Journal of Theoretical and Applied Information Technology
15th March 2024. Vol.102. No 5
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

HARNESSING REINFORCEMENT LEARNING IN FOG-CLOUD COMPUTING: CHALLENGES, INSIGHTS, AND FUTURE DIRECTIONS

MUSTAFA AL-HASHIMI[1], AMIR RIZAAN RAHIMAN[2], ABDULLAH MUHAMMED[3], NOR ASILAH WATI HAMID[4]

[1,2,3,4] Department of Communication Technology and Networks, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia

E-mail: [1]gs59883@student.upm.edu.my, [2]amir_r@upm.edu.my, [3]abdullah@upm.edu.my, [4]asila@upm.edu.my

ABSTRACT

The fast-changing world of fog-cloud computing poses various challenges and opportunities, especially in terms of optimizing resources, adaptability, and system efficiency. Reinforcement Learning (RL) is a powerful tool to tackle these challenges due to its ability to learn and adjust from interactions. This article explores the different RL algorithms, emphasizing their distinct strengths, weaknesses, and practical implications in fog-cloud environments. We present a comprehensive comparative analysis, from the deterministic nature of Q-Learning to the scalability of DQN and the adaptability of PPO, providing insights that can assist both practitioners and researchers. Additionally, we discuss the ethical considerations, real-world applicability, and scalability challenges associated with deploying RL in fog-cloud systems. In conclusion, while integrating RL in fog-cloud computing shows promise, it requires a comprehensive, interdisciplinary approach to ensure that advancements are ethical, efficient, and beneficial for everyone.

Keywords: *Fog-Cloud Computing, Q-Learning, Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO), Deep Q-Network (DQN), Reinforcement Learning.*

1. INTRODUCTION

We are living in a time where data generation has reached unprecedented levels, largely due to the widespread use of Internet of Things (IoT) devices. These devices range from smart homes and wearables to industrial sensors and urban infrastructure, and they are drastically changing the way we interact with the world [1]. The continuous flow of data from these devices requires efficient processing mechanisms to extract meaningful insights in real-time. Historically, cloud computing has been the backbone of data processing, providing centralized solutions with substantial computational power. However, as the number of IoT devices grew, it became clear that the traditional cloud-centric model had its limitations. The primary challenge was latency. Transmitting data from edge devices to distant cloud data centers introduced significant delays, making real-time processing difficult [2]. Additionally, the large volume of data generated at the edge overwhelmed network bandwidth, leading to congestion and further delays [3]. Now, fog-cloud computing has emerged as a paradigm shift that promises to revolutionize data processing in the IoT ecosystem. Fog-cloud computing decentralizes data processing and brings computational resources closer to the data source, effectively reducing latency and ensuring real-time or near-real-time processing [4]. This approach not only improves the responsiveness of IoT applications but also reduces the load on core networks, leading to more efficient bandwidth utilization [5]. However, transitioning from a centralized to a decentralized model is not without its challenges. Fog-cloud computing introduces complexities related to dynamic resource management, task scheduling, energy optimization, and fault tolerance [6]. These challenges are compounded by the inherent variability and unpredictability of fog nodes, which can range from powerful edge servers to resource-constrained devices [7].

Recently, Reinforcement Learning (RL) has emerged as a promising solution to address these challenges. RL, a type of machine learning, learns by interaction. Agents in RL environments continuously adapt their strategies based on feedback, making them particularly suited for the dynamic and ever-evolving landscape of fog-cloud computing [8]. Early studies suggest that RL-based algorithms can optimize task scheduling, reduce energy consumption, and improve fault tolerance in fog-cloud environments [9]. Despite the potential of RL in fog-cloud computing, there is a significant gap in the literature. While many studies have explored specific RL algorithms, there is a lack of comprehensive research comparing the performance, adaptability, and efficiency of various RL-based approaches in fog-cloud scenarios. This paper aims to fill this gap by providing a detailed comparative analysis that will serve as a foundation for future research and practical implementations. The significance of this research cannot be overstated. As the world becomes increasingly interconnected, with billions of IoT devices coming online, the need for efficient, responsive, and robust data processing mechanisms will continue to grow. By shedding light on the strengths and weaknesses of various RL-based algorithms in fog-cloud computing, this paper aims to guide researchers, practitioners, and industry stakeholders in their quest for optimal solutions.

The remainder of this paper is structured as follows: Section 2 provides a background on RL and its applications in fog-cloud computing. Section 3 details the various RL-based algorithms considered in this study. Section 4 outlines the evaluation metrics, followed by the comparative analysis in Section 5. We conclude with a discussion and future research directions in Sections 6 and 7, respectively.

2. BACKGROUND

Over the past twenty years, there has been a significant change in how data is processed and stored. At first, cloud computing was a revolutionary solution that allowed for centralized data storage and processing in large data centers [10]. This model worked well for applications that did not require real-time processing. However, with the rise of IoT devices, the need for real-time or near-real-time data processing has become a top priority. Sending data from edge devices to centralized cloud data centers resulted in latency issues, making the traditional cloud model less practical for many IoT applications [11].

In response to these challenges, fog-cloud computing was developed. By decentralizing data processing and bringing computational resources closer to the data source, fog-cloud computing effectively bridged the gap between edge devices and centralized cloud resources [12]. This shift was not just a technological evolution but also a conceptual one that highlighted the importance of processing data where it was created [13].

2.1 Fundamental Principles of Fog-Cloud Computing

Fog-cloud computing operates on several core principles:

- **Decentralization:** Unlike traditional cloud computing, which centralizes data processing in large data centers, fog-cloud computing distributes processing across multiple nodes closer to the data source [14].
- **Low Latency:** By processing data closer to its source, fog-cloud computing significantly reduces transmission delays, ensuring real-time or near-real-time processing [15].
- **Scalability:** Fog-cloud architectures are inherently scalable, allowing for the seamless addition of new nodes as the network grows [16].
- **Efficiency:** By reducing the load on core networks and optimizing bandwidth utilization, fog-cloud computing enhances overall system efficiency [17].

2.2 Reinforcement Learning: An Overview

Reinforcement Learning (RL) is a type of machine learning where agents learn by interacting with their environment. Unlike supervised learning, where models are trained on labelled data, RL operates in an environment where the correct decision or action is not explicitly known. Instead, agents take action, receive feedback through rewards or penalties, and adjust their strategies accordingly [18].

The fundamental components of RL according to [19] include:

- **Agent:** The decision-maker that interacts with the environment.
- **Environment:** The external system with which the agent interacts.
- **State:** A representation of the environment at a given time.

- **Action:** A decision or move made by the agent.
- **Reward:** Feedback received by the agent after acting in a particular state.

The goal of the agent is to maximize the cumulative reward over time, which it achieves by exploring various actions, learning from the feedback, and refining its strategy.

2.3 RL in Fog-Cloud Computing

The dynamic and decentralized nature of fog-cloud computing presents unique challenges, making traditional algorithmic solutions less effective. RL, with its ability to adapt and learn from interactions, has shown significant promise in this domain [20]

Several studies have explored the application of RL in fog-cloud scenarios, including:

- **Task Scheduling:** Using RL to determine the optimal node for processing a given task, considering factors like computational power, energy consumption, and current load [21].
- **Resource Allocation:** Dynamically allocating resources based on current demand and system constraints using RL-based strategies [22].
- **Fault Tolerance:** Implementing RL-based mechanisms to detect, predict, and mitigate faults in fog-cloud systems [23].

2.4 Challenges in Integrating RL with Fog-Cloud Computing

While RL offers numerous advantages, its integration with fog-cloud computing is not straightforward. Challenges include:

- **Sparse Rewards:** In many fog-cloud scenarios, feedback (rewards) may be infrequent, making it challenging for RL agents to learn effectively [24].
- **Continuous State and Action Spaces:** Fog-cloud environments often have continuous state and action spaces, complicating the RL problem [25].
- **Dynamic Environments:** The ever-changing nature of fog-cloud systems, with nodes joining or leaving the network, introduces additional complexities [26].

Despite these challenges, the potential benefits of integrating RL with fog-cloud computing are immense, making it a research area of significant interest and importance.

3. RL-BASED ALGORITHMS FOR FOG-CLOUD COMPUTING

In the dynamic landscape of fog-cloud computing, the need for adaptive and efficient resource management is paramount. Reinforcement Learning (RL) offers a promising avenue, with algorithms tailored to learn and optimize from interactions within such environments. This section delves into various RL algorithms, highlighting their applicability, unique features, and implementation nuances in the context of fog-cloud computing.

3.1 Algorithm A: Q-Learning for Task Scheduling

Description:

Q-Learning, a foundational model-free RL algorithm, has been widely adopted in various domains due to its simplicity and effectiveness. In the context of fog-cloud computing, Q-Learning can be tailored for task scheduling, determining the optimal node for processing a given task based on the current state of the system and historical data.

Implementation Details:

The core of Q-Learning revolves around the Q-table, a matrix where each entry Q(s, a) represents the expected future reward of taking action a in state s. The Q-values are updated iteratively using the Bellman Equation (1):

$$Q(S,a) = (1-a) \times Q(s,a) + \alpha \times \left(r + \gamma \times max_a\, Q(s',a)\right) \quad (1)$$

Where:

- α is the learning rate, determining the weight given to the new update.
- γ is the discount factor, which models the agent's consideration for future rewards.
- r is the immediate reward received after taking action a in state s.

- $\acute{s}$ is the subsequent state after taking action a.

The Q-table is initialized with arbitrary values, and as the agent interacts with the environment, it updates the Q-values based on the received rewards, gradually converging to the optimal policy.

Unique Features:

1. **Model-free Nature:** Q-Learning doesn't rely on a predefined model of the environment's dynamics, granting it versatility and adaptability across various fog-cloud computing scenarios.
2. **Exploration vs. Exploitation:** Through techniques like ε-greedy, Q-Learning ensures a balance between exploring new actions and exploiting known beneficial actions, making it adept at decision-making in dynamic environments.
3. **Convergence Guarantee:** Q-Learning offers a theoretical assurance of converging to the optimal policy, provided specific conditions, such as ensuring every state-action pair is visited infinitely often [27].
4. **Adaptability in Dynamic Environments:** The model-free nature of Q-Learning allows it to readily adapt to changes in the environment, making it suitable for the ever-evolving landscape of fog-cloud systems.
5. **Balanced Decision-making:** The ε-greedy approach in Q-Learning facilitates a balance between exploration and exploitation, ensuring efficient learning and decision-making.
6. **Stability in Learning:** With its convergence guarantee, Q-Learning ensures that the Q-values stabilize over time, leading to consistent and reliable decision-making in fog-cloud systems.

3.2 Algorithm D: DQN (Deep Q-Network) for Load Balancing Description:
DQN is an extension of Q-Learning that employs deep neural networks to approximate the Q-value function. This allows it to handle high-dimensional state spaces efficiently.

Implementation Details:

Instead of a Q-table, DQN uses a neural network to estimate Q-values. The network is trained to minimize the difference between the predicted Q-value and the target Q-value (derived from the Bellman equation). Experience replay, where the algorithm stores past experiences and samples from them for training, and target networks, which stabilize the training process, are key components of DQN. The loss function for DQN is given by Equation (2):

$$L(\theta) = \mathbb{E}_{(s,A,R,s')\sim U(D)}\left[\left(r + \gamma_{\max a'} Q(s',a';\theta^{-}) - Q(s,a;\theta)\right)^{2}\right] \quad (2)$$

Where:

- θ are the parameters of the primary Q-network.
- θ^{-} are the parameters of the target Q-network.
- D is the experience replay buffer.
- $U(D)$ represents a uniform random sample from the buffer.

Unique Features:

1. **Deep Neural Network Approximation:** Unlike traditional Q-Learning that uses a table to store Q-values, DQN employs deep neural networks to approximate the Q-value function. This allows it to handle high-dimensional state spaces, making it scalable for complex environments.
2. **Experience Replay:** DQN introduces the concept of experience replay, where it stores past experiences (state, action, reward, next state) in a replay buffer. During training, it samples mini-batches from this buffer to update the network. This mechanism breaks the correlation between consecutive experiences, stabilizing the training process and improving data efficiency.

3. **Target Network:** DQN uses a separate target network to compute the target Q-values during training. This network's weights are periodically updated with the primary network's weights. The target network stabilizes the learning process by providing fixed Q-value targets for a set number of updates.

4. **Clip Rewards and Normalize Frames:** In specific implementations, especially for tasks like game playing, DQN clips rewards to a fixed range (e.g., -1 to 1) to ensure stable training dynamics. Additionally, input frames (like game screens) are often normalized, ensuring consistent input data scales.

5. **Double DQN Extension:** To address the overestimation bias of Q-values in DQN, the Double DQN (DDQN) variant was introduced. DDQN decouples the action selection from Q-value estimation, leading to more accurate Q-value predictions and improved performance.

6. **Dueling Architecture:** Another variant of DQN, the Dueling DQN, employs a specialized network architecture that separately estimates the state value and the advantage of each action. This separation allows for a more nuanced Q-value estimation, especially in scenarios where the action choice doesn't significantly affect the outcome.

3.3 Algorithm B: Deep Deterministic Policy Gradient (DDPG) for Resource Allocation

Description:

DDPG, an off-policy actor-critic algorithm, is designed for environments with continuous action spaces. Its application in fog-cloud computing can be particularly beneficial for resource allocation, where decisions often lie in a continuous domain, such as allocating a specific amount of bandwidth or computational power.

Implementation Details:

DDPG employs two neural networks: the actor-network, which determines the optimal action, and the critic network, which evaluates the quality of the action taken by the actor. The actor-network is defined by:

$\mu(s \mid \theta^{\mu})$

Where θ^{μ} are the parameters of the actor-network. The critic network estimates the Q-value of the taken action:

$Q(S, a \mid \theta^{\mu})$

Where θ^{μ} are the parameters of the critic network.

The critic is trained using the Bellman equation, similar to Q-Learning, while the actor is trained using the policy gradient derived from the critic's output.

Unique Features:

1. **Continuous Action Spaces:** DDPG's design for continuous action environments makes it adaptable for fog-cloud computing tasks like resource allocation [28].

2. **Actor-Critic Architecture:** The dual network setup of DDPG ensures a balance between action selection and evaluation, enhancing training stability [29].

3. **Experience Replay:** DDPG's use of a replay buffer boosts data efficiency and training stability, ensuring robustness in dynamic environments.

4. **Off-Policy Learning:** DDPG's ability to learn from past experiences allows efficient data utilization and adaptability.

5. **Optimization in Continuous Domains:** DDPG's operation in continuous action spaces makes it suitable for tasks requiring granular control in fog-cloud computing.

6. **Neural Network-Based Estimation:** DDPG's use of neural networks for both actor and critic allow scalability in high-dimensional spaces, making it versatile for complex scenarios [30].

3.4 Algorithm C: Proximal Policy Optimization (PPO) for Fault Tolerance

Description:
PPO, a state-of-the-art policy optimization algorithm, has gained popularity due to its effectiveness and stability across a wide range of environments. In fog-cloud computing, PPO can be instrumental in enhancing fault tolerance, ensuring that the system remains robust and adaptive in the face of unexpected challenges.

Implementation Details:
PPO modifies the standard policy gradient objective to prevent large policy updates, which can destabilize training. The objective function is given by Equation (3):

$$L(\theta) = min\left(\frac{\pi_{\theta(a|s)}}{\pi_{\theta old}((a|s))} A_{\theta old}\,(s,a), clip\left(\frac{\pi_{\theta(a|s)}}{\pi_{\theta old}(a|s)}, 1-\epsilon, 1+\epsilon\right) A_{\theta old}(s,a)\right) \quad (3)$$

Where:

- π is the policy parameterized by θ.
- $A_{\theta old}$ is the advantage function under the old policy.
- ϵ is a hyperparameter that determines the degree of clipping.

The clipping mechanism ensures that the policy updates remain within a specified range, preventing drastic changes that could harm performance.

Unique Features:

1. **Stability:** The implementation of the clipping mechanism in PPO results in a steady and reliable training process, effectively overcoming the issues faced by traditional policy gradient methods [31].
2. **Simplicity and Efficiency:** PPO is a simpler and more efficient alternative to trust region methods. It produces comparable or better results, requiring much less computing power [31].
3. **Adaptability:** PPO has an inherent adaptability that makes it suitable for fog-cloud infrastructures, which are known for their constantly changing environments. This feature allows PPO to adjust and perform well even in dynamic settings [31].

4. EVALUATION METRICS

When evaluating RL algorithms in the context of fog-cloud computing, it is essential to use a comprehensive set of metrics. These metrics not only measure the effectiveness of the algorithms but also highlight areas that need improvement. This section will discuss the key evaluation metrics necessary for assessing RL algorithms in fog-cloud computing scenarios.

4.1 Latency

Definition:

Latency refers to the time taken to process a task from the moment it's initiated to its completion.

Importance:

In fog-cloud computing, where real-time or near-real-time processing is often essential, latency becomes a critical metric. Lower latency ensures faster data processing, which is vital for applications like autonomous driving, real-time analytics, and emergency response systems.

Measurement:

Latency can be measured in milliseconds (ms) or seconds, depending on the application's requirements.

4.2 Energy Consumption

Definition:

Energy consumption measures the amount of energy utilized by the fog nodes during data processing.

Importance:

With the proliferation of IoT devices and edge nodes, energy efficiency becomes paramount. Reducing energy consumption not only saves costs but also enhances the sustainability of the system, especially if nodes are battery-powered.

Measurement:

Energy consumption can be quantified in Joules (J) or watt-hours (WH), depending on the duration and intensity of the tasks.

4.3 Fault Tolerance

Definition:

Fault tolerance assesses the system's ability to continue functioning correctly in the presence of failures or faults within the system.

Importance:

Due to the distributed nature of fog-cloud computing, nodes may encounter failures caused by hardware malfunctions or network problems. Therefore, it is crucial for an RL algorithm to tackle these failures without compromising the overall system performance.

Measurement:

Fault tolerance can be measured as a percentage, representing the system's uptime or the number of successfully processed tasks despite node failures.

4.4 Adaptability

Definition:

The adaptability of an algorithm refers to its capacity to adapt to changes in the system dynamics or the environment.

Importance:

In fog-cloud environments, the nodes are constantly joining or leaving, the workload keeps changing, and the network conditions are never static. Therefore, it is crucial for RL algorithms to be able to adapt to these changes so that they can deliver consistent performance while utilizing resources optimally.

Measurement:

Adaptability can be assessed qualitatively through scenarios or test cases that introduce changes to the environment. Alternatively, it can be quantified by measuring the algorithm's performance deviation before and after the introduced changes.

4.5 Task Throughput

Definition:

Task throughput measures the number of tasks processed per unit of time.

Importance:

In high-demand scenarios, the ability of the system to handle a large number of tasks efficiently is crucial. Higher throughput indicates better system efficiency and resource utilization.

Measurement:

Throughput can be measured in tasks per second (TPS) or tasks per minute (TPM), depending on the system's scale and the granularity of the tasks.

4.6 Resource Utilization

Definition:

Resource utilization evaluates how effectively the computational resources (like CPU, memory, and bandwidth) of the fog nodes are used.

Importance:

Optimal resource utilization ensures that no node is underutilized or overburdened, leading to balanced system performance and prolonged hardware lifespan.

Measurement:

Resource utilization can be quantified as a percentage, indicating the proportion of the resource used compared to its total availability.

In conclusion, these evaluation metrics provide a holistic view of the RL algorithm's performance in fog-cloud computing environments. By considering these metrics, researchers and practitioners can make informed decisions, refine their algorithms, and ensure that the systems meet the desired performance standards.

5. COMPARATIVE ANALYSIS

There are many different RL algorithms that are designed specifically for fog-cloud computing. To determine which algorithm will work best for a particular application, it's important to do a thorough comparative analysis. In this section, we will examine the performance, adaptability, efficiency, and fault tolerance of various RL algorithms and discuss their strengths, weaknesses, and potential uses.

5.1 Performance

Performance is usually the primary metric of interest when evaluating algorithms. It measures how well an algorithm performs based on predefined metrics such as latency, task throughput, and energy consumption.

- **Q-Learning:** is a method that uses tables and is known for its quick convergence in smaller state-action spaces, resulting in reduced latency. However, in larger and more complex environments, its performance may decrease due to the curse of dimensionality.

- **DQN:** With the use of its deep neural network, DQN is capable of efficiently managing complex tasks that involve high-dimensional state spaces. This makes it a desirable option for intricate fog-cloud computing scenarios.
- **DDPG:** With its continuous action space optimization, DDPG often outperforms in scenarios demanding fine-grained resource allocation, ensuring optimal task throughput and energy efficiency. However, its reliance on neural networks might introduce slight latency in decision-making.
- **PPO:** When it comes to fog-cloud scenarios, PPO is known for its stability and consistent performance. The balance between exploration and exploitation allows for optimal latency and energy consumption metrics. The clipped objective function is a key factor in achieving these results.

5.2 Adaptability

In the ever-evolving landscape of fog-cloud computing, the ability of an algorithm to adapt to changing conditions is paramount.

- **Q-Learning:** While Q-Learning can adapt to changes, its table-based nature might make it slower to respond to drastic shifts in the environment, especially if the Q-table is large.
- **DQN:** The neural network in DQN allows it to generalize across states, making it adaptable to changing conditions. However, it might require retraining if the changes are drastic.
- **DDPG:** The actor-critic architecture of DDPG allows it to swiftly adapt to changes. The actor, guided by the critic, can quickly adjust its policy to accommodate new environmental dynamics.
- **PPO:** PPO's adaptability is one of its hallmarks. The algorithm's objective function, designed to prevent large policy updates, ensures that the agent remains adaptable without making drastic, potentially harmful changes to its policy.

5.3 Efficiency

Efficiency encompasses both resource consumption and computational overhead, determining the feasibility of deploying an algorithm in resource-constrained fog nodes.

- **Q-Learning:** Q-Learning, being a simpler algorithm, often has lower computational overhead. However, in large state-action spaces, the memory requirement for the Q-table can become a bottleneck.
- **DQN:** While DQN can handle larger state spaces than traditional Q-Learning, its reliance on deep neural networks increases computational overhead, especially during training.
- **DDPG:** DDPG's neural network-based approach can be computationally intensive, especially during training. However, once trained, the actor network's forward pass to determine actions is relatively efficient. Memory consumption, primarily due to the replay buffer, can be significant but is manageable with efficient data structures.
- **PPO:** PPO strikes a balance between computational and memory efficiency. While it employs neural networks like DDPG, its simpler objective function often leads to faster convergence and reduced training overhead.

5.4 Fault Tolerance

In distributed systems like fog-cloud computing, failures are unavoidable. The resilience of an algorithm in the face of such challenges determines its robustness and reliability.

- **Q-Learning:** Q-Learning's deterministic nature can be a double-edged sword. While it ensures consistent decision-making, it might struggle in the face of unexpected system failures, unless explicitly trained for such scenarios.

- **DQN**: experience replay mechanism allows it to learn from past mistakes, enhancing its fault tolerance. However, like other RL algorithms, it's only as robust as the range of experiences it has been exposed to.

- **DDPG:** DDPG's continuous action space optimization allows it to find alternative strategies in case of node failures or resource constraints. The replay buffer, storing past experiences, can help the algorithm learn from previous failures and enhance its fault tolerance.

- **PPO:** PPO's emphasis on stable and incremental policy updates ensures that the algorithm doesn't make rash decisions even when faced with failures. Its ability to learn from both positive and negative rewards makes it inherently fault-tolerant, as it can adjust its policy based on feedback from the environment.

To summarize, the effectiveness of each RL algorithm depends on the specific needs and limitations of the fog-cloud computing scenario. A thorough comprehension of these algorithms, combined with extensive testing and validation, will enable their successful implementation in real-life situations.

6. DISCUSSION

The combination of Reinforcement Learning (RL) and fog-cloud computing is a merging of two revolutionary fields, each presenting unique challenges and opportunities [32]. As we delve into the intricacies of algorithms such as Q-Learning, DDPG, PPO, and DQN, it's clear that the landscape is varied and intricate. This discussion aims to provide multiple perspectives on these algorithms, providing valuable insights that can inform future research and applications.

6.1 Strengths and Weaknesses of Each Algorithm:

- **Q-Learning:**
    - **Strengths:** Simplicity, rapid convergence in smaller state-action spaces, deterministic policy.
    - **Weaknesses:** Struggles with large state-action spaces due to the curse of dimensionality; table-based approach might need to generalize better across states.
- **DQN:**
    - **Strengths:** Scalability to high-dimensional state spaces, experience replay stabilizes training, target network ensures consistent Q-value targets.
    - **Weaknesses:** Potential overestimation of Q-values, reliance on deep networks increases computational overhead
- **DDPG:**
    - **Strengths:** Tailored for continuous action spaces, actor-critic architecture balances action selection and evaluation, and experience replay enhances data efficiency.
    - **Weaknesses:** Reliance on neural networks can introduce latency, and might require careful hyperparameter tuning.
- **PPO:**
    - **Strengths:** Stable and consistent training, adaptability to changing environments, simpler alternative to trust region-based methods.
    - **Weaknesses:** Might require more samples for training compared to other algorithms, hyperparameter sensitivity.

6.2 Deep Dive into Algorithmic Nuances:

Q-Learning's Deterministic Policy: Although Q-Learning deterministic policy provides consistency, it may limit the algorithm's ability to explore new strategies, especially in dynamic fog-cloud environments where adaptability is essential [33].

DQN Scalability: DQN ability to handle high-dimensional state spaces is impressive, but it raises concerns about the interpretability of its decisions, which is a vital aspect when deploying in critical applications.

DDPG Continuous Action Space: DDPG optimization of continuous action space offers precise control, but it also presents challenges in terms of convergence stability, particularly in environments with sharp reward gradients.

PPO Balance: PPO prioritizes stable policy updates, which is beneficial, but it also means the algorithm may be slower to adapt to sudden, significant changes in the environment.

6.3 Practical Implications of the Findings:
Beyond the metrics, the real-world implications of these algorithms are profound:

- **Infrastructure Considerations:** The choice of algorithm can influence infrastructure requirements. For instance, DDPG and DQN, with their reliance on deep networks, might necessitate more robust computational resources, impacting deployment costs.

- **Adaptability in Dynamic Environments:** In the ever-evolving landscape of fog-cloud computing, where nodes can frequently join or leave the network, the adaptability of algorithms like PPO becomes invaluable.

- **Safety and Reliability:** In applications where safety is paramount, the deterministic nature of Q-Learning might be preferred, even if it sacrifices some level of optimality.

6.4 Recommendations for Practitioners and Researchers:

1. **Bespoke Algorithm Design:** Consider designing algorithms tailored to specific fog-cloud scenarios, leveraging the strengths of existing algorithms while mitigating their weaknesses.

2. **Ethical Considerations:** As RL algorithms make decisions that can impact real-world systems, ethical considerations, especially in terms of fairness, transparency, and accountability, become paramount.

3. **Interdisciplinary Collaboration:** Engage with experts from fields like distributed computing, network design, and ethics to ensure a holistic approach to RL in fog-cloud computing.

4. **Future Directions:** Explore the potential of combining RL with other AI domains, like transfer learning or meta-learning, to enhance the adaptability and efficiency of algorithms in fog-cloud environments.

As we delve into the intricacies of RL in fog-cloud computing, it becomes clear that we have only just started our journey [34]. While there are numerous challenges, there are also many opportunities. By promoting a collaborative culture that emphasizes continuous learning and ethical practices, we can pave the way for RL's effective and responsible implementation in fog-cloud systems.

7. CONCLUSION AND FUTURE WORKS

In our study of Reinforcement Learning (RL) in the field of fog-cloud computing, we have encountered many complexities, challenges, and possibilities. There are various RL algorithms, each with unique strengths and weaknesses, so it's important to choose the appropriate one based on the context. Q-Learning is deterministic, PPO is adaptable, and DQN is scalable, and each has far-reaching implications beyond just theoretical performance. As computational resources become more distributed and edge devices become more prevalent, RL's role in optimizing, enhancing, and ensuring the robustness of these systems becomes increasingly significant.
While there are promising opportunities for the future, there are also challenges. The integration of RL in fog-cloud computing isn't just about algorithmic efficiency, but it also involves ethical considerations, real-world applicability, and scalability. We've made significant progress, but there's still a long way to go. Interdisciplinary collaboration, a deeper understanding of real-world dynamics, and a commitment to ethical and transparent research are essential. RL and fog-cloud computing offer a frontier of opportunities, but we must use them in a holistic, inclusive, and forward-thinking way to transform distributed computing.

REFERENCES

[1] S. E. Shukri, R. Al-Sayyed, H. Al-Bdour, E. Alhenawi, T. Almarabeh, and H. Mohammad, “Internet of Things: Underwater routing based on user’s health status for smart diving,” *International Journal of Data and Network Science*, vol. 7, no. 4, pp. 1715–1728, 2023, doi: 10.5267/J.IJDNS.2023.7.019.

[2] Y.-A. Daraghmi, E. Y. Daraghmi, R. Daraghma, H. Fouchal, and M. Ayaida, “Edge-Fog-Cloud Computing Hierarchy for Improving Performance and Security of NB-IoT-Based Health Monitoring Systems,” 2022, doi: 10.3390/s22228646.

[3] M. A. M. Al-Alwan, S. S. Al-Nawafah, H. M. Al-Shorman, F. A. Khrisat, F. F. Alathamneh, and S. I. S. Al-Hawary, “The effect of big data on decision quality: Evidence from telecommunication industry,” *International Journal of Data and Network Science*, vol. 6, no. 3, pp. 693–702, Jun. 2022, doi: 10.5267/J.IJDNS.2022.4.003.

[4] S. R. Hassan, M. Rashad, S. R. Hassan, and M. Rashad, “Cloud Computing to Fog Computing: A Paradigm Shift,” *Edge Computing - Technology, Management and Integration [Working Title]*, Apr. 2023, doi: 10.5772/INTECHOPEN.110751.

[5] M. Sheikh Sofla, M. Haghi Kashani, E. Mahdipour, and R. Faghih Mirzaee, “Towards effective offloading mechanisms in fog computing,” *Multimed Tools Appl*, vol. 81, no. 2, pp. 1997–2042, Jan. 2022, doi: 10.1007/S11042-021-11423-9/FIGURES/10.

[6] M. Hajvali, S. Adabi, A. Rezaee, and M. Hosseinzadeh, “Decentralized and scalable hybrid scheduling-clustering method for real-time applications in volatile and dynamic Fog-Cloud Environments,” *Journal of Cloud Computing*, vol. 12, no. 1, pp. 1–35, Dec. 2023, doi: 10.1186/S13677-023-00428-4/FIGURES/21.

[7] F. Alenizi and O. Rana, “Dynamically Controlling Offloading Thresholds in Fog Systems,” *Sensors (Basel)*, vol. 21, no. 7, Apr. 2021, doi: 10.3390/S21072512.

[8] M. Zdravković, H. Panetto, and G. Weichhart, “AI-enabled Enterprise Information Systems for Manufacturing,” vol. 2022, no. 4, pp. 688–720, doi: 10.1080/17517575.2021.1941275ï.

[9] P. Lahande, P. Kaveri, and J. Saini, “Reinforcement Learning for Reducing the Interruptions and Increasing Fault Tolerance in the Cloud Environment,” *Informatics 2023, Vol. 10, Page 64*, vol. 10, no. 3, p. 64, Aug. 2023, doi: 10.3390/INFORMATICS10030064.

[10] M. Al Rawajbeh *et al.*, “A new model for security analysis of network anomalies for IoT devices,” *International Journal of Data and Network Science*, vol. 7, no. 3, pp. 1241–1248, 2023, doi: 10.5267/J.IJDNS.2023.5.001.

[11] R. O. Aburukba, T. Landolsi, and D. Omer, “A heuristic scheduling approach for fog-cloud computing environment with stationary IoT devices,” *Journal of Network and Computer Applications*, vol. 180, p. 102994, 2021, doi: 10.1016/j.jnca.2021.102994.

[12] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, “On Reducing IoT Service Delay via Fog Offloading,” in *IEEE Internet of Things Journal*, Institute of Electrical and Electronics Engineers Inc., Apr. 2018, pp. 998–1010. doi: 10.1109/JIOT.2017.2788802.

[13] N. A. Perifanis and F. Kitsios, “Edge and Fog Computing Business Value Streams through IoT Solutions: A Literature Review for Strategic Implementation,” *Information 2022, Vol. 13, Page 427*, vol. 13, no. 9, p. 427, Sep. 2022, doi: 10.3390/INFO13090427.

[14] R. O. Aburukba, M. AliKarrar, T. Landolsi, and K. El-Fakih, “Scheduling Internet of Things requests to minimize latency in hybrid Fog–Cloud computing,” *Future Generation Computer Systems*, vol. 111, pp. 539–551, 2020, doi: 10.1016/j.future.2019.09.039.

[15] J. Li, X. Li, J. Yuan, and G. Li, “Load Balanced Data Transmission Strategy Based on Cloud–Edge–End Collaboration in the Internet of Things,” *Sustainability 2022, Vol. 14, Page 9602*, vol. 14, no. 15, p. 9602, Aug. 2022, doi: 10.3390/SU14159602.

[16] F. Firouzi, B. Farahani, and A. Marinšek, “The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT),” *Inf Syst*, vol. 107, p. 101840, Jul. 2022, doi: 10.1016/J.IS.2021.101840.

[17] M. Vijarania, S. Gupta, A. Agrawal, M. O. Adigun, S. A. Ajagbe, and J. B. Awotunde, “Energy Efficient Load-Balancing Mechanism in Integrated IoT–Fog–Cloud Environment,” *Electronics 2023, Vol. 12, Page 2543*, vol. 12, no. 11, p. 2543, Jun. 2023, doi: 10.3390/ELECTRONICS12112543.

[18] S. A. Fayaz, S. J. Sidiq, M. Zaman, and M. A. Butt, “Machine Learning: An Introduction to Reinforcement Learning,” *Machine Learning and Data Science: Fundamentals and Applications*, pp. 1–22, May 2021, doi: 10.1002/9781119776499.CH1.

[19] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. 2018. Accessed: Aug. 22, 2023. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=uWV0DwAAQBAJ&oi=fnd&pg=PR7&ots=miwGt7Z-l4&sig=9Az_l_Xu-ERX04wzIb7ne8pad1o

[20] S. Iftikhar *et al.*, "AI-based fog and edge computing: A systematic review, taxonomy and future directions," *Internet of Things*, vol. 21, p. 100674, Apr. 2023, doi: 10.1016/J.IOT.2022.100674.

[21] F. M. Talaat, M. S. Saraya, A. I. Saleh, H. A. Ali, and S. H. Ali, "A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment," *J Ambient Intell Humaniz Comput*, vol. 11, no. 11, pp. 4951–4966, Nov. 2020, doi: 10.1007/S12652-020-01768-8.

[22] M. Zangooei, N. Saha, M. Golkarifard, and R. Boutaba, "Reinforcement Learning for Radio Resource Management in RAN Slicing: A Survey," *IEEE Communications Magazine*, vol. 61, no. 2, pp. 118–124, Feb. 2023, doi: 10.1109/MCOM.004.2200532.

[23] R. Besharati, M. H. Rezvani, and M. M. Gilanian Sadeghi, "An Auction-Based Bid Prediction Mechanism for Fog-Cloud Offloading Using Q-Learning," *Complexity*, vol. 2023, 2023, doi: 10.1155/2023/5222504.

[24] L. Yu, T. Yu, C. Finn, and S. Ermon, "Meta-Inverse Reinforcement Learning with Probabilistic Context Variables," *Adv Neural Inf Process Syst*, vol. 32, 2019.

[25] D. H. Abdulazeez and S. K. Askar, "Offloading Mechanisms Based on Reinforcement Learning and Deep Learning Algorithms in the Fog Computing Environment," *IEEE Access*, vol. 11, pp. 12554–12585, 2023, doi: 10.1109/ACCESS.2023.3241881.

[26] R. K. Naha, S. Garg, and M. B. Amin, "Fuzzy Logic-based Robust Failure Handling Mechanism for Fuzzy Logic-based Robust Failure Handling Mechanism for Fog Computing," *arXiv preprint arXiv:2103.06381.*, Mar. 2021, Accessed: Apr. 20, 2021. [Online]. Available: http://arxiv.org/abs/2103.06381

[27] Y. Xie, S. Mou, and S. Sundaram, "Communication-Efficient and Resilient Distributed <inline-formula> <tex-math notation="LaTeX">Q</tex-math> </inline-formula>-Learning," *IEEE Trans Neural Netw Learn Syst*, 2023, doi: 10.1109/TNNLS.2023.3292036.

[28] H. Tran-Dang, S. Bhardwaj, T. Rahim, A. Musaddiq, and D.-S. Kim, "Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues," *Journal of Communications and Networks*, vol. 24, no. 1, pp. 83–98, Feb. 2022, doi: 10.23919/JCN.2021.000041.

[29] Q. Yang and R. Parasuraman, "BSAC: Bayesian Strategy Network Based Soft Actor-Critic in Deep Reinforcement Learning," Aug. 2022, Accessed: Aug. 23, 2023. [Online]. Available: https://arxiv.org/abs/2208.06033v1

[30] V. Padhye, K. Lakshmanan, and A. Chaturvedi, "Proximal policy optimization based hybrid recommender systems for large scale recommendations," *Multimed Tools Appl*, vol. 82, no. 13, pp. 20079–20100, May 2023, doi: 10.1007/S11042-022-14231-X/METRICS.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. K. Openai, "Proximal Policy Optimization Algorithms," Jul. 2017, Accessed: Aug. 23, 2023. [Online]. Available: https://arxiv.org/abs/1707.06347v2

[32] M. K. Pandit, R. N. Mir, and M. A. Chishti, "Adaptive task scheduling in IoT using reinforcement learning," *International Journal of Intelligent Computing and Cybernetics*, vol. 13, no. 3, pp. 261–282, Aug. 2020, doi: 10.1108/IJICC-03-2020-0021/FULL/XML.

[33] M. M. Alam and S. Moh, "Survey on Q-Learning-Based Position-Aware Routing Protocols in Flying Ad Hoc Networks," *Electronics 2022, Vol. 11, Page 1099*, vol. 11, no. 7, p. 1099, Mar. 2022, doi: 10.3390/ELECTRONICS11071099.

[34] Z. Wang and T. Hong, "Reinforcement learning for building controls: The opportunities and challenges," *Appl Energy*, vol. 269, p. 115036, Jul. 2020, doi: 10.1016/J.APENERGY.2020.115036.