

ROBB: Recurrent Proximal Policy Optimization Reinforcement Learning for Optimal Block Formation in Bitcoin Blockchain Network

by

Amit Dutta
21166028

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
M.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
July 2023

© 2023. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Amit Dutta
21166028

Approval

The thesis/project titled "ROBB: Recurrent Proximal Policy Optimization Reinforcement Learning for Optimal Block Formation in Bitcoin Blockchain Network" submitted by

1. Amit Dutta (21166028)

Of Summer, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of M.Sc. in Computer Science on July 26, 2023.

Examining Committee:

Supervisor:
(Member)



Dr. Md. Golam Rabiul Alam
Professor

Department of Computer Science and Engineering
BRAC University

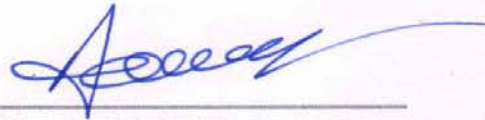
Examiner:
(External)



Prof. Mohammad Mehedi Hassan, Ph.D.
Professor

Information Systems Department
King Saud University, Riyadh

Examiner:
(Internal)



Dr. Amitabha Chakrabarty
Professor

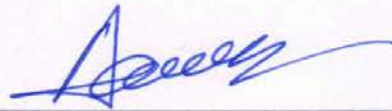
Department of Computer Science and Engineering
BRAC University

Examiner:
(Internal)



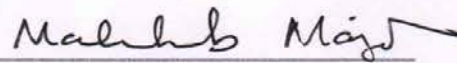
Dr. Md. Ashraful Alam
Associate Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)



Dr. Amitabha Chakrabarty
Professor
Department of Computer Science and Engineering
BRAC University

Chairperson:
(Chair)



Sadia Hamid Kazi, Ph.D.
Associate Professor,
Department of Computer Science and Engineering
BRAC University

Abstract

Blockchain is a ground-breaking technology that has changed how we manage and store protected data. It is a decentralized ledger that enables safe, open, and unchangeable record-keeping. It relies on a distributed network of nodes rather than a single central authority to check and verify transactions, guaranteeing that each entry is correct and unchangeable. Transactions in a blockchain network are grouped into blocks, which are then linked together in a chronological and immutable chain. Block size is a critical parameter in blockchain technology, which refers to the maximum size of each block in the chain. However, we cannot just change the block size of the blockchain. It is challenging and will create security issues. The Block size is crucial because it affects the number of transactions processed per second, the confirmation time, and overall network efficiency. The confirmation time should be faster to ensure stable earnings for the miners. Moreover, it needs help with broader applications due to high transaction fees and long verification times. We have proposed a reinforcement learning model named ROBB that can efficiently create a block considering the current network state and previous transactions. At first, the problem was converted into a reinforcement learning environment to solve using multiple reinforcement algorithms. We developed a blockchain simulator to replicate the network environment. To transform it into a reinforcement learning environment, we integrated it with OpenAI Gym. The simulator was trained by generating random transactions. Finally, we designed a reward function that enables the simulator to hold transactions and create blocks with the pending transactions when it determines that the environment is favourable. In the final results, ROBB successfully minimized the waiting time for transactions and utilized the blocks to their full potential, which is crucial for private blockchains used in medical records. Additionally, it optimized the block space, building upon the findings of previous researchers.

Keywords: Reinforcement Learning; Machine Learning; Blockchain; Recurrent neural network; OpenAiGym; Proximal Policy Optimization,, RNN

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.

Secondly, to our advisor **Dr. Md. Golam Rabiul Alam** sir for his kind support and advice in our work. He helped us whenever we needed help.

And finally to our parents without their throughout sup-port it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

Declaration	i
Approval	ii
Abstract	iv
Dedication	v
Acknowledgment	v
Table of Contents	vi
List of Figures	viii
List of Tables	ix
Nomenclature	x
1 Introduction	1
1.1 Research Motivation	2
1.2 Research Objective	2
1.3 Research Challenges	3
1.4 Research Contributions	3
1.5 Thesis Orientation	4
2 Related Work	5
3 Background Study	9
3.1 Blockchain components	9
3.1.1 Block	9
3.1.2 Hash	10
3.1.3 Chain	12
3.1.4 Transaction	12
3.1.5 Consensus Mechanism	13
3.1.6 Peer-to-Peer Network	15
3.1.7 Distributed Ledger	17
3.2 Blockchain Architecture	18

4	Methodologies	22
4.1	Proposed Scheme	22
4.1.1	Policy and value network as Predictive Model	24
4.1.2	RPPO based reinforcement learning agent design	25
5	Result analysis	30
5.1	Blockchain Simulator for ROBB	30
5.1.1	Defining Blockchain	31
5.1.2	Defining Block	31
5.1.3	Defining Transaction	32
5.1.4	Simulator	32
5.2	Blockchain environment	33
5.2.1	Experimentation with multiple algorithms	33
5.2.2	Testing environment	35
5.3	Model training	37
5.4	Reward tuning	38
5.5	Comparison	38
6	Conclusion	43
6.1	Conclusion	43
6.2	Future Plan	44
	Bibliography	47

List of Figures

3.1	Structure of a block in Blockchain	9
3.2	Consensus Mechanism	14
3.3	P2P network diagram	15
3.4	Current bitcoin blockchain Architecture	18
4.1	ROBB system workflow	23
5.1	Blockchain Simulator	30
5.2	ROBB Blockchain Environment	34
5.3	Testing environment Fixed block size	35
5.4	Testing environment random block size	36
5.5	Value loss over time	37
5.6	Policy loss over time	37
5.7	Comparison between different models and their waiting time	40
5.8	Comparison between different models and their block utilization	41
5.9	Model vs waiting time and block utilization	41

List of Tables

4.1	Constants values for reward function	29
5.1	Reward function tuning result	38
5.2	Performance of different models (DQN, PPO, A2C, RPPO)	38
5.3	Model vs waiting time and block utilization	39

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ϵ	Epsilon
3	Stable baseline 3
v	Upsilon
A	Action
b_r	block size utilization rate
P	Pending transaction in mempool
p_r	Pending transaction utilization rate
RL	Reinforcement Learning
T	Transaction included in the block
W	Average waiting time
w_c	Waiting time utilization rate for creating block
w_h	Waiting time utilization rate for holding
BW	Block wastage
MB	Megabyte
MDP	Markov Decision Process
PPO	Proximal policy optimization
RNN	Recurrent Neural network
ROBB	Recurrent Proximal Policy Optimization in Reinforcement Learning for Optimal Block Formation in Bitcoin Blockchain Network
RPPO	Recurrent proximal policy optimization

Chapter 1

Introduction

Blockchain is a decentralized digital ledger [8] that eliminates intermediaries to facilitate secure and transparent transactions. The block size, which determines the maximum amount of data that can be contained in a single block, is one of the critical factors in blockchain technology. The transaction size has a significant impact on the efficiency and scalability of a blockchain system. The volume of transactions and data saved on the blockchain has recently increased due to the blockchain applications' explosive expansion [26]. As a result, there is a demand for effective, scalable blockchain systems that can manage massive amounts of data. Therefore, the timing of block creation has emerged as a crucial problem in blockchain research. Due to the explosive growth of blockchain applications [13], the quantity of transactions and data preserved on the blockchain has increased recently. As a result, there is a need for efficient, scalable blockchain systems that can manage vast quantities of data. Consequently, the timing of block creation has emerged as a significant issue in blockchain research. Reducing transaction wait periods is necessary for a variety of reasons. First, short confirmation intervals are essential for facilitating seamless and efficient financial transactions[28], particularly in cryptocurrency systems. Users anticipate that their transactions will be processed expeditiously, enabling them to continue with their intended actions uninterrupted. In addition, private blockchains, such as those utilized in sensitive sectors such as medical data, place a greater emphasis on reducing transaction wait times. In healthcare settings where rapid access to patient data is crucial for accurate diagnoses and prompt treatment, lengthy processing delay times may result in life-threatening conditions. Significantly increasing the effectiveness and efficiency of medical services would enhance patient care outcomes. This can be accomplished by streamlining and accelerating the process of transaction confirmation. Additionally, the scalability and throughput of a blockchain network can be enhanced by decreasing transaction delay times [28]. A network with faster transaction confirmations guarantees the seamless operation of decentralized applications, which can manage a greater volume of transactions and encourage greater user adoption. The best block size has been predicted using various methods, including statistical models, time-series analysis, and machine learning methods. These methods frequently have drawbacks, too, like poor accuracy, high processing complexity, or a lack of scalability [9]. Our research presents a novel method for RL-based blockchain block size optimization. RL is a method of machine learning that learns through feedback and experience. Our method employs a deep neural network to determine the optimal ways to construct

blocks and maintain transactions based on historical data and the current state of the network. In addition, we have carefully calibrated our reward function to prevent excessively frequent or excessively large block formation. Instead, we aim to identify the optimal point for constructing blocks.

1.1 Research Motivation

Blockchain technology has changed several industries by providing safe and open transactional platforms. However, as the number of blockchain apps increases, problems with block size and transaction wait times have also surfaced. These difficulties directly impact blockchain networks' effectiveness, scalability, and user experience. It's critical to cut down on transaction wait times, especially for sensitive industries like healthcare and finance. Quick confirmation times are crucial for smooth financial transactions since they guarantee prompt settlements and let users carry out their tasks without being held up. Similarly, lengthy transaction wait times can have fatal results in the medical field, where quick access to patient data is essential for precise diagnoses and rapid therapies. The efficiency and effectiveness of financial and medical services can be significantly improved by reducing these waiting times, which will also improve patient care outcomes and customer satisfaction.

Adjusting the block size is crucial for a blockchain network's overall scalability and throughput. A thoughtful block size algorithm is essential to meet the expanding needs as the volume of transactions and data recorded on the blockchain rises. Decentralized apps can run smoothly and promote greater user adoption when block sizes are determined efficiently, allowing blockchain networks to manage higher transaction volumes.

Numerous prediction methods have been proposed to solve these issues, but they frequently have drawbacks like low accuracy, computational cost, or scalability. To forecast the ideal block size, this research intends to present a novel strategy that uses RL. This method tries to identify the ideal balance between transaction waiting time and block size using a deep neural network, historical data, and the current network state, improving blockchain systems' effectiveness, scalability, and user experience. In conclusion, the necessity to address the issues with transaction waiting times and block sizes in blockchain networks is the driving force behind this research. This project intends to optimize the block size determination process, minimizing transaction waiting time, and enhancing the overall efficiency and scalability of blockchain systems by proposing a novel reinforcement learning-based approach. As a result of this research, blockchain technology will be more widely used and developed across various industries, including finance, healthcare, and other areas.

1.2 Research Objective

The primary goal of this project is to construct a unique model that uses reinforcement learning to figure out the best course of action for holding and creating blocks in blockchain systems. The suggested model seeks to learn from experience and feedback to identify the ideal timing for block formation, minimizing transaction waiting time and enhancing the effectiveness of blockchain networks. It does this by utilizing a deep neural network and historical data. A correctly crafted reward

function will direct the model's behaviour, achieving a balance between producing blocks too frequently and/or with overly big sizes. The ultimate objective is to optimize the block production process using reinforcement learning techniques to improve the performance, scalability, and user experience of blockchain systems.

1.3 Research Challenges

There are many issues in the current Bitcoin which makes it slow and inefficient, but we focused on a specific problem. In doing so, we have faced some limitations. At first, there was no universal simulator for training blockchain using machine learning, Some papers proposed some simulators, but those do not work in our settings. Secondly, historical data can be used to create a simulator, but we do not have access to every second of the data. Instead, we have daily data, which is not optimal. Apart from these, our proposed architecture is novel, and there is not much work to optimize the block size of a block.

1.4 Research Contributions

To improve transaction management in decentralized blockchain networks, this paper will propose a novel approach that combines blockchain simulation and reinforcement learning approaches. We want to create a blockchain-like environment utilizing OpenAI Gym and create a reward mechanism to direct a reinforcement learning agent's decision-making. We aim to improve transaction processing effectiveness and overall system throughput in blockchain networks by comparing the performance of several reinforcement learning methods. The research work has the following contributions:

- This research proposes an effective blockchain architecture (ROBB) that leverages reinforcement learning to enhance transaction management within a blockchain-based system. By minimizing waiting time and reducing block wastage, the proposed architecture aims to optimize the system's overall efficiency.
- By optimizing the block-creating cycle, transaction throughput has also increased, assuring a higher profit for miners.
- Our proposed system tackles the scalability problem for Bitcoin during peak hours; also, it can efficiently generate blocks while necessary.
- We proposed a reward function that has been thought- fully created to address the blockchain transaction management issue. Our reward function offers a comprehensive assessment metric for RL agents by considering elements like the backlog of remaining transactions, the time it takes to create a block, and successful transaction confirmations. The reward function directs the agent, allowing it to learn the best transaction management techniques that increase system throughput and efficiency.
- To conduct the research, we have also proposed a blockchain simulator where we can train any reinforcement learning algorithms.

Overall, this study offers a complete analysis of the application of blockchain simulation, and RL approaches to enhance transaction management. The suggested approach can fundamentally alter the efficiency and scalability of blockchain networks, leading to a boost in transaction throughput and a more durable decentralized environment.

1.5 Thesis Orientation

This is how we organize the paper.

- Chapter 1 is an Introduction where motivation, problem statement, objectives, and contribution is discussed.
- In Chapter 2, we discussed the related works regarding the optimization of the blockchain.
- In chapter 3, we have discussed the background study of the system. We also discussed different terminology and the current blockchain architecture.
- Then, in chapter 4, we discussed our proposed model and briefly discussed how our system works.
- In Chapter 5, we have discussed our model, which is more efficient than our current model. We have also discussed the different environments used throughout this paper. We also discussed the final results of the model in different settings.
- In chapter 6, we write a simple conclusion of the paper and discuss the future plan for blockchain optimization.

Chapter 2

Related Work

Numerous studies are currently being conducted to explore the performance aspects of blockchain technology. Some studies show that improving the blockchain's main architecture may solve this performance issue. And blockchain, at its current architecture, has scalable issues due to its block size restrictions. Many researchers try to implement machine learning with blockchain to improve the performance of blockchain. But a few developers have tried to work on the scalability issue, so we have worked on the performance and scalability issues in the blockchain. Besides these issues, many researchers create a simulated environment so that other researchers can apply their architecture and measure performance. But most of the simulator does not focus on the scalability issue.

In this [10] paper author mentioned the scalability issue of the Bitcoin network as Nakamoto introduced a block size limit of 1MB for every block in the public Blockchain [1]. This was a security measure, so the P2P network would immediately reject any block over that limit. The author also discusses possible solutions that have been implemented or will be implemented in the future, like Segwit, Sharding, and Proof of Stake. The author also identifies the problem of the speed of transaction verification.

The paper [10] offers a thorough analysis of the obstacles and opportunities for advancing blockchain technology. Recognizing the limitations of blockchain in terms of scalability, security, privacy, and consensus mechanisms, the authors explore possible enhancements in each of these areas. Regarding scalability, the paper examines the concept of sharding, which entails dividing the blockchain into smaller, more manageable pieces. Sharding enables parallel transaction processing, thereby increasing the network's throughput and capacity. The authors also discuss off-chain solutions such as payment channels and sidechains, which can alleviate the burden on the main blockchain by conducting transactions off-chain and intermittently settling them. The authors examine cryptographic protocols that can improve blockchain technology's security and privacy. It examines techniques such as zero-knowledge proofs, ring signatures, and homomorphic encryption, which can enhance the privacy of transaction details and identity protection. The research offers valuable insights into the possibilities for enhancing blockchain technology, as well as suggestions for future research. By resolving the challenges of scalability, security, privacy, and consensus, this paper contributes to the development of blockchain technology

and paves the way for its widespread adoption across a variety of industries and applications.

The authors of paper [17] provide a thorough analysis of the efficacy of machine learning techniques applied to Bitcoin mining, providing valuable insight into the potential benefits and limitations of this approach. The study focuses on a number of crucial aspects, including the investigation of hash rate enhancement, energy efficiency advances, and overall profitability attained by employing machine learning-based mining algorithms. The authors investigate and contrast a variety of machine learning algorithms, including support vector machines, random forests, and deep learning models, in order to assess their efficacy in enhancing mining operations. The research methodology entails thorough data preprocessing, technique-based feature selection, and the application of suitable evaluation metrics. The dataset used in the experiments is described, including its size, origin, and pertinent characteristics, with an emphasis on transparency and limitations. Using quantitative metrics, visualizations, and in-depth analysis, the authors present experimental results comparing the performance of machine learning algorithms to conventional mining techniques. The paper offers valuable insights into the implications of the results, taking into account trade-offs between hash rate, energy efficiency, and profitability, as well as recommending future research directions and opportunities for further optimization. Overall, this paper contributes to our comprehension of the application of machine learning to Bitcoin mining and demonstrates its potential to enhance mining performance.

In the paper, [21] a comprehensive and exhaustive analysis of the efficacy of machine learning techniques for predicting the price of cryptocurrencies is conducted. In order to investigate the potential benefits and limitations of employing diverse machine learning algorithms for accurate cryptocurrency price forecasting, the authors delve into the difficulties associated with this endeavor in the introduction. Subsequently, the methodology section provides extensive information about the dataset used, including its source, size, and relevant characteristics, as well as a detailed description of the various machine learning algorithms used, such as support vector machines, random forests, and neural networks. In addition, the authors discuss the selection of evaluation metrics and the experimental design in detail. The ensuing presentation of experimental results evaluates the precision and performance of the applied machine learning techniques, enabling in-depth analysis. It also provides valuable insights into the implications of the results, including the strengths and limitations of the machine learning models employed for predicting the price of cryptocurrencies. The author also talks about the importance of the blockchain simulator. They also mentioned that some simulators are easily extendable, but some are not, but none of the simulators are working on the scalability feature; instead, they focus on the main part of the blockchain. That's why we need to create our own simulator, which focuses on the scalability feature of blockchain. But some of the simulators, like BlockSim, are pretty standard, and we took inspiration from them. The authors also emphasize on the significance of future research and optimization efforts.

In the paper, [5] the authors address the crucial issue of scalability in blockchain networks, concentrating on the Bitcoin protocol in particular. The paper highlights

the transaction throughput and confirmation time limitations of the original Bitcoin design, which become increasingly problematic as the network expands. The authors propose the Bitcoin-NG protocol as a scalable solution to surmount these obstacles. The Bitcoin-NG protocol incorporates a number of key components and scaling mechanisms. It employs a leader-based strategy in which a leader is designated for a predetermined period of time to process transactions and create microblocks. The subsequent confirmation of these micro-blocks by a group of followers ensures more efficient and parallel processing of transactions. The paper provides a comprehensive explanation of the protocol's leader selection procedure, micro-block creation mechanism, and transaction confirmation procedures. In order to evaluate the performance of the Bitcoin-NG protocol, the authors conduct exhaustive experiments and provide comprehensive performance evaluations. They compare the throughput and latency of the new protocol to those of the original Bitcoin protocol under different network conditions. The results demonstrate Bitcoin-NG's superior scalability, with substantially increased transaction throughput and decreased confirmation times. Overall, the paper contributes to blockchain scalability research by introducing Bitcoin-NG as an innovative solution. It provides a thorough comprehension of the design and mechanisms of the protocol, supported by experimental evaluations. The research paves the way for further investigation and implementation of scalable blockchain protocols, providing potential solutions to the scalability challenges faced by decentralized networks.

There are no good blockchain simulators that can be used generally because of it it's difficult to measure the performance of the model. In the paper, [7] author developed two blockchain simulators which they used to simulate a network of homogeneous miners, and they evaluate how the block size and the end-to-end data transmission delay. They have experimented with different block sizes, and finally, they show that the Bitcoin transaction rate can be increased by increasing the block size. The author also says that if a block size is larger then there will be inconsistency between different copies of the blockchains. Also, if the block size is increased, then the end-to-end transmission delay will increase. So the block size should not be very large instead it should be in the tolerable amount so that transmission delay should not increase. The authors investigate the implications of increasing the block size on the Bitcoin blockchain's dynamics. Recognizing that the Bitcoin network faces scalability challenges, the authors investigate the potential benefits and drawbacks of larger block sizes. Through empirical analysis and simulations, the authors evaluate the impact of increased block sizes on a variety of blockchain metrics. They pay particular attention to crucial factors like block propagation time, orphaned block rates, and blockchain latency. The study provides vital insights into the relationship between block size and network performance by measuring and analyzing these parameters under various block size scenarios. The paper's findings contribute to the ongoing conversation about scalability in blockchain protocols. By evaluating the implications of larger block sizes, the authors cast light on the trade-offs between improved throughput and potential challenges, such as longer block propagation times and an increase in the rate of orphaned blocks. These insights provide a nuanced comprehension of the Bitcoin blockchain's dynamics and offer valuable considerations for future decisions regarding block size adjustments.

In the paper, [17] author discussed applying machine learning to Bitcoin miners and also author did a performance analysis of using it for miners. They used prototype-based experiments for measuring the performance overhead and efficiencies of implementing different Machine learning algorithms. Both supervised and unsupervised measures of performance were utilized. The findings revealed that semi-supervised ML performs worst than supervised algorithms. Interestingly, Logistic regression performs the best, impacting the lowest mining rate.

In the paper, [24] The authors acknowledge the significance of mining efficacy in blockchain networks and propose utilizing machine learning algorithms to optimize the mining procedure. This paper investigates the application of machine learning techniques and methodologies to mining operations. It examines the viability of these techniques for predicting mining outcomes and optimizing mining strategies, taking into account variables such as block generation time, energy consumption, and mining difficulty. The authors assess the efficacy of various machine learning algorithms, including support vector machines, neural networks, and decision trees, for predicting mining variables and optimizing mining decisions. The paper demonstrates the possibility of obtaining an optimal mining strategy by combining blockchain data with machine learning models. Machine learning enables miners to make informed decisions based on historical data, patterns, and predictive models. This strategy seeks to increase mining productivity, maximize mining rewards, and reduce energy consumption. The paper offers insights into the application of machine learning to mining strategies. The findings demonstrate the potential for machine learning techniques to improve mining operations and, ultimately, the efficacy and longevity of blockchain networks. The author uses reinforcement learning and predicts the optimal Bitcoin-like blockchain mining strategy. They formulated the problem as a Markov Decision Process (MDP). They have also designed a multidimensional RL algorithm to solve the problem. In this algorithm, we don't need to know all the parameters of a MDP. They designed a reward that gave the RL agent a sample coin just like in bitcoins. And after applying their multidimensional algorithm, they show that it can find the optimal mining strategy.

The expanding corpus of research discussed illustrates the growing concern regarding the performance of blockchain-based systems. These studies emphasize the need for optimization and efficiency improvements, with a focus on the incorporation of machine learning techniques into blockchain architectures. In addition, the absence of active participation in the development of Bitcoin's core architecture and the lack of focus on reducing block confirmation times are identified as areas for improvement. In addition, researchers acknowledge the acceptability of a small cost associated with the construction of each block. Informed by these insights, our model is innovative in that it eliminates the concept of static block size and implements a mechanism for dynamic block size. By adopting a dynamic block size, our research seeks to investigate the performance ramifications and potential gains in blockchain systems. Through extensive experimentation and analysis, we demonstrate the efficacy and benefits of our dynamic block sizing model, which offers new opportunities for improved blockchain performance, efficiency, and scalability.

Chapter 3

Background Study

3.1 Blockchain components

3.1.1 Block

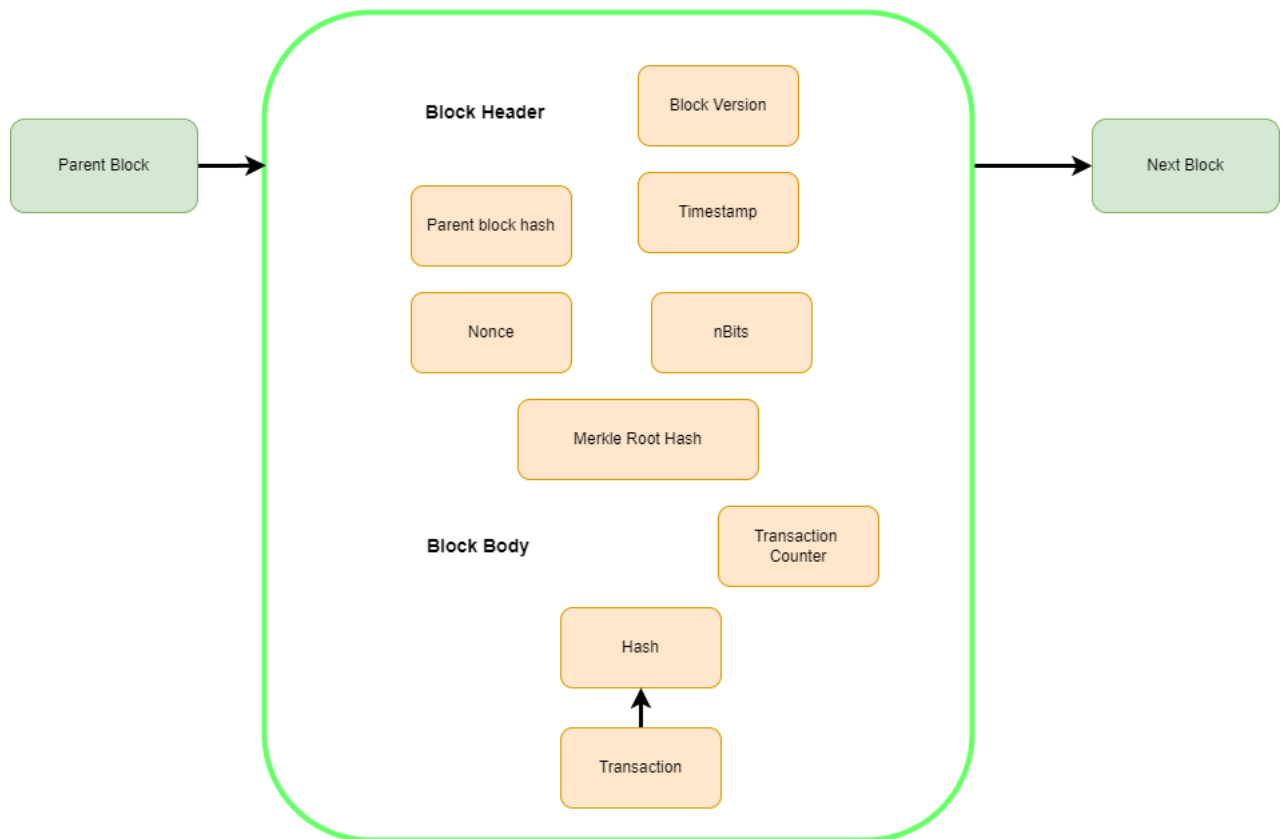


Figure 3.1: Structure of a block in Blockchain

A block is a fundamental blockchain component that contains a set of transactions [6]. It is a data structure that organizes and stores information within the blockchain. Here are the key elements and concepts associated with a block:

Block Header

The block header is a crucial part of a block and contains metadata or information. It typically includes the following components:

- **Version:** The version number of the block structure being used.
- **Previous Block Hash:** The cryptographic hash of the previous block's header. It establishes the link or reference to the previous block in the blockchain, forming the chain.
- **Merkle Root:** A Merkle tree is a data structure that summarizes all the transactions in a block by constructing a hash tree. The Merkle root is the cryptographic hash of all the transaction hashes in the block, forming a single hash representing the entire set of transactions.
- **Timestamp:** The timestamp indicates when the block was created or mined.
- **Nonce:** A nonce is a number used in mining. Miners repeatedly change the nonce value to find a hash that meets specific criteria, such as a certain number of leading zeros. It provides a proof-of-work mechanism to ensure the security and immutability of the blockchain.
- **Difficulty Target:** The difficulty target is a value that determines the difficulty level for miners to find a valid hash. It adjusts dynamically to maintain a consistent block creation rate, typically through difficulty retargeting.

3.1.2 Hash

In blockchain technology, a hash is a fundamental concept to ensure data integrity, security, and immutability. It is a unique digital fingerprint or fixed-length string of characters generated by applying a cryptographic hash function to input data. Let's explore the details of hashes in the blockchain:

Cryptographic Hash Function

A cryptographic hash function is a mathematical algorithm that takes an input (data) and produces a fixed-size output (hash value)[27]. The hash function is designed to be deterministic, meaning that the same input will always produce the same output. It should also be fast to compute the hash but computationally infeasible to reverse-engineer the original input from the hash value.

Unique Identifier

The primary purpose of a hash in the blockchain is to provide a unique identifier for data. A hash function often produces a lengthy string of alphanumeric characters as its output, serving as the input data's digital fingerprint. Any modification to the input data, no matter how little, will result in a dramatically altered hash value.

Data Integrity

Hashes play a crucial role in ensuring data integrity within a blockchain. When a block is created, the hash function is applied to the block's data, including transactions and the block header. The resulting hash value is stored in the block header. The resulting hash will be entirely different if any part of the block's data is modified, even a single character. This property allows for detecting tampering or manipulating data within the blockchain.

Merkle Trees

In many blockchain implementations, including Bitcoin, hashes are organized using a Merkle tree data structure. A Merkle tree is a binary tree structure in which every leaf node represents a transaction hash, and each non-leaf node represents the hash of its child nodes. The topmost node, the Merkle root, represents the hash of all the transactions in a block. Using Merkle trees, it is efficient to verify the presence and integrity of a specific transaction within a block without having to store all the transaction data.

Block Linkage

Hashes are used to establish the linkage between blocks in a blockchain. Each block contains a reference to the previous block's hash within its header. This linkage creates a chronological chain of blocks, with each block's hash effectively linking it to the previous block. Changing any data within a block would alter its hash value, breaking the link to subsequent blocks and making the tampering evident.

Security

Hashes contribute to the security of a blockchain through their cryptographic properties [14]. The irreversible nature of a hash function ensures that it is computationally infeasible to derive the original input data from the hash value. This property is crucial for securing user identities, verifying digital signatures, and protecting sensitive information within a blockchain.

Efficiency

Hashes are computationally efficient to compute, allowing for quick verification and validation processes in a blockchain network. Nodes can easily verify the integrity of a block by recomputing its hash and comparing it with the stored hash in the block header. This efficiency contributes to the scalability and performance of blockchain systems.

By utilizing cryptographic hash functions and maintaining the integrity of hashes, blockchain networks create a transparent, tamper-resistant, and auditable record of transactions and data. Hashes ensure the immutability of previous blocks, prevent tampering, and enable efficient data verification and validation within the blockchain ecosystem.

3.1.3 Chain

A chain refers to the sequential arrangement of blocks in a blockchain. Each block references the previous block's hash, forming a chain-like structure. This ensures the immutability of previous transactions since altering a block would also require changing the subsequent blocks.

3.1.4 Transaction

In blockchain technology, a transaction represents an action or exchange of value recorded on the blockchain[18]. It is a fundamental component of a blockchain system, and here are the key details about transactions:

1. **Data Structure:** A transaction is typically structured as a data object containing various fields that provide information about the transaction. The specific structure may vary depending on the blockchain protocol or platform. Common fields found in transactions include:
 - **Input(s):** Inputs refer to the source of the funds or assets being transferred in the transaction. It typically includes details such as the sender's address or public key and the amount sent.
 - **Output(s):** Outputs represent the recipient(s) of the funds or assets being transferred. Like inputs, outputs contain information such as the recipient's address or public key and the amount received.
 - **Digital Signature:** A transaction includes a digital signature that provides cryptographic proof of its authenticity and ensures that it has not been tampered with. The digital signature is created using the sender's private key and can be verified using the sender's public key.
 - **Transaction ID:** Each transaction is assigned a unique identifier called a transaction ID or hash. The transaction ID is a cryptographic hash of the transaction data and serves as its unique identifier on the blockchain.
2. **Value Transfer:** Transactions in a blockchain primarily involve the transfer of value. The value can be cryptocurrency tokens, digital assets, or other value representations. For example, in Bitcoin, transactions involve the transfer of Bitcoin tokens from one address to another.
3. **Multiple Inputs and Outputs:** A single transaction can have multiple inputs and outputs. This means that a transaction can involve multiple sources of funds and multiple recipients. For instance, a transaction may aggregate funds from different addresses to be sent to multiple recipients in a single transaction.
4. **Transaction Validation:** Before a transaction is considered valid and included in a block, it needs to be validated by the blockchain network. The validation process varies depending on the consensus mechanism employed by the blockchain. Validators or miners verify that the transaction adheres to specific rules, such as the availability of sufficient funds and the correct digital signatures.

5. **Transaction Fees:** Transactions in some blockchains may require the payment of transaction fees. These fees incentivise miners or validators to include the transaction in a block and prioritize its processing. The fees help prevent spam transactions and contribute to the security and sustainability of the blockchain network.
6. **Transaction Confirmation:** Once a transaction is validated, it enters a confirmation state. Confirmation refers to including the transaction in a block, which is then added to the blockchain. The number of confirmations a transaction receives indicates the number of blocks added to the block containing the transaction. A higher number of confirmations increases the security and finality of the transaction.
7. **Transaction History and Transparency:** On the blockchain, transactions are a chronological account of all activities. All network users can see every transaction, ensuring accountability and openness. Anyone may check the movement of money and follow the history of transactions within the blockchain, thanks to the public nature of transactions.

Blockchain technology relies heavily on transactions since they make it possible to transfer money and carry out smart contracts. They guarantee data security and immutability by providing a transparent and auditable record of all transactions made on the blockchain.

3.1.5 Consensus Mechanism

An essential element of blockchain technology that ensures agreement and validation of transactions across a decentralized network is the consensus mechanism[19]. In Fig 3.2, we see that we have multiple nodes; a block will only be added to the blockchain if 50% of the miners confirm it. It makes it possible for nodes in the network to agree on the legitimacy and order of transactions as well as the current state of the blockchain. The specifics of consensus processes are as follows:

1. **Problem of Consensus:** Achieving consensus becomes crucial in a decentralized network because various nodes validate and add transactions to the blockchain. The goal is to ensure that all trustworthy nodes agree on a single, consistent version of the blockchain and to stop bad actors from interfering with the system.
2. **Byzantine Fault Tolerance:** Consensus mechanisms aim to achieve Byzantine fault tolerance, meaning that the network can tolerate faulty or malicious nodes without compromising the integrity and security of the blockchain. Byzantine faults refer to arbitrary and potentially malicious behavior by nodes, such as sending conflicting information or attempting to tamper with transactions.
3. **Different Consensus Mechanisms:** Several consensus mechanisms are used in blockchain networks, each with its own rules and algorithms. Some commonly used consensus mechanisms include:

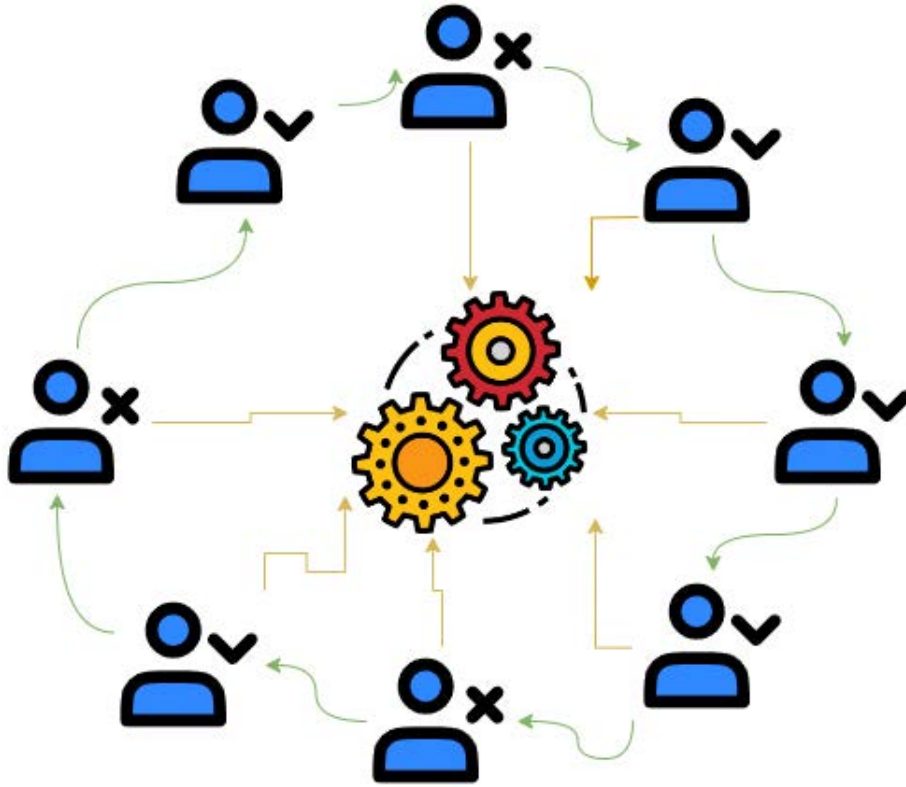


Figure 3.2: Consensus Mechanism

- **Proof of Work (PoW):** PoW [20] is the consensus mechanism introduced by Bitcoin. Miners compete to solve complex mathematical puzzles using computational power. The first miner to find a solution broadcasts it to the network, and other nodes verify it. Once a solution is accepted, the miner can add a new block to the blockchain and receive a reward.
 - **Proof of Stake (PoS):** In PoS [23], validators are chosen to create new blocks based on the amount of cryptocurrency they hold and "stake" in the network. Validators are selected through a deterministic process that takes into account their stake.
 - **Delegated Proof of Stake (DPoS):** DPoS [22] is an extension of PoS where token holders vote for a limited number of "delegates" responsible for validating transactions and producing blocks. Delegates take turns producing blocks; token holders can vote to replace delegates if they misbehave.
 - **Proof of Authority (PoA):** In PoA[29], a limited number of trusted nodes or validators are authorized to create new blocks. Validators are typically known entities, such as reputable organizations or individuals.
 - **Practical Byzantine Fault Tolerance (PBFT):** PBFT [11] is a consensus mechanism for permissioned blockchains. It requires a predetermined set of validators known as replicas. Replicas communicate to agree on the order of transactions and the state of the blockchain.
4. **Consensus Process:** The consensus process involves steps that allow nodes to agree on the validity and order of transactions. These steps typically include

a proposal, validation, agreement, and block addition.

5. **Trade-offs:** Different consensus mechanisms offer various trade-offs regarding security, scalability, decentralization, energy consumption, and throughput. The choice of consensus mechanism depends on the blockchain network's specific use case and requirements.

Consensus mechanisms play a critical role in ensuring the trustworthiness and integrity of blockchain networks. By establishing agreements among decentralized participants, consensus mechanisms enable secure and transparent transactions without a centralized authority.

3.1.6 Peer-to-Peer Network

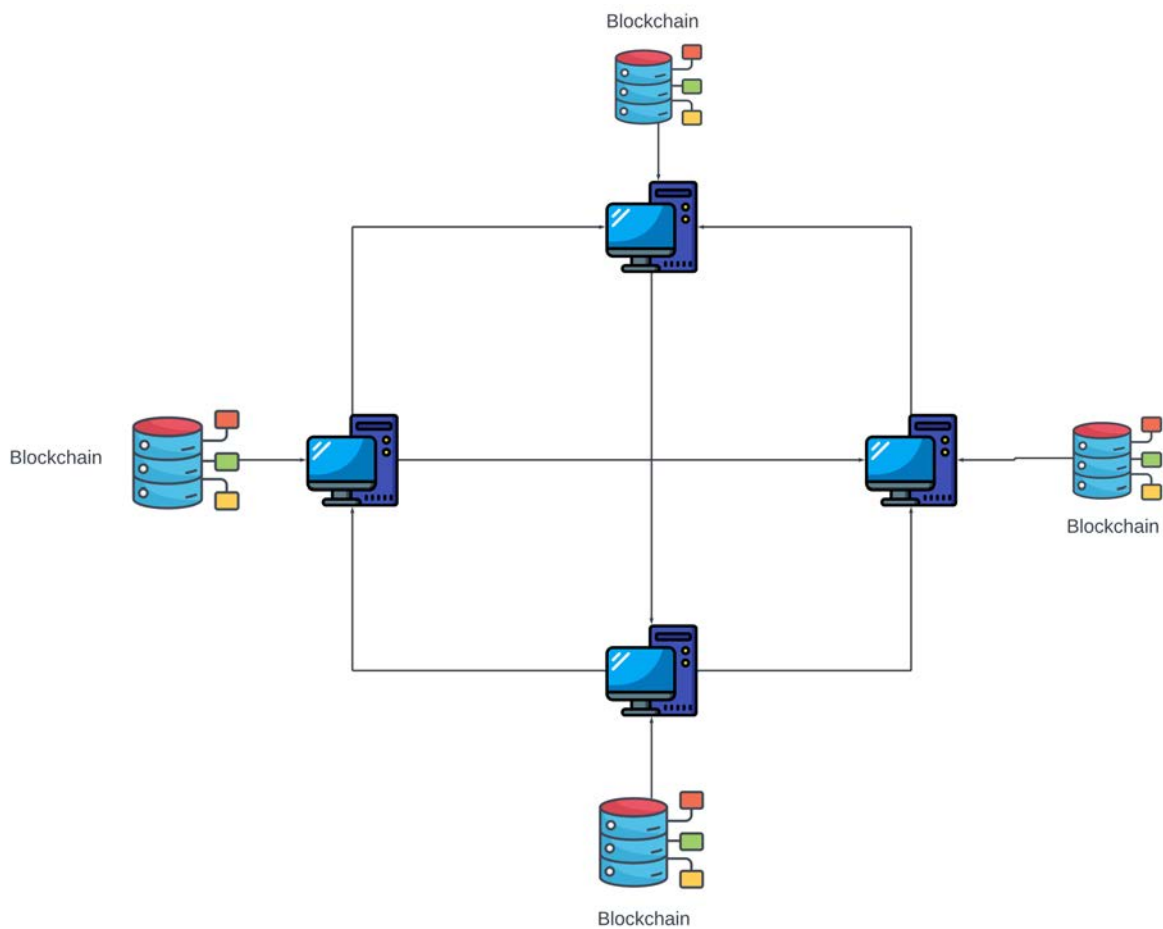


Figure 3.3: P2P network diagram

In the context of blockchain, a peer-to-peer (P2P) network refers to the decentralized network architecture used by blockchain systems[25]. Blockchain technology leverages the principles of P2P networks to enable secure and transparent transactions without a centralized authority. Let's discuss peer-to-peer networks in the context of blockchain:

1. **Decentralization:** Blockchain networks operate decentralised, where multiple nodes validate and store transactions. Each node in the network maintains

a copy of the entire blockchain ledger. This decentralization ensures that no single entity controls the entire network, making it resistant to censorship and single points of failure.

2. **Peer Nodes:** Every participating node in a blockchain's P2P network functions as an equal peer, able to start transactions, verify blocks, and spread information. Each node is distinguished from other nodes by a unique cryptographic identifier (such as a public key).
3. **Network Communication:** A particular protocol is used by nodes in a blockchain P2P network to connect directly. They exchange transactions, blocks, and other pertinent data through peer-to-peer connections. Data propagation is efficient and safe thanks to this direct connectivity, eliminating the need for a central server or middleman.
4. **Consensus Mechanism:** In a blockchain system, attaining consensus requires P2P networks. Nodes can agree on the legitimacy and chronological order of transactions thanks to consensus algorithms like Proof of Work (PoW) and Proof of Stake (PoS). Nodes collaborate to obtain consensus and preserve the integrity of the blockchain using peer-to-peer communication and consensus mechanisms.
5. **Synchronization and Blockchain Validation:** Nodes work together in a P2P blockchain network to synchronize and verify the state of the blockchain. New network nodes can get one by asking their peers for a copy of the blockchain. Nodes guarantee the accuracy and consistency of the blockchain across the network by validating transactions and blocks.
6. **Data Distribution and Replication:** P2P networks in blockchain enable the distribution and replication of data across multiple nodes. Each node stores a copy of the entire blockchain, ensuring redundancy and fault tolerance. This distributed storage mechanism enhances the security and resilience of the blockchain network.
7. **Network Scalability:** P2P networks in blockchain are designed to scale horizontally as more nodes join the network[15]. With increasing participation, the network can handle higher transaction volumes and maintain decentralization. The collaborative nature of P2P networks allows for efficient resource utilization and scalability.
8. **Network Security:** P2P networks in blockchain provide inherent security benefits. Due to their decentralized nature, they resist various attacks that target central points of control. Additionally, cryptographic techniques are used to secure transactions and ensure the authenticity and integrity of the blockchain data.
9. **Incentive Mechanisms:** P2P networks in blockchain often incorporate incentive mechanisms to motivate node participation and cooperation. For example, in PoW-based blockchains like Bitcoin, miners are rewarded with cryptocurrency for their computational efforts in securing the network.

By leveraging P2P network architecture, blockchain technology enables the creation of decentralized, transparent, and tamper-resistant systems. The collaborative nature of peer-to-peer networks enhances security, scalability, and resilience, making them a foundational element in blockchain implementations.

3.1.7 Distributed Ledger

In the context of blockchain, a distributed ledger[12] refers to a decentralized and immutable record of transactions or data that is shared and synchronized across multiple nodes or participants in a network. It is a fundamental concept in blockchain technology and crucial to ensuring transparency, security, and consensus.

Distributed ledgers operate decentralised, where multiple nodes in a network maintain a copy of the ledger. Each participant has equal access and rights to validate and update the ledger. This decentralized nature eliminates the need for a central authority and enhances the trust and resilience of the system.

The data in a distributed ledger is consistent across all participants. Every transaction or data entry is recorded across multiple nodes, and consensus mechanisms are used to agree on the validity and order of transactions. This consensus ensures that all participants have the same view of the ledger and helps prevent fraud or manipulation.

One of the key features of distributed ledgers is immutability[16] and append-only nature. Once a transaction or data entry is added to the ledger, it becomes virtually impossible to alter or delete it. The records in the ledger are typically stored using cryptographic hashing, creating a unique identifier for each data block. Any block changes would require the network consensus, making the ledger highly secure and resistant to tampering.

Transparency and audibility are inherent in distributed ledgers. The ledger provides transparency, as all participants can access the entire transaction history. This transparency enhances accountability and enables the auditing of transactions. Anyone with access to the ledger can independently verify the integrity and validity of transactions, promoting trust and reducing reliance on intermediaries.

Trust and security are ensured in distributed ledgers through cryptographic techniques. Transactions are digitally signed, and consensus mechanisms validate the authenticity of transactions. The decentralized nature of the ledger reduces the risk of single points of failure and enhances security against malicious attacks.

Distributed ledgers can scale horizontally by adding more nodes to the network. As the number of participants increases, the distributed network can handle higher transaction volumes without compromising performance. This scalability is crucial for blockchain applications that require processing many transactions.

Distributed ledgers have a wide range of applications beyond cryptocurrencies. They can be used for supply chain management, asset tokenization, decentralized identity, smart contracts, etc. The distributed ledger ensures transparency, security, and efficiency in recording and validating transactions in these applications.

Interoperability is another important aspect of distributed ledgers. They can be designed to interoperate with other distributed ledgers or traditional systems. Interoperability allows seamless integration and data exchange between networks, enabling more complex and interconnected blockchain ecosystems.

In summary, as implemented in blockchain technology, distributed ledgers provide

a decentralized, transparent, and secure way to record and manage transactions or data. They eliminate the need for intermediaries, enhance trust, and offer new possibilities for innovation in various industries.

3.2 Blockchain Architecture

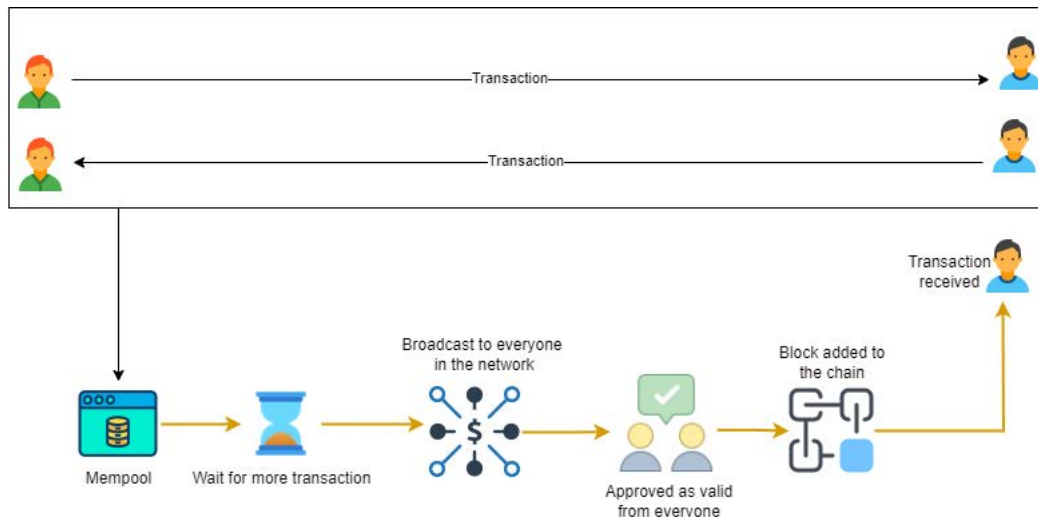


Figure 3.4: Current bitcoin blockchain Architecture

For example, I am "A," and I want to send a bitcoin to "B" to send the bitcoin; specific steps happen in between. It's not instantaneous, and that's called a waiting time when the transaction is created and before it's committed in the blockchain.

Wallet Setup: To send Bitcoin, the initial step is to select a Bitcoin wallet provider[3] or software and adhere to their instructions for creating a new wallet. Once the wallet is set up, it generates a new Bitcoin address, also known as a public key, which serves as the destination for receiving the Bitcoin.

Creating a Transaction: To create a Bitcoin transaction, you must access your Bitcoin wallet and follow a few steps. Firstly, locate the option to initiate a new transaction within your wallet interface. Then, enter the recipient's Bitcoin address (B), ensuring that it is accurate to avoid unintended transfers. Next, specify the amount of Bitcoin you wish to send to the recipient. Some wallets may provide predefined options to choose from to simplify this process. Afterwards, you must select the transaction fee you are willing to pay. It's important to note that higher fees can lead to faster confirmation times for your transaction on the Bitcoin network. Take a moment to review all the transaction details diligently. Double-check the recipient's Bitcoin address, the specified amount, and the chosen transaction fee to prevent errors or mishaps. Once you have verified everything, proceed to confirm the transaction. By doing so, you initiate the transfer of Bitcoin from your wallet to the recipient's address. To maintain the correctness and security of your Bitcoin transactions, it is essential to proceed with caution and attention.

Transaction Signing: Transaction signing in the context of blockchain involves using cryptographic techniques to verify and authenticate transactions on the blockchain network. Users create a digital signature using their private key when they initiate a transaction.

The transaction data is hashed to create a distinct digital fingerprint of the transaction to start the process. The information about the transaction is compressed into this hash. This hash is then encrypted with the private key to produce a digital signature unique to that transaction.

The encrypted hash and any additional metadata are both included in the digital signature. It demonstrates ownership and guarantees the fairness of the exchange. The transaction data is broadcast to the blockchain network with the signature attached.

After receiving the transaction, network users can confirm its legitimacy by decrypting the signature with the appropriate public key. The signature can be verified using the public key, which is obtained from the private key and prevents the private key from being made public.

The transaction is regarded as genuine and valid once the digital signature has been successfully encrypted and validated. Miners can then incorporate the transaction into a block added to the network.

Only the private key owner can approve transactions linked to that key, according to the transaction signing protocol. It offers a system for security and confidence in the blockchain network, ensuring that transactions are authentic and impervious to manipulation.

Broadcasting the Transaction: A transaction is transmitted across the network after a user starts one so that it reaches the nodes and miners. All participants are informed of the transaction and are able to confirm its validity thanks to broadcasting. Typically, the transaction is sent from node to node until it is received by a miner, who adds it to a block for verification and adding to the blockchain. The blockchain network provides transparency by broadcasting the transaction because all users have access to the transaction data, facilitating consensus, validation, and the general security of the blockchain ecosystem.

Transaction Verification: When a blockchain network node receives a transaction, transaction verification takes place when the nodes go through several tests to confirm the transaction's authenticity. These checks entail comparing the digital signature to the transaction data and the sender's private key using the public key connected to the transaction as evidence. To avoid overspending or double spending, the nodes also confirm that the sender has enough money in their wallet to cover the transaction amount. The nodes additionally check the transaction format to ensure that it adheres to the guidelines and regulations established by the Bitcoin network. The blockchain network maintains the integrity, security, and consensus required to authenticate and approve legal transactions through these thorough verification procedures.

Mining: Mining is a fundamental process in blockchain networks where specialized nodes called miners compete to solve a cryptographic puzzle known as Proof of Work (PoW) to create new blocks. These miners collect a set of valid transactions, including yours, from the mempool and embark on finding a solution to the puzzle. The solution requires substantial computational power, and miners employ specialized hardware to perform countless calculations to discover a valid solution. The miner who successfully finds the solution first broadcasts it to the network, verifying the transactions in the block and adding it to the blockchain. By rewarding miners for their computing labour and contribution to the consensus mechanism, the mining process protects the blockchain network's security, immutability, and

decentralization.

Block Formation: A miner who has solved the cryptographic riddle successfully creates a new block containing a series of transactions, including the one you started. To preserve the blockchain’s chronological order and continuity, the miner creates a chain of connected blocks by referencing the previous block within the freshly formed block. This chain, also called the blockchain, acts as a secure and unchangeable transactional ledger. Once the block has been created, the miner broadcasts it to the whole network, ensuring everyone knows the most recent blockchain update. By facilitating consensus among the nodes and enabling the confirmation and verification of transactions, the new block’s distribution further reinforces the integrity and openness of the blockchain network.

Block Confirmation: Other nodes in the network receive the freshly mined block once it has been published and start a series of checks to verify its legitimacy. These checks validate the transactions in the block, confirming their accuracy and adherence to the established rules of the blockchain network. Additionally, nodes verify the Proof of Work solution provided by the miner, ensuring that the computational puzzle was correctly solved. If the block successfully passes all these checks, it is deemed valid, and participating nodes add it to their local copies of the blockchain, establishing consensus across the network. At this pivotal moment, your transaction is officially confirmed, as it is now a permanent and immutable part of the blockchain, providing transparency, security, and trust in the transaction history of the blockchain network.

Multiple Confirmations: For enhanced security and assurance, it is recommended to wait for multiple confirmations before considering a transaction fully settled with each subsequent block added to the blockchain after the block containing your transaction, the number of confirmations increases. The exact number of confirmations required may vary depending on the nature and sensitivity of the transaction and the specific policies of the recipient or service provider. Waiting for multiple confirmations provides a higher level of confidence that the transaction is valid and irreversible, as it has been verified and included in multiple blocks within the blockchain. This practice helps mitigate the risks associated with potential blockchain reorganizations or forks, ensuring a more reliable and secure confirmation of your transaction.

Balance Update: After your transaction is successfully confirmed and added to the blockchain, the balances associated with addresses A (the sender) and B (the recipient) are updated accordingly. The amount sent in the transaction reduces the balance of address A, reflecting the outgoing funds. Conversely, the balance of address B is increased by the exact amount received, indicating the arrival of the transferred funds. These balance updates are recorded on the blockchain, accurately representing the current holdings for each address involved in the transaction. This process ensures transparency and accountability, allowing users to track and manage their Bitcoin balances accurately and confidently.

OpenAI Gym[4] is a software toolkit designed to facilitate the development and comparison of reinforcement learning algorithms. It provides access to a standardized set of environments enabling interaction between an *agent* and its surrounding *environment*. The agent can take specific actions within the environment, and in return, it receives observations and rewards based on its actions.

During each step taken by the agent, the environment provides four values:

Observation (`object`): This represents the agent’s perception or state of the environment. It could be a game board state, sensor readings, or any other relevant information specific to the environment. The observation serves as the input to the agent’s decision-making process, allowing it to assess the current state and make informed choices for the next action.

Reward (`float`): The `reward`[4] indicates the amount of success or score obtained as a result of the agent’s previous action. The reward scale may vary across different environments, but maximising the total reward or score is the ultimate objective. Positive rewards typically indicate progress or desirable outcomes, while negative rewards represent penalties or setbacks. By receiving rewards, the agent can learn from the consequences of its actions and adjust its behavior accordingly.

Done (`boolean`): This `flag`[4] indicates whether it is necessary to reset the environment. For example, a game scenario could signify the loss of the agent’s last life or the completion of a task. When the `done` flag is `True`, the current episode or task has reached its termination point. At this stage, the agent may need to reset the environment to start a new episode and continue its learning process.

Info (`dict`): The `info` dictionary[4] contains additional diagnostic information that can be useful for debugging purposes. It may provide insights into the internal state of the environment, such as intermediate metrics, debug logs or statistics. However, it is essential to note that official evaluations of the agent should not utilize this information for learning purposes to ensure fair and unbiased assessments.

In summary, OpenAI Gym provides a flexible framework where an agent interacts with an environment, receiving observations and rewards and using them to learn and improve decision-making capabilities. By exploring the environment, taking action, and receiving feedback in the form of rewards, the agent aims to discover strategies that maximize its long-term cumulative reward and achieve optimal performance in the given task or environment. The standardized nature of OpenAI Gym environments allows researchers and developers to compare different algorithms. It approaches on a level playing field, fostering collaboration and advancement in reinforcement learning.

Chapter 4

Methodologies

4.1 Proposed Scheme

Fig. 4.1 shows the full framework of the proposed scheme named ROBB. We have used RPPO in the model and defined a blockchain environment where we are training the model. We also show the state, action and reward function of the model and how it's integrated with the whole model. This study consists of 2 networks one is called the policy network and another is called the value network. The policy network is responsible for representing and learning the policy in RPPO. It is implemented as a deep recurrent neural network that inputs the environment state and outputs the parameters of a probability distribution over actions. The main goal of this network is to learn a policy that maximizes the expected cumulative reward over time. During the training process, the policy network interacts with the environment and generates actions based on its current policy. These actions are then executed in the environment, and the resulting states, rewards, and other relevant information are observed. Our system optimizes the policy by iteratively collecting data from the environment, computing advantages using value function estimates, and then performing multiple iterations of policy optimization steps. In each iteration, the policy-network is trained using a surrogate objective function. Overall, the policy network in RPPO represents the policy, generates actions based on it, and updates its parameters to improve its performance over time. Another network defined in the system is called the value network. The value network plays a crucial role in estimating the value or advantage function. It estimates the expected cumulative reward or advantage of being in a particular state and following the current policy. The value network is also a deep recurrent neural network that takes the environment state as input and outputs a value estimate or advantage estimate. It also estimates the state-value function, which represents the expected cumulative reward starting from a particular state and following the current policy. It also estimates the advantage function, which represents the advantage of taking a particular action in a given state compared to the average value of all actions in that state. The advantage estimate helps determine which actions are more favourable than others and guides the policy optimization process. During training, the value network is updated by comparing its estimates with the actual observed rewards or advantages. The difference between the estimated value or advantage and the observed reward or advantage is used to compute a loss, which is then used to update the parameters of the value network via backpropagation. The value network

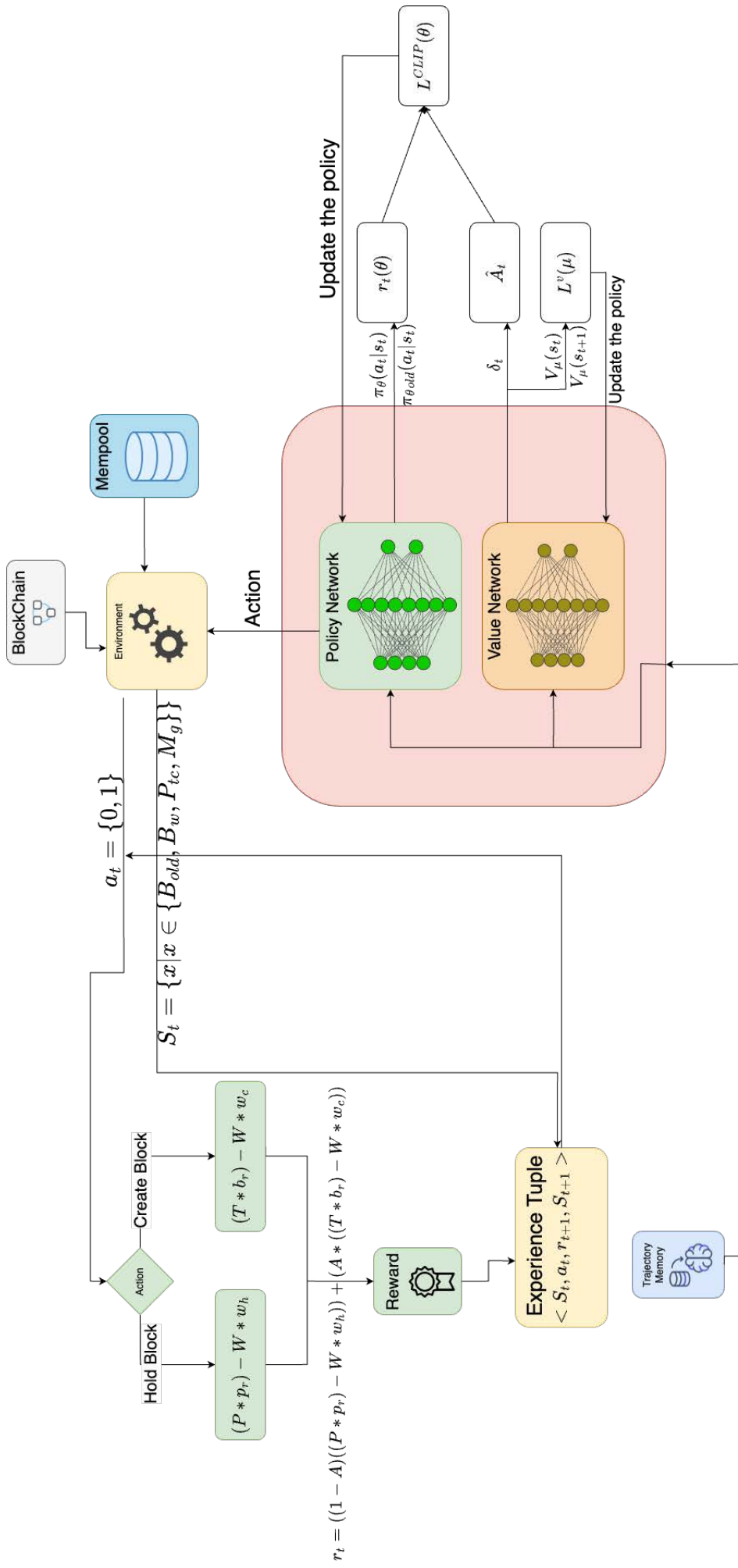


Figure 4.1: ROBB system workflow

is responsible for estimating the value or advantage function, providing feedback to the policy network regarding the quality of different states and actions, and assisting in the policy optimization process to maximize the cumulative reward or advantage.

In Algorithm 1, we have described our workflow as an algorithm where we can individually see each step. At first, we need to initialize the parameter of our policy and value network. Then we can Initialize our custom environment. After that, we need to get the experience tuple to simulate the environment multiple times and get the state, action, reward, next state, and next reward. After creating the tuple, we can use this data to train our model. As previously mentioned, we have two neural networks, so we need to update the weights of the neural networks using Equation 4.8 and Equation 4.9.

Algorithm 1 Algorithm for ROBB

- 1: Initialize policy network π_θ with parameters θ_0
 - 2: Initialize value function network V_ϕ with parameters ϕ_0
 - 3: Initialize environment E
 - 4: State : $S_t = \{x|x \in \{B_{old}, B_w, P_{tc}, M_g\}\}$
 - 5: Action: $a_t = \{0, 1\}$
 - 6: Reward: $r_t = ((1 - A)((P * p_r) - W * w_h)) + (A * ((T * b_r) - W * w_c))$
 - 7: **for** $k = 0, 1, 2, \dots$ **do**
 - 8: Collect set of trajectories $D_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment
 - 9: Compute rewards-to-go \widehat{R}_t .
 - 10: Compute advantage estimates, \widehat{A}_t (using any method of advantage estimation) based on the current value function V_{ϕ_k} .
 - 11: Update the policy by maximizing the RPPO-clip objective:

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^T \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\in A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$
 Use Adam optimizer to update the gradient.
 - 12: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \underset{\phi}{\operatorname{argmin}} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^T (V_\phi(s_t) - \widehat{R}_t)^2$$
 use gradient descent algorithm to update the weights
 - 13: **end for**
-

4.1.1 Policy and value network as Predictive Model

As mentioned in the previous section, we have used two different neural networks. In this section, we are going to discuss how we are going to update the policy. Using online policy updating means updating the policy in real-time as the agent interacts with the environment. The policy network’s policy update is performed using a surrogate objective function. The policy update aims to improve the policy by adjusting the parameters of the policy network. It is designed to strike a balance between policy improvement and policy divergence. It encourages the policy to move towards actions that have higher advantages while ensuring that the policy changes are not too large. This helps to maintain stability during the policy update process.

$$L^{PG}(\theta) = E_t [\log \pi_\theta(a_t|s_t) * A_t] \quad (4.1)$$

The idea was that by taking a gradient ascent step on this function (equivalent to taking gradient descent of the negative of this function), we would push our agent to take actions that lead to higher rewards and avoid harmful actions. But there are two problems in this function if the step size is too small, the training time will increase. On the other hand, if it's too high, there is too much variability in the training. As we are using RPPO, In the original PPO paper, they have modified the surrogate objective function, designed to avoid destructively large weights update.

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathbb{E}_{\tau \sim \pi_k} \left[\sum_{t=0}^T \left[\min \left(r_t(\theta) \hat{A}_t^{\pi_k}, \text{clip} \left(r_t(\theta), 1 - \epsilon, 1 + \epsilon \right) \hat{A}_t^{\pi_k} \right) \right] \right] \quad (4.2)$$

Another important thing we need to know is Ratio Function. The ratio function measures the difference between the new and old policies and controls the amount of policy updates during optimization. The ratio function is denoted by $r_t(\theta)$ It compares the probability of taking an action under the new policy (π_{new}) to the probability of taking the same action under the old policy (π_{old}), both given the current state. If $r_t(\theta) > 1$, then the action a_t , at state s_t , is more likely in the current policy than the old policy. But if $r_t(\theta)$ is between 0 and 1, the action is less likely for the current policy than for the old one. So this probability ratio is an easy way to estimate the divergence between old and current policy.

$$r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \quad (4.3)$$

Finally, the value model, to update the policy of the value network, involves two things computing advantages and surrogate loss, while the value network update focuses on minimizing the mean squared error between the estimated values and the actual discounted returns. One of the main things to do before updating the policy is to calculate the loss function. To calculate the loss of a value network, we can use mean squared error (MSE) between the predicted value from the value network $V(s_t)$ and the target value for each state in the collected trajectories. If we assume the predicted value as $(V_\theta(s_t))$, then the loss function L will be.

$$L = \frac{1}{|T|} \sum_t (V(s_t) - V_\theta(s_t))^2 \quad (4.4)$$

We can use this 4.5 formula to calculate the $V(s_t)$. Here, r_t represents the reward obtained after taking action a_t in state s_t , γ (gamma) is the discount factor, and $V(s_{t+1})$ represents the target value for the next state s_{t+1} .

$$V(s_t) = r_t + \gamma \cdot V(s_{t+1}) \quad (4.5)$$

4.1.2 RPPO based reinforcement learning agent design

There are existing deep reinforcement learning methods like DQN, PPO, A2C, and DDPG, which are commonly used in the fields of reinforcement learning. And the best thing is that these models do not require mathematical modeling but require thousands of interactions with the environment for training. In our case, RPPO

outperforms all other models we have tried with PPO, and it did not go well as it cannot converge and cannot create blocks in a timely manner. So to test the models, we first have to create an environment compatible with most of the algorithms available for reinforcement learning. But to create a reinforcement learning environment, there are three important variables that we need to define those are observation, action, and reward.

Observation: To create an observation we need to provide the model the full picture to decide whether to create a block or not. So to provide the model the full picture we have provided block size of the previous block, the average waiting time of the last block, pending transactions count, mempool growth, and current transaction block size as an observation.

- **Block size of the previous block (B_{old}):** Including the size of the previous block as an observation provides the model with insights into the recent block's capacity. By considering the previous block's size, the model can make decisions about the size and contents of the current block. For example, if the previous block was small, the model might prioritize including more transactions in the current block to maximize efficiency.
- **Average waiting time of the last block (B_w):** The average waiting time of the previous block is a valuable observation for the model. It reflects the time taken for transactions in the mempool to be included in a block. Considering this information, the model can learn to optimize block creation strategies to minimize waiting times and improve transaction throughput. For instance, if the average waiting time was relatively high in the previous block, the model might prioritize including time-sensitive transactions in the current block.
- **Pending transaction count (P_{tc}):** Observing the number of pending transactions in the mempool gives the model a sense of the current backlog. It enables the model to balance the need for prompt inclusion of transactions with avoiding frequent block creation. The model can learn to find an optimal threshold for pending transaction count, ensuring efficient block utilization while maintaining an acceptable backlog level.
- **Mempool Growth (M_g):** Including the current size of the mempool as an observation allows the model to monitor the growth rate of pending transactions. This information helps the model decide based on the transaction arrival rate and network conditions. If the mempool size rapidly increases, the model might create a block sooner to prevent congestion and alleviate transaction delays.
- **Current Transaction block size:** Observing the current transaction block size provides the model with real-time information about the accumulated transactions. This information allows the model to assess whether the current block size has reached a predefined threshold or if additional transactions should be included before creating a block. The model can learn to make decisions that optimize block size and maximize resource utilization while considering transaction volume.

So our final state S_t will be Equation 4.6.

$$S_t = \{x|x \in \{B_{old}, B_w, P_{tc}, M_g\}\} \quad (4.6)$$

By incorporating these observations, the model gains a holistic view of the blockchain’s state and can adapt its decision-making based on block capacity, waiting times, pending transactions, mempool growth, and current transaction block size. This comprehensive understanding enables the model to learn strategies that improve efficiency, transaction throughput, and overall performance in the simulated blockchain environment.

Action: The action component plays a crucial role in designing a reinforcement learning environment. In our simulation, we have carefully crafted the actions to mimic real-world blockchain behavior while ensuring efficient block utilization closely.

By considering the real-world environment, we aim to create a simulation that reflects the dynamics of blockchain systems. Our focus is maximizing block utilization and avoiding wastage of space within each block. This means the model learns to optimize the selection of transactions to include in a block, ensuring that valuable space is utilized effectively.

By emphasizing efficient block utilization in our simulation, we enable the model to learn behaviours resembling real-world blockchain systems. This ensures the model’s decision-making aligns to optimize transaction throughput and resource utilization in the simulated environment. In our simulation, we have designed a discrete action space, allowing our model to choose between two values: 0 and 1.

- **Action (A) 0:** The "hold" strategy: By choosing action 0, the model decides to hold off on creating a block and waits for more transactions to accumulate in the mempool. This action allows the model to be patient and maximize block space utilisation. By waiting for additional transactions, the model can potentially include a larger number of transactions in each block, thus optimizing the overall efficiency of block utilization.
- **Action (A) 1:** Selecting action 1 signifies creating a block using the transactions in the mempool. Unlike a fixed block size approach, our simulation supports dynamic block sizing. This means that the size of the block is determined by the number and size of transactions in the mempool at the time of block creation. By dynamically sizing the block, we avoid wasting any excess space, ensuring the block is filled to its maximum capacity.

So the action at a particular timestamp will be Equation 4.7

$$a_t = \{0, 1\} \quad (4.7)$$

Utilizing a discrete action space with these two actions gives the model a flexible block creation and resource management approach. The model can learn to balance waiting for more transactions to maximize block efficiency (action 0) and efficiently utilise available space by creating dynamic-sized blocks (action 1).

This design choice reflects the real-world behaviour of blockchain systems, where block sizes can vary based on transaction volume and network conditions. By incorporating these actions into our simulation, we enable the model to adapt its

decision-making process and optimize block utilization, ultimately improving the efficiency and effectiveness of the simulated blockchain environment.

Reward: The reward function is critical in designing a reinforcement learning model, as it heavily influences behaviour. Designing a practical reward function for blockchain simulations can be challenging, requiring extensive experimentation with different parameters and types of reward functions. In our approach, we have invested significant time and effort to refine the reward function. We have divided it into two functions to maximize efficiency: one for action 0 and another for action 1. Our reward function incorporates multiple components, including the number of pending transactions in the mempool, waiting time for the current transaction, the average waiting time during block creation, and the length of transactions in the block. We aim to incentivize efficient transaction processing, minimize waiting times, optimize block creation, and enhance block space utilization by carefully designing and iterating on the reward function.

There are a few constants that we need to know before designing our reward function:

- **pending transaction (P):** The size of pending transactions in the mempool provides crucial information about the state of the network. It indicates the number and size of transactions awaiting inclusion in a block. By considering this constant, our model gains insights into the congestion level and workload of the network. It can learn to prioritize transaction processing based on transaction size. Optimizing the handling of pending transactions can lead to a reduced backlog, faster transaction confirmations, and improved overall network efficiency
- **waiting time (W):** The average waiting time of transactions in the mempool is a vital metric for assessing transaction congestion and the urgency of inclusion. Transactions with longer waiting times are typically more time-sensitive and require prioritized processing to minimize delays. By incorporating the waiting time constant into the reward function, our model can learn to prioritize transactions based on their waiting time. This encourages the timely processing of transactions and reduces transaction latency, leading to improved user experience and increased network throughput.
- **Transaction included in the block (T):** The metric of transactions included in the block is valuable when the model decides to create a block. It allows us to reward the model based on its chosen block size appropriately. By utilizing this metric, we can provide positive rewards when the model creates a sufficiently large block, incentivizing the inclusion of a higher number of transactions. Conversely, we can decrease the reward if the model decides to go with smaller block sizes, encouraging it to aim for more optimal block utilization and avoid unnecessary space. This aspect of the reward function guides the model toward maximizing block capacity and transaction throughput, promoting efficient resource allocation within the simulated blockchain environment.
- **Pending transaction utilization rate (p_r):** This constant represents the utilization rate of pending transactions when the model decides to hold a block. It indicates the efficiency with which the model utilizes the available

pending transactions in the mempool during the "hold" action. Considering this utilization rate, we can reward the model for effectively managing pending transactions and minimizing their accumulation over time.

- **Waiting time utilization rate for holding (w_h):** This constant denotes the average waiting time utilization rate when the model chooses to hold a block. It reflects how efficiently the model should use the waiting time of transactions in the mempool during the "hold" action. Incorporating this utilization rate into the reward function allows us to incentivize the model to reduce waiting times and prioritize timely transaction processing.
- **block size utilization rate (b_r):** This constant represents the utilization rate of block size when the model decides to create a block. It indicates how efficiently the model utilizes the available block space during the "create block" action. Considering this utilization rate, we can reward the model for creating blocks that utilize a significant portion of the available block space, thus optimizing the block size and improving overall resource utilization.
- **Waiting time utilization rate for creating a block (w_c):** This constant reflects the average waiting time utilization rate when the model chooses to create a block. It captures how efficiently the model incorporates waiting times of transactions into the decision-making process during the "create block" action.

Reward for action 0:

$$r_{th} = (P * p_r) - W * w_h \quad (4.8)$$

Reward for action 1:

$$r_{tc} = (T * b_r) - W * w_c \quad (4.9)$$

Combine reward function will be:

$$r_t = ((1 - A)((P * p_r) - W * w_h)) + (A * ((T * b_r) - W * w_c)) \quad (4.10)$$

Table 4.1: Constants values for reward function

Parameter	Symbol	Value
Pending transaction utilization rate	p_r	0.9
Waiting time utilization rate for holding	w_h	0.15
Block size utilization rate	b_r	0.7
Waiting time utilization rate for creating block	w_c	0.2

Chapter 5

Result analysis

5.1 Blockchain Simulator for ROBB

In this section, first, we will explain how we create a blockchain simulator and how it can be generalized to measure the performance of multiple machine learning models.

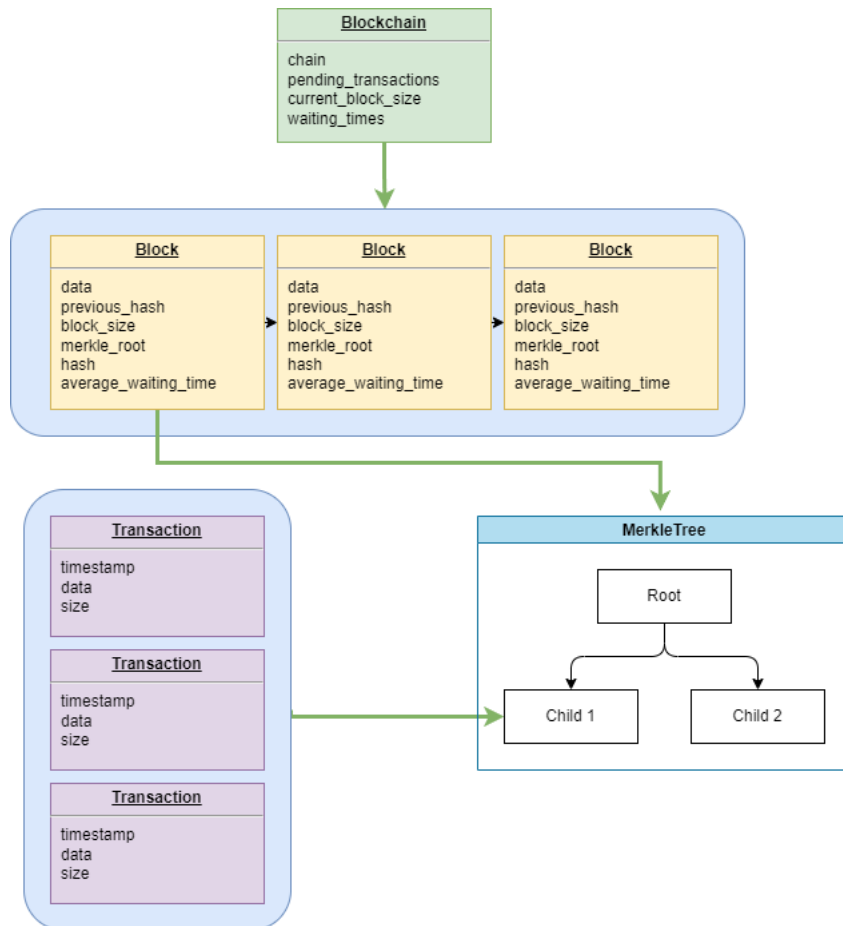


Figure 5.1: Blockchain Simulator

5.1.1 Defining Blockchain

Initially, we created a class containing the property of chain, pending_transactions, current_block_size, waiting_times.

- **chain:** This property is a data structure that holds the entire blockchain. It is an array. The chain property starts with the genesis block, the first block in the blockchain. Subsequent blocks are added to the chain as they are created and validated.
- **pending_transactions :** The pending_transactions property acts as a mempool for the blockchain simulator. It is a collection that temporarily stores uncommitted transactions. When a new transaction is received, it is added to the pending_transactions collection until it is processed and included in a block.
- **current_block_size :** The current_block_size property keeps track of the size of the current block being constructed. It is updated dynamically as transactions are added or removed from the pending_transactions collection. The size of a block can be measured based on the total data size in the mempool.
- **waiting_times:** The waiting_times property represents the average transaction waiting time for each block in the blockchain.

5.1.2 Defining Block

The Block class is a fundamental blockchain component and serves as a container for storing information. It encapsulates various properties essential for maintaining the integrity and security of the blockchain. This class consists of different properties, timestamp, data, previous_hash, block_size, merkle_root, hash.

- **timestamp** The timestamp property represents the exact time the block is created. It is recorded in a Unix format to ensure consistency across different nodes in the network.
- **data** The data property allows for the inclusion of additional information or metadata associated with the block. For example, in a cryptocurrency blockchain, the data may include details about the transactions within the block.
- **previous_hash** The immutability and integrity of the blockchain are established by the previous hash property. It saves the hash value of the chain's preceding block. A connection between blocks is made possible by using the previous block's hash to establish a sequential and impenetrable chain.
- **merkle_root** The Merkle tree, a data structure intended to effectively represent and confirm the integrity of a group of transactions within a block, is the subject of the Merkle root property. Hashing transaction pairs until a single root hash is found creates the Merkle tree. This final hash value, which succinctly encapsulates all the transactions in the block, is stored in the Merkle root property.

- **hash** A key element that contributes to the block's security and immutability is its hash feature. It represents the Merkle tree's overall hash value. Any alteration, no matter how little, to any one of the transactions in the block will produce an entirely new hash value. This characteristic serves as the block's cryptographic fingerprint, enabling other nodes in the network to confirm its integrity.

5.1.3 Defining Transaction

Transaction class is the barebone of the blockchain. This class has many properties, which are essential to represent a transaction, timestamp, data, size, sender and recipient.

- **Timestamp:** The timestamp property captures the exact moment a transaction enters the system. It is recorded using a Unix time format to ensure consistency and chronological ordering of transactions.
- **Data:** the data property holds the primary payload or information associated with the transaction. It can vary depending on the specific use case of the blockchain. For instance, in a cryptocurrency blockchain, the data may include details such as the amount of currency being transferred, any additional message or note, or even intelligent contract instructions.
- **size:** The size property represents the transaction's data size. It indicates the amount of space occupied by the transaction within the blockchain. This information is crucial for optimizing storage efficiency and managing resource utilization in the blockchain network.
- **sender:** The sender property identifies the entity or address that initiates the transaction. It represents the account or party responsible for sending or initiating the transfer of value or information. In a cryptocurrency blockchain, the sender is the account owner authorising funds transfer.
- **recipient:** The entity or address that is supposed to receive the transaction is identified by the recipient property. It symbolizes the account or party receiving the information or value that has been transmitted. The recipient in the context of a blockchain for a cryptocurrency is the account linked to the recipient.

The Transaction class offers a thorough representation of a transaction within the blockchain by including these attributes. It makes saving, retrieving, and verifying transaction information possible, assuring accountability, transparency, and the precise transfer of value or information between parties. Blockchain systems are decentralized, unchangeable, and secure because to the qualities like date, data, size, sender, and recipient.

5.1.4 Simulator

We can start by making an instance of the Blockchain class to simulate a blockchain. We must use the Transaction class to define the data included in transactions. The

transaction size and the sender and receiver addresses must also be included.

Using the `add_transaction` method of the `Blockchain` class, we can add a transaction to the blockchain’s mempool after it has been created. This enables us to add transactions to the mempool continuously. The system continuously monitors the size of the mempool as new transactions are added. The blockchain will generate a block if the size exceeds a set limit. Unless somebody stops it, it will keep adding transactions to the mempool.

The blockchain determines the average block waiting time before creating a block. This is accomplished by deducting the current time from the beginning timestamps of each transaction within the block. This computation gives information on how long transactions typically take to add to a block.

The current simulation setup has a fixed block size of 1MB. However, it is worth noting that this restriction may not be present in a real-world implementation or an OpenAI Gym environment, and the block size can vary dynamically based on the network’s requirements.

By following these steps and incorporating additional details as needed, we can effectively simulate the behaviour of a blockchain system.

5.2 Blockchain environment

To create an OpenAI Gym environment using the proposed blockchain simulator, we need to define three key components of reinforcement learning: observation, action, and reward. The observation space represents the information available to the agent within the environment. The action space defines the possible actions the agent can take. The reward system provides feedback to the agent based on its actions, incentivizing desired behavior and penalizing undesired behavior.

Furthermore, evaluating the performance of the OpenAI Gym environment is essential. This evaluation can be done by tracking various metrics, such as cumulative rewards, transaction waiting times, or successful block mining rates. By comparing the agent’s performance against other models, we can assess the effectiveness of the agent’s decision-making and optimization capabilities within the simulated blockchain environment.

By specifying the observation, action, and reward components and conducting thorough performance evaluations, we created a robust OpenAI Gym environment for training and assessing reinforcement learning agents in the context of blockchain simulations.

5.2.1 Experimentation with multiple algorithms

After preparing our environment, we ran multiple algorithms and evaluated their performance in the given environment. During the training process, we generated

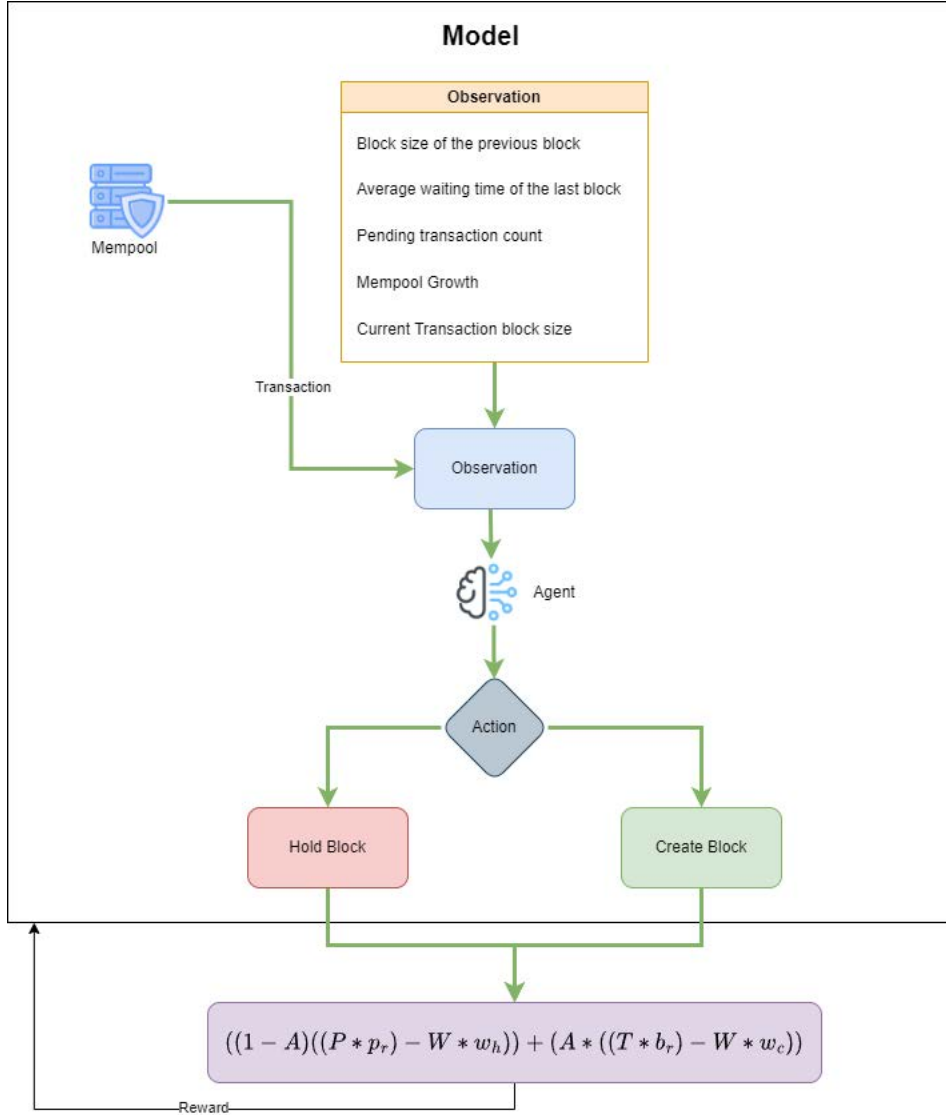


Figure 5.2: ROBB Blockchain Environment

random transactions with varying sizes. Initially, the model’s performance was sub-par, which was expected as it began by taking random actions. Often, it would include only one or two transactions in a block.

However, after training for approximately 25 million timesteps, we observed that the model was still creating blocks too frequently. We hypothesized that this behaviour might be due to the model’s inability to remember past transactions. To address this, we modified our algorithm and introduced a recurrent feature. We opted for Recurrent PPO (Proximal Policy Optimization) this time.

The introduction of the recurrent feature resulted in significant improvement. The model became capable of holding onto transactions and creating blocks in the future when more suitable. Moreover, we noticed a positive trend in the cumulative reward over time, indicating the model’s increasing effectiveness. We have also tried the DQN(Deep Q-Network) and A2C(Advantage Actor Critic) but the models with non-recurrent policy tend to create blocks with single transactions.

5.2.2 Testing environment

To compare our work with existing works, we need a test environment. As work on this topic is not so much so, we must use creative ways to measure our work performance. At first, we saw that some papers want to predict the block size based on the network and create it every timestep so it's not feasible, So instead we choose to use random block size in every timestep and create the block and measure the performance. On the other hand in the bitcoin architecture, we are using a fixed block size of 1MB.

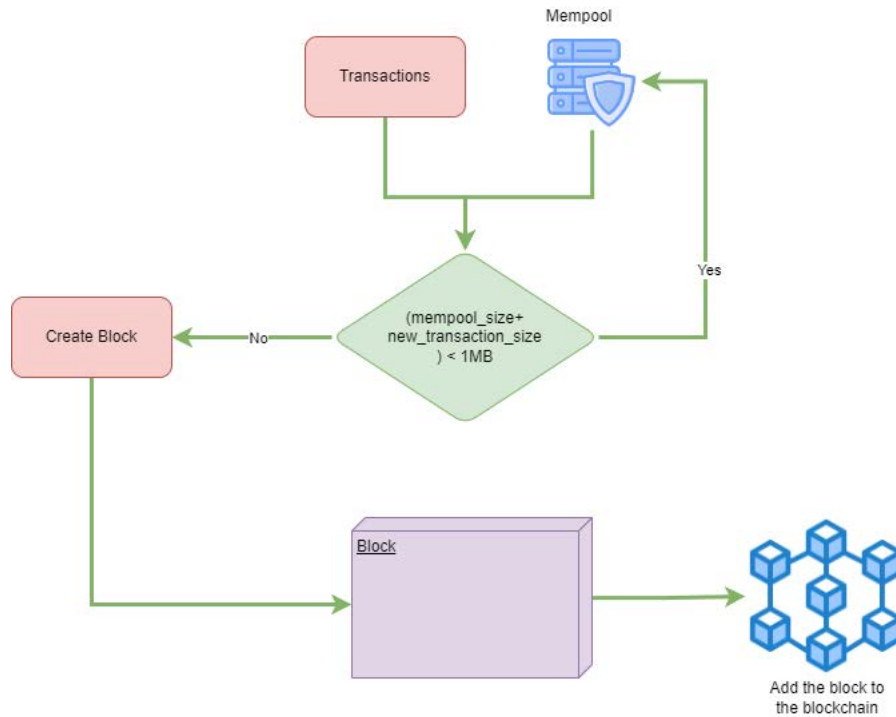


Figure 5.3: Testing environment Fixed block size

The workflow depicted in Figure 5.3 outlines setting up the testing environment with a fixed block size. Initially, we generate a dataset of 5 thousand random transactions. Then, we iterate through each transaction individually, calculating the current size of the mempool and the size of the new transaction. If the total size remains below 1 MB, we add the transaction to the mempool; otherwise, we proceed to create a block. During the block creation, we also measure the waiting time and block utilization, which will be used for later comparisons.

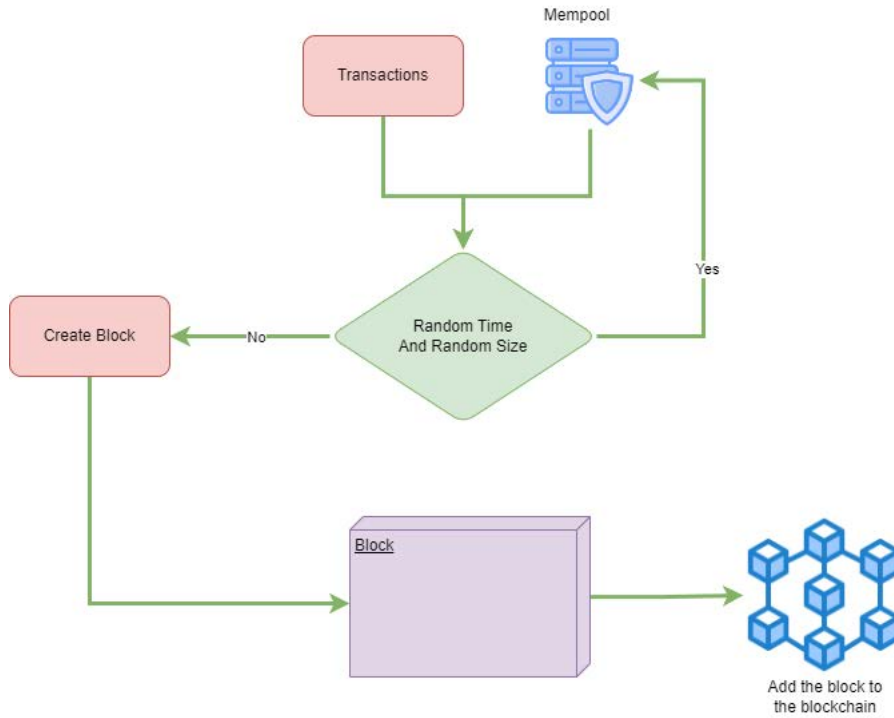


Figure 5.4: Testing environment random block size

The workflow in Figure 5.4 illustrates the procedure for establishing a testing environment with a random block size generated at random intervals. A simulation is created to facilitate comparisons between block size prediction models, wherein a block is generated randomly at a specific time with a random size. During the block creation, we measure both the waiting time and block utilization, enabling subsequent comparisons between different models.

5.3 Model training

Let's first discuss our train metrics. As previously mentioned, we have a value network, so let's discuss the value loss. Fig 5.5 represents the loss of the value network. As we know, that loss function could be noisy, and it's hard to visualize, and that's why we apply a smoothing of 0.5 to smooth the noise. The light colour denotes the original value, and the deep colour determines the smoothed value. So here, the x-axis denotes the timestep, and the y-axis denotes the loss value.

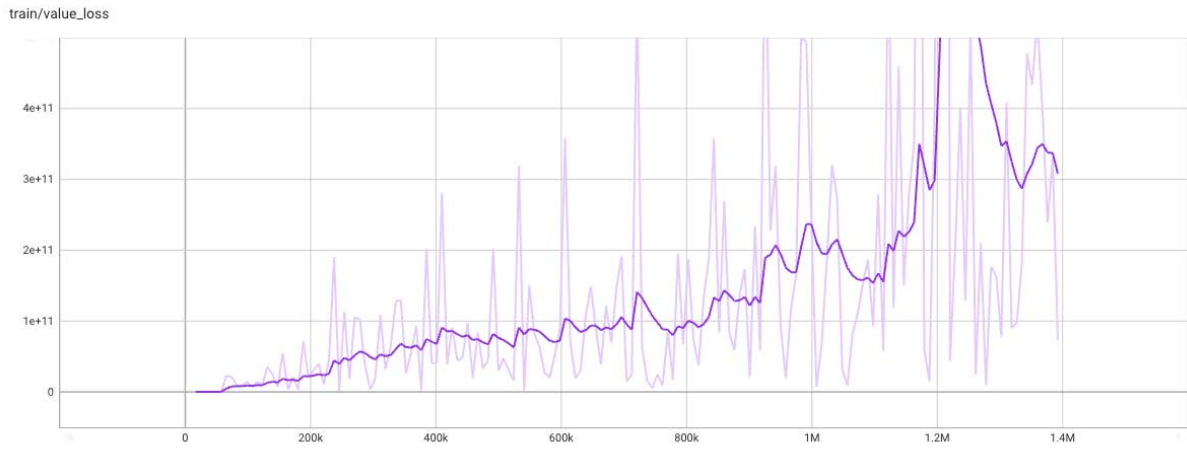


Figure 5.5: Value loss over time

As previously mentioned, we have a policy network as well, so let's discuss the policy gradient loss. Fig 5.6 represents the loss of the policy network. As we know, that loss function could be noisy, and it's hard to visualize, and that's why we apply a smoothing of 0.3 to smooth the noise. The light color denotes the original value, and the deep color determines the smoothed value. So here, the x-axis denotes timestep, and the y-axis denotes the policy gradient loss.

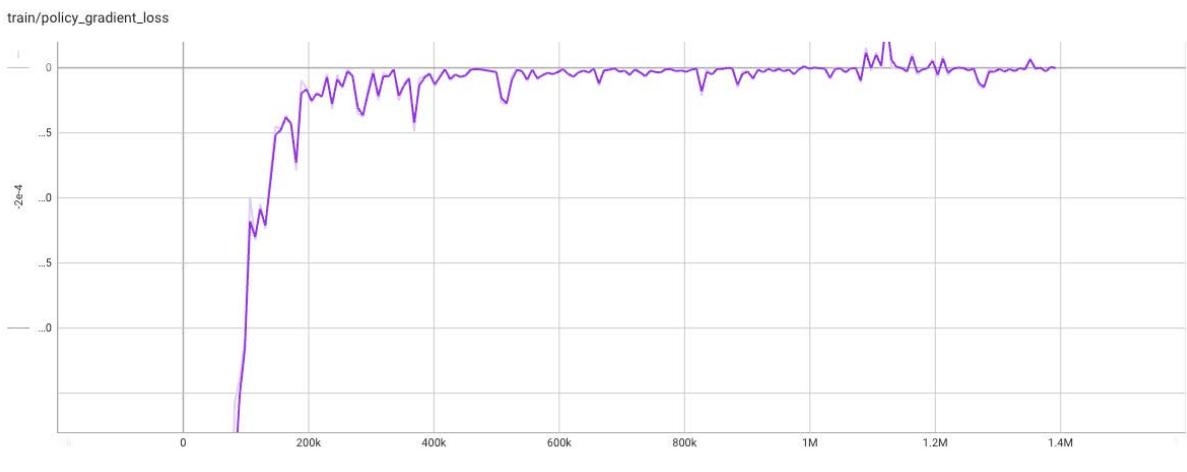


Figure 5.6: Policy loss over time

5.4 Reward tuning

We have some utilization parameters in our reward function. So we have tried out many combinations based on intuition and found out that a set of combinations works perfectly fine. Fig 5.1 describes our different experimentation results for different parameters. And we can see that after using $p_r = 0.9$, $w_h = 0.15$, $b_r = 0.7$, $w_c = 0.2$ we got the lowest waiting time.

Table 5.1: Reward function tuning result

p_r	w_h	b_r	w_c	Waiting Time
0.9	0.15	0.7	0.2	1.8s
0.9	0.3	0.4	0.5	21.3s
0.3	0.4	0.5	0.1	12.5s
0.5	0.6	0.9	0.4	11.3s
0.9	0.7	0.5	0.1	2.6s
0.8	0.3	0.1	0.2	39.1
0.9	0.2	0.8	0	12.6s
1	1	1	1	35.8s
1	0.2	0.5	0.7	11.7s

5.5 Comparison

As mentioned in the previous section, we have created 3 environments to measure the performance of the model. Basically, our goal was to decrease the waiting time for the transactions and use efficient use of blocks.

Let’s discuss the behaviors of different models. We have used multiple algorithms in our environment, hoping to get the best result possible. We have used PPO, DQN, A2C, and RPPO. After experimenting with these algorithms, we got that the RPPO model works best in our scenario. Now let us see the model performance of different models. From the table, we can see that RPPO (Recurrent proximal policy optimization) based model works best in blockchain scenarios where the models because model is not taking small transactions and creating blocks with it; instead, it takes a reasonable mount of blocks and creating the block with it.

Table 5.2: Performance of different models (DQN, PPO, A2C, RPPO)

Model	Waiting time	Total block	BU
RPPO	1.8 s	155 block	100%
PPO	0.3s	4602 block	100%
DQN	0.9s	4357 block	100%
A2C	1.1s	3976 block	100%

To understand the metrics of our result analysis, we have to go through the metrics at first.

- **Block size:** The block means how much data a single block has. We chose a fixed block size of 1 MB for several of the cases in our testing. For other

Table 5.3: Model vs waiting time and block utilization

Model	Block size	Waiting time	Total block	BU
ROBB model	2.67 MB (avg)	1.8 s	155 block	100%
Fixed block size	1 MB	3.7s	237 block	90%
Fixed block size	2 MB	6.2s	124 block	88%
Fixed block size	4 MB	13.1s	63 block	86%
Random timing	N/A	1.3s	1132 block	30%

studies, though, we looked at various block sizes to see how they affected the system’s performance. We can investigate how changing the block size affects transaction performance, storage needs, and overall efficiency.

- **Waiting time:** The time measure sheds light on the typical time a transaction spends in the system’s transaction pool before being included in a block. We created a sample of 1,000 transactions and calculated their typical waiting times for our investigation. This statistic aids in assessing how well the system handles incoming transactions and might reveal any possible bottlenecks or confirmation delays.
- **Total block:** The total blocks metric quantifies the number of blocks the evaluated model generates. It represents the division of transactions into separate blocks based on specific criteria, such as block size or time intervals. By examining the total number of blocks, we can understand the system’s partitioning strategy and assess its ability to handle large transactions.
- **Block utilization (BU):** Block utilization refers to the used space within a block after accommodating all the included transactions. Low block utilization implies ineffective block space utilization and may cause scaling problems. A well-designed system should ideally try to increase the block utilization rate to improve resource usage and transaction throughput.

Firstly, utilising block space is crucial for efficient resource allocation. When blocks have low utilization, most available block space remains unused. This results in inefficient storage capacity utilisation, increasing costs and decreased scalability. By increasing block utilization, we can optimize resource allocation, maximize transaction throughput, and improve the system’s overall efficiency.

Additionally, for a smooth user experience and to enable quicker transaction confirmations, there must be a minimal waiting time. Users anticipate rapid and prompt processing of their transactions in today’s fast-paced world. Users may become irritated and dissatisfied if there is a long wait time. Transactions may be handled quickly by reducing waiting times, resulting in a seamless and practical customer experience.

Furthermore, it’s important to note that even if the waiting time is lower, the block utilization is low, indicating a lack of system efficiency. A high waiting time may suggest that transactions are being processed quickly, but valuable block space is wasted if the block utilization is low. This inefficiency can lead to scalability issues and hinder the system’s ability to handle larger transaction volumes effectively.

In the context of our model, the achieved results demonstrate its efficiency. With a total waiting time of 1.8 seconds and 100% block utilization, our model effectively

minimizes both metrics. This indicates optimal resource utilization and timely transaction processing. In comparison, the random model achieved a lower waiting time of 1.3 seconds but suffered from a significantly low block utilization of 70%. This emphasizes the importance of balancing both metrics, as low block utilization can undermine the benefits of reduced waiting time. Additionally, the Fixed block size model resulted in a waiting time of 3.8 seconds, which is impractical for real-life scenarios where faster transaction confirmations are crucial. We have also found that the average block created by our system is 2.67MB.

We can design a system that optimizes resource utilization, enhances transaction processing speed, and ensures scalability in real-life scenarios by utilising block and waiting time. This leads to a more efficient and reliable network that meets user expectations and supports the demands of a rapidly evolving digital landscape.

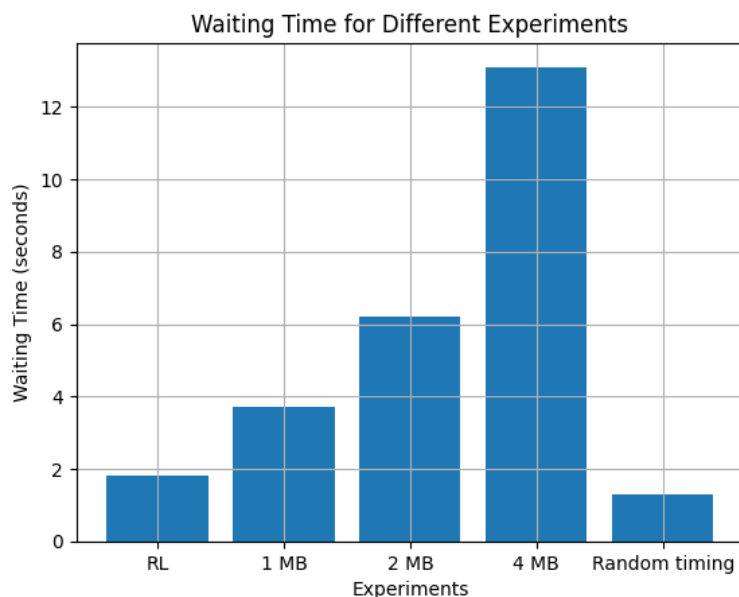


Figure 5.7: Comparison between different models and their waiting time

In Fig 5.7, we can see that for the random timing model, the waiting time is the lowest. It indicates that for random timing mode user needs to wait for the lowest time. We can see from the graph that for our RL model we got a waiting time of 1.8 seconds, for the 1MB model where the blocksize is fixed by 1 MB we got a waiting time of 3.7 seconds, for the 2MB fixed block size model we got a waiting time of 6.2 seconds, for 4MB fixed block size model we got a waiting time of 13.1 seconds finally for random timing model we got a waiting time of 1.3 seconds. The results show we should go with the random timing model but we cannot rely on the random timing model it could be possible that the simulated environment was suitable for the random timing model but in the real world, it will not work.

In fig 5.8, we can see the block utilization percentage of different models. As we can see from the figure for the RL model block utilization is 100% because we are using a dynamic block size so it will only create a block with the transactions present in mempool that's why we don't have any wastage. But for the 1MB fixed block size model we can see that the block utilization rate is 90%, for 2MB fixed block size we can see the block utilization rate is 88% for the 4MB fixed block size model we got a block utilization of 86% finally for random timing model we got a block utilization

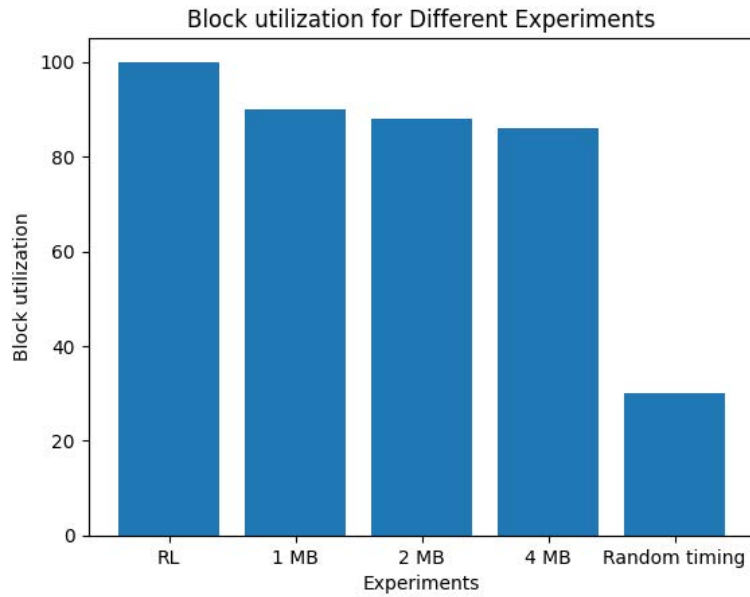


Figure 5.8: Comparison between different models and their block utilization

of 30%. So the final results indicate that our model wins in the criteria of block utilization. Block utilization rate is an important parameter that cannot be ignored even in the current implementation of the blockchain we got poor block utilization rate.

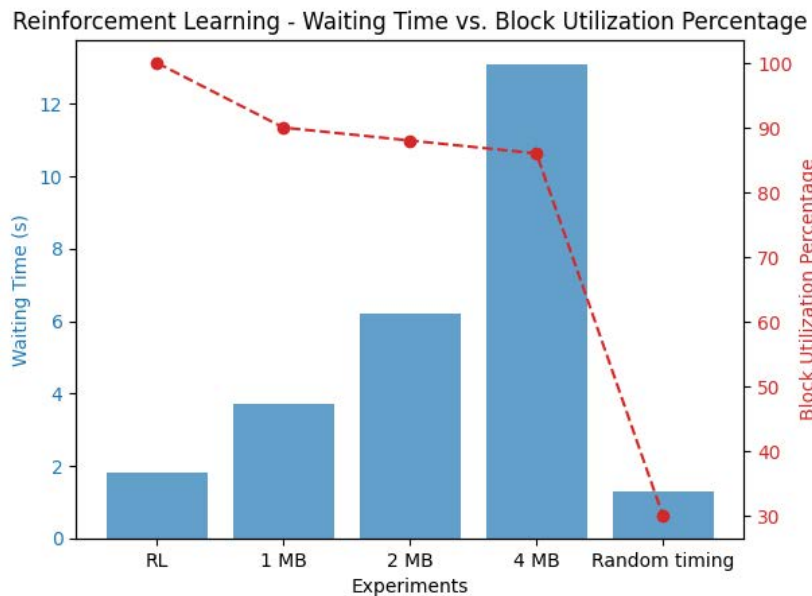


Figure 5.9: Model vs waiting time and block utilization

In Fig 5.9, we observe the waiting time comparison among various models. It is noteworthy that our reinforcement learning model demonstrates the second-lowest waiting time. However, it also exhibits a block utilization of 100%. Conversely, the random timing model displays the lowest waiting time but is associated with nearly 30% block utilization. Consequently, based on these findings, we can assert that ROBB surpasses the other models in terms of performance. We just compute the

waiting time only after a transaction is put into the mempool and end the waiting time if the transaction is included in a block and we discard the processing time of the RL model.

Chapter 6

Conclusion

6.1 Conclusion

In conclusion, this research paper successfully explored integrating reinforcement learning models and blockchain simulation to predict optimal block creation and holding strategies. Multiple reinforcement learning methods were assessed through thorough testing and analysis, with RPPO (Recurrent Proximal Policy Optimization) producing the best outcomes.

A blockchain simulation using reinforcement learning models produced impressive results, such as a 0% block wastage rate and an average waiting time of 1.3 seconds. These outcomes demonstrate how the suggested approach may optimize block generation choices and boost blockchain performance.

Blockchain simulation's realism and reinforcement learning's capacity for intelligent decision-making have significantly advanced the subject. This study shows how blockchain systems can be improved and offers helpful details on the broader applicability of reinforcement learning strategies in various fields.

But it's important to recognize the limitations of this study. It is necessary to perform additional research to examine the scalability and generalizability of the suggested approach across various blockchain networks and variable settings because the tests and evaluations were conducted under certain constraints and presumptions. This research has improved our grasp of the connections between blockchain technology and reinforcement learning. The outcomes highlight the possibility of using sophisticated decision-making algorithms to raise the effectiveness and efficiency of blockchain systems. This study also lays the groundwork for future studies that aim to develop reinforcement learning's applicability in decentralized systems and continuously enhance blockchain performance.

It is important to note that many reinforcement learning algorithms, including Proximal Policy Optimization (PPO) and RPPO, were used to implement this research. It was discovered through comparison analysis that RPPO produced the best outcomes for enhancing block generation tactics. Additionally, a reinforcement learning-based environment specifically designed for blockchain simulations was created by the research using OpenAI Gym.

6.2 Future Plan

Reinforcement learning is a versatile approach that extends beyond its applications in games [2]. It holds great potential for various domains, including the field of blockchain. We can achieve enhanced performance and explore new possibilities by leveraging reinforcement learning in the blockchain. In the future, integrating multi-agent reinforcement learning can further amplify these benefits.

In addition to relying on simulators to measure performance, there is a promising opportunity to implement reinforcement learning algorithms directly in real-life private blockchains. This approach enables fine-tuning and optimization within the actual blockchain environment. By doing so, we can improve block utilization and significantly reduce waiting times, ultimately facilitating seamless scalability.

As we look ahead, future research and development should focus on advancing block utilization techniques and minimizing waiting times. By addressing these challenges, blockchain technology can scale effortlessly, ensuring efficient and effective operations in various contexts.

Bibliography

- [1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *Decentralized business review*, 2008.
- [2] I. Szita, “Reinforcement learning in games,” in *Reinforcement Learning: State-of-the-art*, Springer, 2012, pp. 539–577.
- [3] J. Poon and T. Dryja, “The bitcoin lightning network,” *Scalable o-chain instant payments*, 2015.
- [4] G. Brockman, V. Cheung, L. Pettersson, *et al.*, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.
- [5] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, “{Bitcoin-ng}: A scalable blockchain protocol,” in *13th USENIX symposium on networked systems design and implementation (NSDI 16)*, 2016, pp. 45–59.
- [6] M. Di Pierro, “What is the blockchain?” *Computing in Science & Engineering*, vol. 19, no. 5, pp. 92–95, 2017.
- [7] J. Göbel and A. Krzesinski, “Increased block size and bitcoin blockchain dynamics,” in *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, 2017, pp. 1–6. DOI: 10.1109/ATNAC.2017.8215367.
- [8] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, “Blockchain,” *Business & Information Systems Engineering*, vol. 59, pp. 183–187, 2017.
- [9] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, “When machine learning meets blockchain: A decentralized, privacy-preserving and secure design,” in *2018 IEEE international conference on big data (big data)*, IEEE, 2018, pp. 1178–1187.
- [10] D. Mechkaroska, V. Dimitrova, and A. Popovska-Mitrovikj, “Analysis of the possibilities for improvement of blockchain technology,” in *2018 26th Telecommunications Forum (TELFOR)*, IEEE, 2018, pp. 1–4.
- [11] S. Gao, T. Yu, J. Zhu, and W. Cai, “T-pbft: An eigentrust-based practical byzantine fault tolerance consensus algorithm,” *China Communications*, vol. 16, no. 12, pp. 111–123, 2019.
- [12] A. Hughes, A. Park, J. Kietzmann, and C. Archer-Brown, “Beyond bitcoin: What blockchain and distributed ledger technologies mean for firms,” *Business Horizons*, vol. 62, no. 3, pp. 273–281, 2019.
- [13] S. Zeadally and J. B. Abdo, “Blockchain: Trends and future opportunities,” *Internet Technology Letters*, vol. 2, no. 6, e130, 2019.

- [14] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, “A survey on the security of blockchain systems,” *Future generation computer systems*, vol. 107, pp. 841–853, 2020.
- [15] D. Yang, C. Long, H. Xu, and S. Peng, “A review on scalability of blockchain,” in *Proceedings of the 2020 the 2nd International Conference on Blockchain Technology*, 2020, pp. 1–6.
- [16] M. Asante, G. Epiphaniou, C. Maple, H. Al-Khateeb, M. Bottarelli, and K. Z. Ghafoor, “Distributed ledger technologies in supply chain security management: A comprehensive survey,” *IEEE Transactions on Engineering Management*, 2021.
- [17] W. Fan, J. Kim, I. Kim, X. Zhou, and S.-Y. Chang, “Performance analyses for applying machine learning on bitcoin miners,” in *2021 International Conference on Electronics, Information, and Communication (ICEIC)*, 2021, pp. 1–4. DOI: 10.1109/ICEIC51217.2021.9369776.
- [18] S. Gupta and M. Sadoghi, “Blockchain transaction processing,” *arXiv preprint arXiv:2107.11592*, 2021.
- [19] B. Lashkari and P. Musilek, “A comprehensive review of blockchain consensus mechanisms,” *IEEE Access*, vol. 9, pp. 43 620–43 652, 2021.
- [20] P. R. Nair and D. R. Dorai, “Evaluation of performance and security of proof of work and proof of stake using blockchain,” in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, IEEE, 2021, pp. 279–283.
- [21] R. Paulavičius, S. Grigaitis, and E. Filatovas, “A systematic review and empirical analysis of blockchain simulators,” *IEEE Access*, vol. PP, pp. 1–1, Mar. 2021. DOI: 10.1109/ACCESS.2021.3063324.
- [22] S. M. S. Saad, R. Z. R. M. Radzi, and S. H. Othman, “Comparative analysis of the blockchain consensus algorithm between proof of stake and delegated proof of stake,” in *2021 International Conference on Data Science and Its Applications (ICoDSA)*, IEEE, 2021, pp. 175–180.
- [23] F. Saleh, “Blockchain without waste: Proof-of-stake,” *The Review of financial studies*, vol. 34, no. 3, pp. 1156–1190, 2021.
- [24] T. Wang, S. C. Liew, and S. Zhang, “When blockchain meets ai: Optimal mining strategy achieved by machine learning,” *International Journal of Intelligent Systems*, vol. 36, no. 5, pp. 2183–2207, 2021.
- [25] T. Wang, C. Zhao, Q. Yang, S. Zhang, and S. C. Liew, “Ethna: Analyzing the underlying peer-to-peer network of ethereum blockchain,” *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 3, pp. 2131–2146, 2021.
- [26] Z. Zhang, X. Song, L. Liu, J. Yin, Y. Wang, and D. Lan, “Recent advances in blockchain and artificial intelligence integration: Feasibility analysis, research issues, applications, challenges, and future work,” *Security and Communication Networks*, vol. 2021, pp. 1–15, 2021.
- [27] Y. Liu, J. Liu, M. A. V. Salles, *et al.*, “Building blocks of sharding blockchain systems: Concepts, approaches, and open problems,” *Computer Science Review*, vol. 46, p. 100 513, 2022.

- [28] G. Ramezan, C. Leung, C. Miao, *et al.*, “A mining strategy for minimizing waiting time in blockchains for time-sensitive applications,” *Wireless Communications and Mobile Computing*, vol. 2022, 2022.
- [29] J. Yang, J. Dai, H. B. Gooi, H. D. Nguyen, and A. Paudel, “A proof-of-authority blockchain-based distributed control system for islanded microgrids,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 11, pp. 8287–8297, 2022.