10.48047/jocaaa.2024.33.06.190

Adaptive Block Size Optimization in Block-chain Networks Using Artificial Intelligence Techniques

Kavindra Kumar
Research Scholar, Department of Computer Science,
Jayoti Vidyapeeth Women's University
Jaipur, India
Dr. Sourabh Kumar jain
Professor, Jayoti Vidyapeeth Women's University, Jaipur, India

ABSTRACT

This paper explored the use of reinforcement learning (RL) to optimise block size in blockchain networks in order to solve scalability and performance issues related to increasing transaction volumes and data needs. Although the decentralised and transparent nature of blockchain technology increases security and trust, problems like lengthy transaction wait times and poor resource allocation limit its effectiveness, especially in vital industries like banking and healthcare. The paper suggests a dynamic RL-based framework that minimises overhead and maximises resource allocation by modelling miners as adaptive agents that learn from their interactions within the blockchain ecosystem. The approach simulates blockchain networks using MATLAB R2020 to gather synthetic data on block sizes, throughput, and latency. The results are then compared to previous research in the field. The results demonstrate how well the RL model balances throughput and latency, opening the door to enhanced scalability and operational performance in blockchain systems and successfully resolving the particular difficulties presented by decentralised networks.

Keywords*: Block Size Optimization, Reinforcement Learning, Block Chain networks.*

INTRODUCTION

Blockchain technology has been extensively used in almost all industrial sectors, including the healthcare sector, cryptocurrencies artificial intelligence the government sector, internet of things, and others. Blockchain offers a decentralized, transparent, and unchangeable ledger for transactions, which improves participant security and confidence. Organizations are using blockchain more and more for a wide range of applications, which has increased demand for scalable and effective solutions and exposed important issues with scalability and performance [1].

10.48047/jocaaa.2024.33.06.190

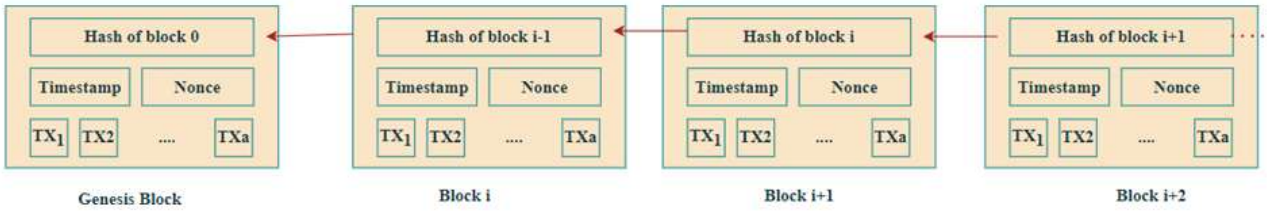


Figure 1 Block chain structure

Blockchain technology has revolutionized several sectors by offering secure and open transaction platforms. The growing number of blockchain applications has led to issues with block size and transaction wait times. These challenges significantly influence the efficacy and scalability of blockchain networks[2].

Reducing transaction wait times is crucial, particularly in sensitive areas like healthcare and banking. Smooth financial transactions need quick confirmation times to ensure rapid settlements and allow users to complete activities without delay. Long transaction wait times may be catastrophic in medical fields, where timely patient data access is crucial for accurate diagnosis and prompt treatments [3]. Reduced waiting times may enhance financial and medical service efficiency, improve patient care results, and boost customer happiness.

Blockchain network scalability and transaction throughput depend on block size optimization. In our research, dynamically adjusting block size using Artificial Intelligence is crucial as transaction volumes and data needs expand. Intelligent, adaptive block size mechanisms can balance throughput, latency, and resource usage in real time [4]. As network demand rises, this optimization keeps decentralized apps (dApps) running smoothly, enhancing user experience and adoption. Blockchain networks can accommodate higher transaction volumes while retaining decentralization and avoiding latency by using AI-driven block size determination methods. This is critical for future scalability[5].

Many methods have been suggested to optimize block size in blockchain networks, however they frequently have poor accuracy, excessive processing cost, or lack of scalability. This paper proposes using AI, specifically reinforcement learning (RL), to dynamically decide the ideal block size. To balance transaction confirmation times and block size, the suggested technique uses a deep learning model that analyzes transaction data, current network circumstances. This adjustable block size approach improves blockchain systems' efficiency, scalability, and user experience by helping them manage variable transaction volumes.

Aim and Objectives

The goal of this project is to create an AI-driven methodology for dynamically modifying

10.48047/jocaaa.2024.33.06.190

blockchain block size depending on real-time transaction load in order to improve performance, speed, and resource usage.

- To investigate the impact of block size on blockchain performance, including transaction speed, latency, and resource utilization.
- To develop a model using artificial intelligence that can determine the optimal block size based on the current transaction load.
- To evaluate the performance improvements of an adaptive block size mechanism compared to static configurations.

Problem Formulation

Reinforcement learning (RL) is a powerful technique for minimizing block size in blockchain networks because it constantly adapts to changing transaction loads and network circumstances. Unlike static block sizes, which may result in inefficiencies such as underutilization during slow traffic times or congestion during high transaction volumes, RL allows the system to alter block sizes in real time. This method enables the blockchain to learn from its surroundings, such as previous transaction data, current network condition, and feedback on system performance, in order to find the best block size for increasing transaction throughput, reducing latency, and improving resource efficiency. By doing so, RL can solve critical difficulties in blockchain scalability and performance, guaranteeing that the network can manage changing transaction demands while preserving system stability and increasing user experience.

This research uses reinforcement learning (RL), which can learn adaptively and make judgements in uncertain and dynamic contexts, to optimise block size in blockchain networks[6]. Due to factors like variable network conditions and transaction volumes, traditional optimisation techniques sometimes cannot handle the complexity and variances present in blockchain applications. Real-time resource allocation is made possible by reinforcement learning, which models miners as agents that constantly interact with their surroundings. This balance between throughput and latency is achieved by identifying the ideal block size. Reinforcement learning is especially well-suited for sensibly and intelligently enhancing the operational performance of blockchain networks because of its adaptive approach, which not only improves efficiency and scalability but also tackles the issues presented by decentralised systems.

LITERATURE REVIEW

10.48047/jocaaa.2024.33.06.190

[7] addressed the blockchain-based secure e-voting system's performance. The study examined performance and scalability limits related to population size, block creation rate, and transaction speed. Block size significantly impacts the performance and scalability of blockchain-based systems. Blockchain-based systems are significantly impacted by block size,

10.48047/jocaaa.2024.33.06.190

transaction count, and generation rate. But [8]proposed a pioneering effort on the scalability of blockchain-based systems. His research highlighted practical limits, including latency, throughput, start-up time, and transaction costs. These factors and their impact on networks are outlined both realistically and broadly. It is a useful resource for identifying scalability concerns. However, the research did not maximize latency and throughput based on mathematical principles. Whereas, [9] introduced sharding mechanism to decrease cross-shard transactions and latency for blockchain scalability and performance. It offers modular account reconfiguration and blockchain transaction account graph network parallel transaction processing. Tests show 34.7% fewer cross-shard transactions, 83.2% lower latency, 52.7% greater throughput, and 7.8% fewer account migrations, improving blockchain performance for different applications.

According to [10], the pace of transaction processing in a blockchain depends on two parameters: block size and block interval. The authors propose a scalable Bitcoin NG protocol that improves throughput and reduces network latencies. [11] analyzed the scalability of the bitcoin blockchain and identifies elements that affect block size and throughput. They consider the advantages and disadvantages of raising or lowering block size. According to [12], reparameterizing block size is a significant step towards creating scalable blockchains capable of handling increased demands. [13] proposed a multiobjective optimisation problem (OP) to minimise block production and transmission time. This multiobjective OP is reduced to a single OP using weighted sum approach. Particle Swarm Optimization (PSO) and Whale Optimization Algorithms (WOA) are used to identify appropriate block size. While both algorithms may approach optimal block size and duration, WOA outperforms PSO in convergence speed and output fluctuation. Additionally, analysing the prediction of optimal block size under various weights generates several optimisation algorithms. Experimental findings show that assigning more weight to transmission time leads to a significant drop in block size. The blockchain's storage optimisation for Internet of Things systems was first described by [14] as a block selection issue for cloud storage. For this purpose, they suggested using the NSGA-C algorithm. On average, they achieved a 30% reduction in storage overhead. On the other hand, their method's run-time was somewhat longer than the benchmarks. The AT-MOPSO method was created by [15] in order to enhance the work of [14]. Compared to the NSGA-C, the AT-MOPSO algorithm exhibited a notably reduced run-time and increased energy efficiency. On one of the trade-off goals, this algorithm performed worse than NSGA-C. [16] proposed Deep reinforcement learning -based performance optimisation methodology

10.48047/jocaaa.2024.33.06.190

for blockchain-enabled IIoT systems has three objectives: 1) Evaluating system scalability, decentralisation, latency, and security; 2) improving blockchain scalability without affecting security; and 3) designing a modulable blockchain for IIoT systems that allows DRL-based selection/adjustment of block producers, consensus algorithm, block size, and block interval. In simulations, our framework improves blockchain-enabled IIoT system performance and adapts to their dynamics. [16] Optimised SeHealthChain (OSeHealthChain) uses a Tuna Swarm Optimisation Algorithm to dynamically alter settings to improve blockchain security and speed. It outperforms current methods with peak throughputs of 7051, 6418, and 6290 tps under different attack scenarios with 3918 tps and 63.8 seconds latency for 1000 nodes.According to [17] PuffChain automatically adjusts throughput depending on user demand to overcome blockchain scalability challenges. For optimised processing, it separates node roles into packers, proposers, and validators utilising a three-phase consensus mechanism. With 100 nodes, PuffChain can process 6061 transactions per second, demonstrating its practicality.

Research gap

We discovered many study gaps on blockchain performance and scalability from the earlier literature. While previous research has brought attention to important performance measures like block size and transaction speed, these studies often lack a thorough mathematical framework that would allow for the efficient optimization of these variables. Furthermore, even though sharding technologies have been created to improve blockchain speed, nothing is known about how well they work with different types of blockchain applications. Although it is an important aspect for scalability, the dynamic nature of block size alterations in response to changing network needs has not been well explored. Additionally, there isn't a complete analysis of how block size reparameterization affects other scalability aspects like transaction costs and start-up time. Above all, there is a clear research gap that has not been filled in the area of using Reinforcement Learning (RL) approaches for block size optimization. In order to close these gaps and perhaps enhance the efficiency and scalability of blockchain systems, this research suggests an RL-based optimization model for block size and transaction throughput.

Methodology

For the purpose of optimizing mining strategies within blockchain networks, this methodology implements a reinforcement learning (RL) framework. It defines miners as agents that adjust their actions in response to observations from the blockchain environment, operating through

10.48047/jocaaa.2024.33.06.190

a sequence of episodes that involve multiple mining cycles. The approach prioritizes the efficient allocation of resources across fragments, which is facilitated by a well-organized reward function that effectively balances the optimization of earnings and the minimization of overhead. The model that is based on RL enables miners to proactively respond to the challenges posed by a decentralized and rapidly changing landscape by continuous updating of local policies.

Data collection

MATLAB R2020 was used to simulate blockchain networks and provide data. The main goal was to gather artificial data on block sizes, throughput, and latency in order to use Reinforcement Learning (RL) to optimise block sizes in blockchain networks. In addition to continuously monitoring resource allocation and cumulative rewards in real-time, the RL agent also made adjustments to cache requests in response to the network's present state. To confirm that these results are consistent with previous studies on blockchain optimisation and reinforcement learning, a literature analysis was also carried out to compare and verify the results. Validating the efficacy of the suggested optimisation techniques required the data gathered from this simulation exercise.

RL-MODEL for Optimization

The Optimization Model for Reinforcement Learning (RL) is a dynamic framework that improves the decision-making process of miners by optimizing resource allocation in competitive environments. In the end, it allows agents to optimize rewards and reduce costs by learning and adapting their strategies through interactions with their environments.

1. **Agent**

The entities running a learning process and upholding a local prediction model are called agents. In this work, every miner i optimizes resource allocation in a blockchain system as an autonomous agent.

$$Agent_i learns: \theta_i^{(t)} \leftarrow \theta_i^{(t-1)} + \alpha \nabla L_i(\theta_i^{(t-1)}) \quad (1)$$

Where $\theta_i^{(t)}$ is agent i's parameter vector at time t, the learning rate is represented by α, while agent i's loss function is represented by Li.

2. **Environment**

10.48047/jocaaa.2024.33.06.190

The environment is represented as a black box that returns the states that arise from the agents' activities as input as shown in **fig 2** . The setting in this research is a blockchain network that all miners share, which is mathematically expressed as follows:

$$Env(A) \rightarrow S \quad (2)$$

where A is the vector of actions taken by all agents and S is the resulting state vector.

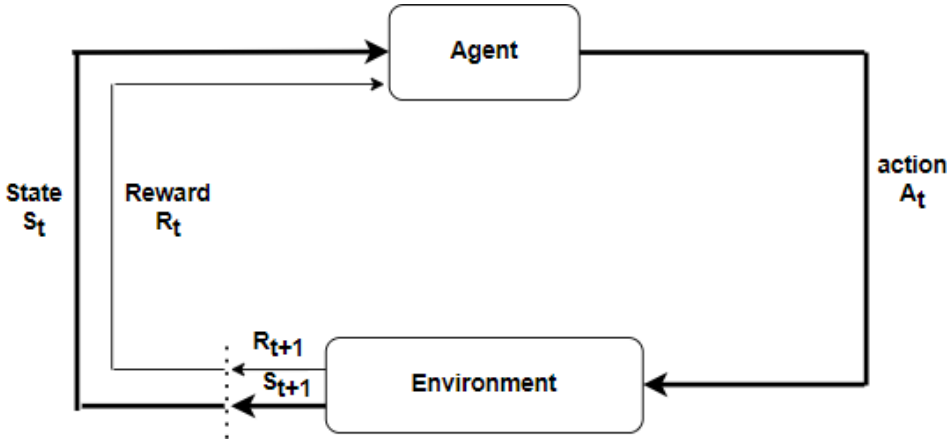


Figure 2 Agent and Environment Interaction

3. Episodes

An episode consists of a predetermined number of mining rounds, K, with no specified ending. The episode might be stated as follows:

$$Episode_E = \{Mining\ Round_1, Mining\ Round_2, \ldots, Mining\ Round_K\} \quad (3)$$

Each mining cycle (k) may be represented as a tuple of actions and states.

$$Mining\ Round_k(A_k, S_k) \quad (4)$$

4. State (S)

Each agent (miner i) gets discrete observations of the environment, indicated by s ∈ S. The state si of miner i indicates the distribution of successful block validations over N shards:

$$s_i = [\nu_{i,1}, \nu_{i,2}, \ldots, \nu_{i,N}] \quad (5)$$

where ν_{ij} is the number of successful validations for miner i in shard j. The probability distribution of states can be computed as:

$$P(si) = \frac{\nu_{i,j}}{\sum_{j=1}^{N} \nu_{i,j}} \quad \forall j \in \{1, \ldots, N\} \quad (6)$$

10.48047/jocaaa.2024.33.06.190

5. Actions A

Each agent takes choices depending on the local policy determined by its learning model. The valid action set for miner i is the allocation of computing resources across all N shards, represented by:

$$a_i = \rho_i = [\delta_{i,1}, \delta_{i,2}, \ldots, \delta_{i,N}] \quad (7)$$

To distinguish resource allocation, we separate it into large-scale and small-scale acts.

$$\delta_{i,j} \in \{0.1R_i, 0.01Ri\} \, for \, eachshard \; j \quad (8)$$

This produces an organized action space, allowing for quicker convergence during training.

6. Policy

The local policy is constantly modified based on observations from the environment. The policy determines the likelihood of adopting action ai in a particular situation si.

$$\pi(s_i, a_i) = P(a_i \mid s_i) \; (9)$$

The policy gradient approach may be used for updating the policy.

$$\theta_i^{(t)} \leftarrow \theta_i^{(t-1)} + \alpha \nabla J(\theta_i) \; (10)$$

The anticipated reward function, denoted by $J\,(\theta i)$, is defined over the trajectory of states and actions.

7. Reward Function R

A well-designed reward function improves the effectiveness of a local policy by providing feedback on the mapping from $S \rightarrow A$. In episode E, the reward function for miner i is as follows:

$$R_i(\alpha_i, j) = \left(\lambda K \sum_{j=1}^{N} \alpha_{i,j} \Phi \right) - (\Omega_{PoW,i} + \Omega_{Comm,i}) \; (11)$$

- $\alpha_{i,j}$ - is the reward attributed to miner i from shard j,
- Φ- is the success rate of transactions processed by miner i,
- $\Omega_{PoW,i}$- represents the total computational cost for miner i,
- $\Omega_{Comm,}$ - captures communication overhead for miner i.

The amount of the rewards is as follows:

$$R_i = \sum_{j=1} R_i(\alpha_i, j) \, (12)$$

8. Rational Decision Making

During the mining process, all miners are logical actors looking to maximize their own profits[14]. The challenge of optimization may be expressed as follows:

10.48047/jocaaa.2024.33.06.190

$$Maximize \sum_{i=1}^{K} R_i \quad (13)$$

Depending on the availability of resources,

$$\sum_{j=1}^{N} \delta_{i,j} \leq R_i^{total} \forall_i \quad (14)$$

This will improve each stage by providing more rigorous mathematical expressions related to our research on optimization in a blockchain network using reinforcement learning.

Algorithm 1 RL-Based Block Size Optimization

Initialize environment (blockchain network)
Initialize RL agent with parameters (learning rate, exploration strategy)
Define state space S (current block size, network conditions)
Define action space A (increase, decrease, maintain block size)
Define reward function R:

$$R_i = \alpha_{i,j} \cdot \Phi - \Omega_{\text{PoW},\,i} - \Omega_{\text{Comm},\,i}$$

for each episode e **do**
  Reset environment and initialize state s_i
  Initialize cumulative rewards and other tracking variables
  for each mining cycle k in episode e **do**
    Select action a_i based on current policy $\pi(s_i)$
    Execute action a_i, adjust block size
    Observe reward R_i and next state s'_i
    Store experience (s_i, a_i, R_i, s'_i)
    Update policy:

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \alpha \cdot \nabla L_i$$

    Update current state to next state s'_i
  end for
  Evaluate performance metrics (cumulative rewards, throughput, latency)
  Adjust hyperparameters if necessary
end for
Terminate training when predefined criteria are met

Simulation environment

To simulate our AI model, we used MATLAB version R2020, using its extensive features for blockchain scenario modelling and algorithm development. In order to allow real-time optimisation in the calculation of ideal block sizes, this version incorporates machine learning algorithms. The flowchart illustrates the steps involved in the RL-based block size optimisation process. First, the environment and the reinforcement learning agent are initialised. Next, the state and action spaces that are essential for decision-making in the blockchain network are defined[20]. In order to encourage effective block size management while taking communication and processing costs into consideration, the reward function was created. Block

10.48047/jocaaa.2024.33.06.190

size is repeatedly adjusted by the training loop based on feedback, and hyperparameters are fine-tuned to improve performance metrics like latency and throughput[13]. Furthermore, the visualisation capabilities of MATLAB allow for a thorough analysis of simulation results, offering insights into every stage of the procedure from termination checks to policy updates ensuring that our optimisation strategies are carefully evaluated before being deployed in real blockchain networks [15].

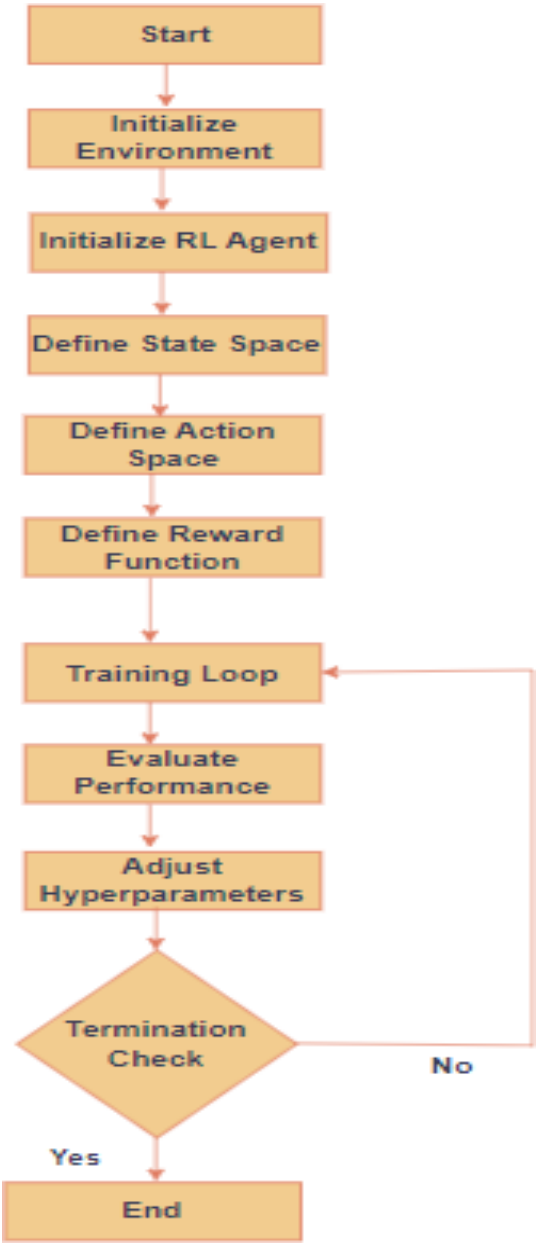


Figure 3 Flowchart

10.48047/jocaaa.2024.33.06.190

The process of system optimisation using Reinforcement Learning (RL), which is applicable to blockchain optimisation in this instance. Based on the flowchart, the following describes each step:

Start: The process begins with the initiation of the simulation.

Initialize Environment: This stage involves creating a simulation environment to replicate the system or blockchain network that is being optimised. The RL agent will interact with all pertinent variables and limitations (block size, throughput, latency, etc.) in this environment.

Initialize RL Agent: Here, several initialisation settings are applied to the RL agent. In order to maximise reward, the agent must make choices depending on environmental factors. Optimising important blockchain metrics (such block size) is the agent's job.

Define State Space: The environment's potential states are all included in the state space. Block size, transaction throughput, and network latency, for example, might all be considered states in a blockchain setting at any one time. To choose the optimal course of action, the RL agent looks at the existing situation.

Define Action Space: The action space defines the set of actions that the RL agent can take. For example, in this blockchain scenario, actions could involve adjusting block size or modifying resource allocation based on the network's performance**.**

Define Reward Function: How the agent is compensated for its behaviours is specified by the reward function. In this instance, the enhancement in throughput, decrease in latency, and optimisation of block size might serve as the foundation for the reward function. Better performance is rewarded with a larger payout, which directs the agent towards the best course of action.

Training Loop: The RL agent enters a loop where it repeatedly interacts with the environment. During each iteration, the agent observes the current state, takes an action, and receives feedback (reward). The agent learns through this process and improves its decision-making over time.

Evaluate Performance: The RL agent's performance is assessed after a number of iterations. This entails assessing the extent to which the agent's choices are enhancing the blockchain's critical metrics, such as latency and throughput.

10.48047/jocaaa.2024.33.06.190

Adjust Hyperparameters: Hyperparameters (such as learning rate, discount factor, etc.) are changed based on the assessment to optimise the agent's performance. Optimising hyperparameters is crucial to guaranteeing optimum learning and performance from the agent.

Termination Check: The system verifies whether the requirements for termination are satisfied. This could depend on how well the agent performs or how many iterations there are. The procedure concludes if the requirements are satisfied; if not, the agent stays in the training loop.

End: Once the termination condition is satisfied, the process concludes, and the RL agent has learned the optimal strategy for the given environment**.**

Simulation Parameters and Conditions:

Block Size: To determine how different block sizes affect latency and performance, simulations of various block sizes are run. These block sizes might be tiny or huge, mimicking the fluctuations in blockchain use that occur in the real world.

Throughput: The amount of data the blockchain network can handle in a certain amount of time is measured by its throughput. Block size is modified by the RL agent to increase throughput without taxing the network's capacity.

Latency: The time it takes for a block to be approved and put to the chain is known as the "latency." The RL agent balances block size and resource allocation to minimise delay.

Network conditions: A variety of network variables, including fluctuating bandwidth, different numbers of active nodes, and cache request rates, are also included in the simulation. By simulating real-world circumstances, these conditions enable the RL agent to dynamically adjust to changes in the blockchain network.

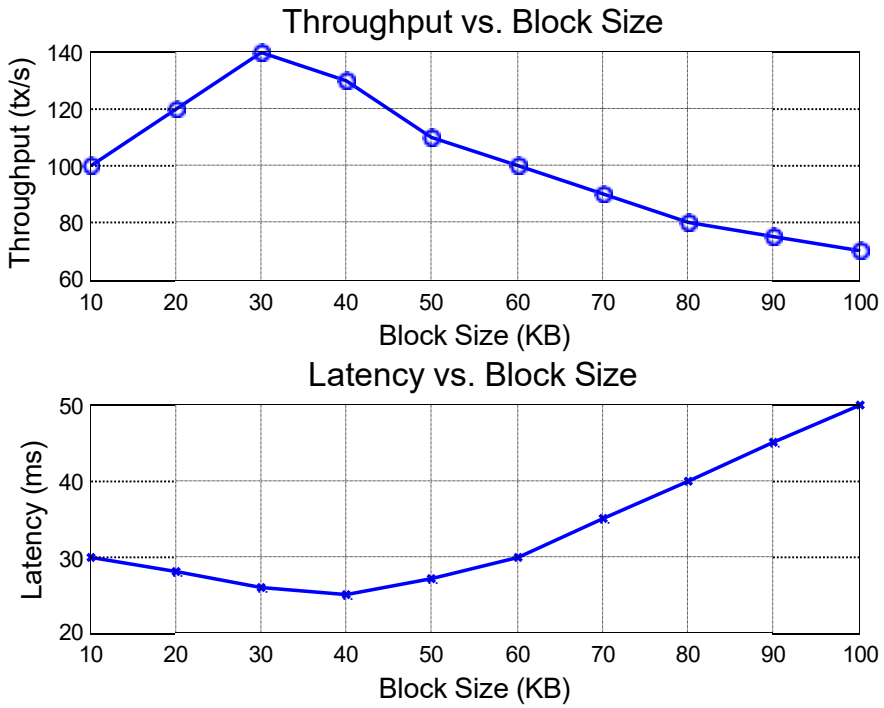
RESULTS AND DISCUSSION

Results

This section covers the findings from our research on optimising block size in blockchain networks using Reinforcement Learning (RL). The results demonstrate the efficiency of our RL technique in adapting to various network circumstances while maximising performance indicators such as cumulative rewards, throughput, and latency. We use a series of charts to show the link between cumulative average rewards and the number of caching requesters, the

10.48047/jocaaa.2024.33.06.190

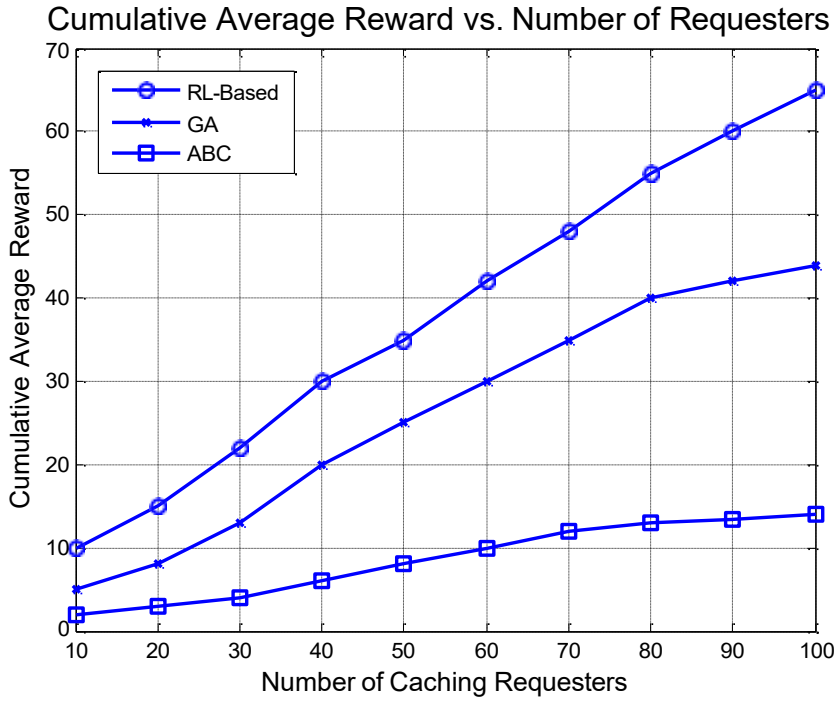
impacts of block size on throughput and latency, and the influence of the learning rate on the agent's performance. These findings support the stability of our optimisation model and demonstrate the potential for RL approaches to improve blockchain efficiency and scalability.



As the block size is changed, we see a notable trade-off between these two performance metrics, as shown by the plots of Throughput vs. Block Size and Latency vs. Block Size. Throughput first rises with block size, indicating the ability to handle more transactions per second, but ultimately falls when network congestion occurs. On the other hand, latency shows an opposite connection, peaking at a smaller block size and then increasing once again. This dual behaviour emphasises how important it is to optimise block size in order to improve network performance as a whole [13].

In order to tackle this difficulty, we present a reinforcement learning (RL) agent that is intended to dynamically optimise block size by using real-time performance feedback. The agent is taught to determine the ideal block size that efficiently strikes a balance between high throughput and low latency by using the adaptive capabilities of reinforcement learning. This improves the efficiency of blockchain activities. The RL-based approach to block size optimisation in blockchain networks will be shown by the next plot, which will display the Cumulative Rewards attained by the RL agent during the training phase .

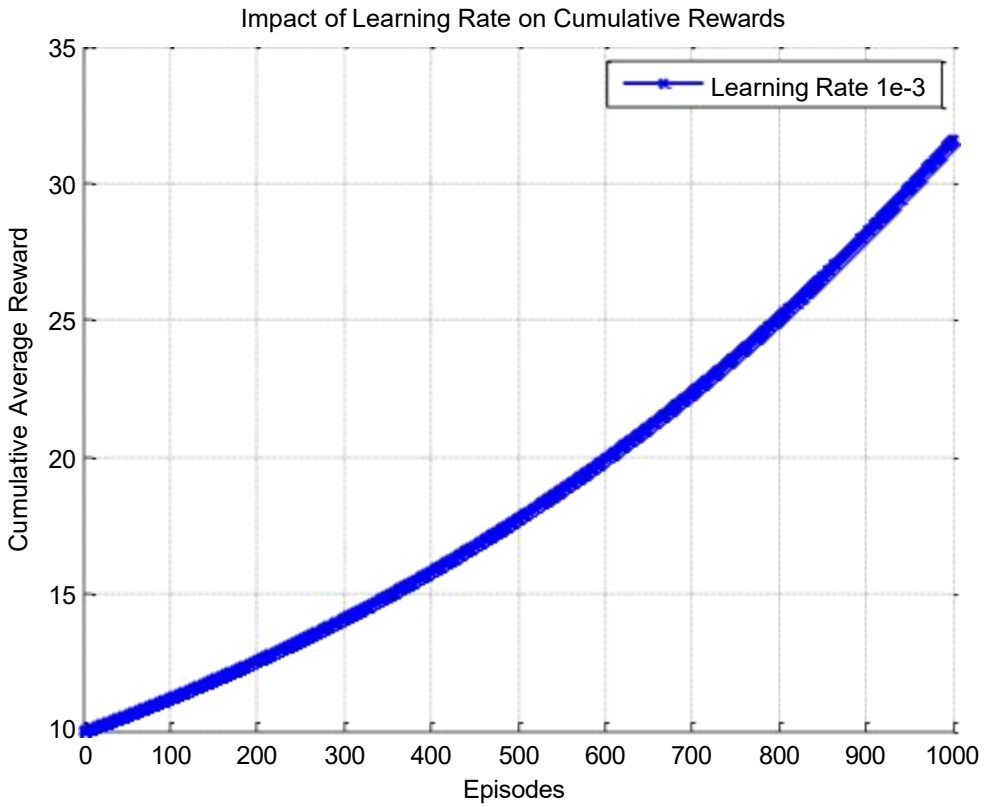
10.48047/jocaaa.2024.33.06.190



The plot of Cumulative Average Reward vs. Number of Requesters effectively illustrates the impact of our reinforcement learning (RL) algorithm on optimizing blockchain performance under varying network conditions. As the number of cache requesters increases, the figure demonstrates that our RL agent's cumulative rewards adjust accordingly, consistently maintaining or improving performance compared to traditional optimization methods. This characteristic emphasizes the stability and scalability of our optimization model, showcasing how our approach outperforms conventional techniques by dynamically allocating resources in response to changing demands [24].

Such adaptability is crucial in real-world blockchain systems, where transaction volumes can fluctuate significantly. By effectiely managing these fluctuations, our RL-based block size optimization strategy enhances both network throughput and user satisfaction, corroborating the assertion that intelligent resource allocation can lead to more efficient and responsive blockchain environments. Overall, this plot reinforces the notion that employing RL can substantially improve the operational performance of blockchain networks.

10.48047/jocaaa.2024.33.06.190



Plotting the Effect of Learning Rate on Cumulative Rewards shows how the reinforcement learning (RL) agent performs over time at a 1e-3 learning rate. The agent is effectively learning to optimise block size as shown by the cumulative incentives, which steadily rise as the episodes go on. The capacity of the agent to modify its decision-making in light of prior experiences is shown by this increasing tendency, which emphasises the significance of choosing an ideal learning rate to promote efficient learning [25].

An optimal learning rate allows the agent to make effective strategy adjustments without going crazy or being too cautious, which might degrade performance. Because it directly affects the learning process's stability and rate of convergence, the balance struck by adjusting this hyperparameter is crucial to reinforcement learning. The agent's continuous performance increase throughout the block size optimisation procedure in blockchain networks, as seen in this figure, highlights the importance of hyperparameter optimisation in reinforcement learning overall.

DISCUSSION

The results of our research on optimising block size in blockchain networks using Reinforcement Learning (RL) provide vital insights into the trade-off between throughput and

10.48047/jocaaa.2024.33.06.190

latency, emphasising the need of carefully modifying block size to improve network performance. While higher block sizes may initially boost throughput, they eventually increase delay, emphasising the significance of dynamic optimisation algorithms that react to changing network circumstances. Our implementation of an RL agent to alter block size based on real-time performance input is a big step forward, enabling the agent to successfully determine the ideal block size that balances high throughput and low latency. This versatility is critical for real-world applications, where transaction volumes might vary greatly. Furthermore, comparing cumulative incentives to the number of cache requesters shows that the agent can maintain or improve performance when compared to standard optimisation approaches. Furthermore, the influence of the learning rate on cumulative rewards emphasises the significance of hyperparameter tuning, since a consistent rise in rewards with a learning rate of 1e-3 suggests successful learning without unpredictable behaviour. Overall, our findings add to the discussion of blockchain optimisation by indicating that RL approaches provide interesting answers to scalability and efficiency issues, opening the way for further study in this field.

This study has limitations despite encouraging results. First, the simulation was run under predetermined parameters, which may not mimic real-world blockchain setups with security flaws or network threats. RL agent performance was studied on a single network model, and its scalability across blockchain platforms is unknown. RL training's computational expense may hinder its real-time application in high-frequency transaction networks.

By studying blockchain scenarios with security risks and multi-platform assessments, future research may overcome these restrictions. Researching computational cost reduction and hyperparameter tweaking might improve RL's real-time usability in blockchain networks. Blending RL with other machine learning methods or classical algorithms may help optimise blockchain scalability and efficiency. RL in decentralised systems might improve performance and user satisfaction via several paths.

CONCLUSION

To summarise, this research successfully studied the optimisation of block size in blockchain networks using Reinforcement Learning (RL) approaches. The findings show a considerable trade-off between throughput and latency, demonstrating that although higher block sizes might initially improve throughput, they eventually increase latency over a certain threshold.

10.48047/jocaaa.2024.33.06.190

This conclusion highlights the need of dynamic optimisation algorithms that can successfully adapt to changing network circumstances.

We illustrated the potential of adaptive approaches to balance conflicting performance indicators by building an RL agent that learns to alter block size in response to real-time performance input. The ability of the RL agent to maintain or enhance cumulative rewards as the number of cache requesters rises demonstrates our optimisation model's resilience and scalability when compared to older techniques.

Furthermore, the effect of the learning rate on cumulative rewards demonstrated the significance of hyperparameter tweaking in reinforcement learning. The successful implementation of a learning rate of 1e-3 resulted in constant performance gains, demonstrating the agent's ability to learn effectively and without unpredictable behaviour. Overall, the outcomes of this study bring vital insights to the current discussion about blockchain optimisation, indicating that RL approaches offer interesting answers to scaling issues while improving overall network performance. Future research may expand on these findings to better RL applications in blockchain environments, opening the door for greater efficiency and user happiness in decentralised systems.

All things considered, our study provides a workable solution to the block size optimisation issue in blockchain networks, providing a flexible, scalable, and effective way to manage throughput against latency trade-offs. These results imply that RL offers a great deal of promise to improve overall network performance while tackling scalability issues on the blockchain. Subsequent investigations may enhance this methodology and investigate alternative reinforcement learning implementations to bolster efficacy and user contentment in decentralised systems.

References

[1] A. Hafid, A. S. Hafid, and M. Samih, “Scaling Blockchains: A Comprehensive Survey,” *IEEE Access*, vol. 8, pp. 125244–125262, 2020, doi: 10.1109/ACCESS.2020.3007251.

[2] N. Papadis, S. Borst, A. Walid, M. Grissa, and L. Tassiulas, “Stochastic Models and Wide-Area Network Measurements for Blockchain Design and Analysis,” *Proc. - IEEE INFOCOM*, vol. 2018-April, no. September, pp. 2546–2554, 2018, doi: 10.1109/INFOCOM.2018.8485982.

[3] T. K. Vashishth, V. Sharma, K. K. Sharma, B. Kumar, S. Chaudhary, and R. Panwar,

10.48047/jocaaa.2024.33.06.190

"Intelligent resource allocation and optimization for industrial robotics using AI and blockchain," *AI Blockchain Appl. Ind. Robot.*, no. December, pp. 82–110, 2023, doi: 10.4018/979-8-3693-0659-8.ch004.

[4] N. Singh and M. Vardhan, "Computing Optimal Block Size for Blockchain based Applications with Contradictory Objectives," *Procedia Comput. Sci.*, vol. 171, no. 2019, pp. 1389–1398, 2020, doi: 10.1016/j.procs.2020.04.149.

[5] A. W. Salehi *et al.*, "A Study of CNN and Transfer Learning in Medical Imaging: Advantages, Challenges, Future Scope," *Sustainability*, vol. 15, no. 7. 2023. doi: 10.3390/su15075930.

[6] K. M. Khan, J. Arshad, and M. M. Khan, "Investigating performance constraints for blockchain based secure e-voting system," *Futur. Gener. Comput. Syst.*, vol. 105, pp. 13–26, 2020, doi: 10.1016/j.future.2019.11.005.

[7] A. P. Paper, K. Croman, C. Decker, I. Eyal, A. E. Gencer, and A. Juels, "On Scaling Decentralized Blockchains | SpringerLink," 2016.

[8] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-NG: A scalable blockchain protocol," *Proc. 13th USENIX Symp. Networked Syst. Des. Implementation, NSDI 2016*, pp. 45–59, 2016.

[9] B. Santander, "Scalability Analysis of BCT through blockchain simulation," , vol. 87, no. 1,2, pp. 149–200, 2017.

[10] A. Kimbonguila, L. Matos, J. Petit, J. Scher, and J.-M. Nzikou, "Effect of Physical Treatment on the Physicochemical, Rheological and Functional Properties of Yam Meal of the Cultivar 'Ngumvu' From Dioscorea Alata L. of Congo," *Int. J. Recent Sci. Res.*, vol. 10, pp. 30693–30695, 2019, doi: 10.24327/IJRSR.

[11] B. Aygün and H. Arslan, "Block size optimization for PoW consensus algorithm based blockchain applications by using whale optimization algorithm," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 30, no. 2, pp. 406–419, 2022, doi: 10.3906/elk-2105-217.

[12] M. Xu, G. Feng, Y. Ren, and X. Zhang, "On cloud storage optimization of blockchain with a clustering-based genetic algorithm," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8547–8558, 2020.

[13] C. Nartey *et al.*, "Blockchain-IoT peer device storage optimization using an advanced

10.48047/jocaaa.2024.33.06.190

time-variant multi-objective particle swarm optimization algorithm," *EURASIP J. Wirel. Commun. Netw.*, vol. 2022, no. 1, p. 5, 2022.

[14] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Performance optimization for blockchain-enabled industrial internet of things (iiot) systems: A deep reinforcement learning approach," *IEEE Trans. Ind. Informatics*, vol. 15, no. 6, pp. 3559–3570, 2019, doi: 10.1109/TII.2019.2897805.

[15] A. Gupta and V. K. Chaurasiya, "Reinforcement learning based energy management in wireless body area network: A survey," in *2019 IEEE Conference on Information and Communication Technology*, IEEE, 2019, pp. 1–6.

[16] H. Xiao, C. Qiu, Q. Yang, H. Huang, J. Wang, and C. Su, "Deep reinforcement learning for optimal resource allocation in blockchain-based IoV secure systems," in *2020 16th International Conference on Mobility, Sensing and Networking (MSN)*, IEEE, 2020, pp. 137–144.

[17] C. D. Paternina-Arboleda, J. R. Montoya-Torres, and A. Fabregas-Ariza, "Simulation-optimization using a reinforcement learning approach," in *2008 Winter Simulation Conference*, IEEE, 2008, pp. 1376–1383.

[18] P. Mars and A. O'Sullivan, "Reinforcement learning and A* search for the unit commitment problem," *Energy AI*, vol. 9, p. 100179, Jul. 2022, doi: 10.1016/j.egyai.2022.100179.

[19] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, "Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains," *IEEE Trans. Ind. informatics*, vol. 13, no. 6, pp. 3154–3164, 2017.

[20] Z. Liu, L. Gao, Y. Liu, X. Guan, K. Ma, and Y. Wang, "Efficient QoS support for robust resource allocation in blockchain-based femtocell networks," *IEEE Trans. Ind. informatics*, vol. 16, no. 11, pp. 7070–7080, 2019.

[21] Y. Huang, C. Xu, C. Zhang, M. Hua, and Z. Zhang, "An overview of intelligent wireless communications using deep reinforcement learning," *J. Commun. Inf. Networks*, vol. 4, no. 2, pp. 15–29, 2019.