. https://doi.org/10.1109/TMC.2024.3437761

Wu, Y., Zhang, R., Huang, J., Guo, L., & Wu, J. (2025). Incentive-based two-level scheduling algorithms for load balance in vehicular edge computing. *IEEE Internet of Things Journal*, *12*(17), 35497–35509. https://doi.org/10.1109/JIOT.2025.3579039

Xie, R., Feng, L., Tang, Q., Huang, T., Xiong, Z., Chen, T., & Zhang, R. (2024). Delay-prioritized and reliable task scheduling with long-term load balancing in computing power networks. *IEEE Transactions on Services Computing*, *17*(6), 3359–3372. https://doi.org/10.1109/TSC.2024.3495500

Xu, J., Guo, H., Shen, H., Raj, M., Wurster, S. W., & Peterka, T. (2023). Reinforcement learning for load-balanced parallel particle tracing. *IEEE Transactions on Visualization and Computer Graphics*, *29*(6), 3052–3066. https://doi.org/10.1109/TVCG.2022.3148745

Xu, X., & Zou, T. (2024). Research on flow load balance scheduling strategy in SDN based on reinforcement learning. *2024 Sixth International Conference on Next Generation Data-driven Networks (NGDN)*, 409–413. https://doi.org/10.1109/NGDN61651.2024.10744167

Ye, M., Hu, Y., Zhang, J., Guo, Z., & Chao, H. J. (2023). Reinforcement learning-based traffic engineering for QoS provisioning and load balancing. *2023 IEEE/ACM 31st International Symposium on Quality of Service (IWQoS)*, 1–10. https://doi.org/10.1109/IWQoS57198.2023.10188762

Yucel, S. (2023). Reinforcement learning approach to server selection and load balancing for collaborative virtual services. *2023 Congress in Computer Science, Computer Engineering & Applied Computing (CSCE)*, 1206–1213. https://doi.org/10.1109/CSCE60160.2023.00202

Zervopoulos, A., Campos, L. M., & Oikonomou, K. (2023). Q-delegation: VNF load balancing through reinforcement learning-based packet delegation. *2023 8th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, 1–6. https://doi.org/10.1109/SEEDA-CECNSM61561.2023.10470625

Zhao, R., Cai, J., Luo, J., Gao, J., & Ran, Y. (2025). Demand-aware beam hopping and power allocation for load balancing in digital twin empowered LEO satellite networks. *IEEE Transactions on Wireless Communications*, *24*(6), 5084–5098. https://doi.org/10.1109/TWC.2025.3545745

Zhu, R., Li, G., Zhang, Y., Fang, Z., & Wang, J. (2023). Load-balanced virtual network embedding based on deep reinforcement learning for 6G regional satellite networks. *IEEE Transactions on Vehicular Technology*, *72*(11), 14631–14644. https://doi.org/10.1109/TVT.2023.3279625