IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Digital Object Identifier

Proximal Policy Optimization for profitable trading

AMIT DUTTA[1], MD SULTAN PARVEZ[1], MD. GOLAM RABIUL ALAM[1]
[1]Department of Computer Science and Engineering, Brac University, 66 Mohakhali, Dhaka 1212, Bangladesh

Corresponding authors: M. Ali Akber Dewan (e-mail: adewan@athabascau.ca) and Md. Golam Rabiul Alam (e-mail: rabiul.alam@bracu.ac.bd).

This work was supported in part by BRAC University, Bangladesh.

ABSTRACT This work aims to create an agent that can trade efficiently in the stock market. This implementation utilizes proximal policy optimization (PPO) to train the agent and OpenAIGym to simulate a financial market environment. The deep learning community finds research on the Financial market less interesting because of the difficulty and the expensive nature of the financial market. The main objective of this research is to introduce a trading model using Reinforcement learning that can help traders to visualize the market better and make decisions on it. The model will create a better solution to the current anomaly and develop efficient trading strategies. This paper describes the challenges of the financial market and how machine learning helps resolve them. Trade market indicators are used to understand the condition of financial or economic trends, and this paper combines reinforcement learning algorithms and financial indicators.

INDEX TERMS Trading Indicators; Reinforcement Learning; Blockchain; OpenAIGym; Proximal Policy Optimization

I. INTRODUCTION

As of 2019, the global trading market is worth $90 trillion[1]. The trading market is crucial for raising capital for businesses. However, Market volatility is perhaps the most misunderstood concept in trade market investing. If the market is relatively stable, it is low-risk volatility, but if it changes now and then, it is called high-risk volatility. Any investor can benefit from high-risk volatility. However, conservative traders want a more stable marketplace. As the price fluctuates more, an investor can buy shares of the company, and with the company's growth, it is easier for them to profit. Reinforcement learning is an area of machine learning that studies how an agent will take actions in an environment against a given set of states. The main goal of this is to maximize the cumulative reward. In the operations research and control literature, reinforcement learning is called approximate dynamic programming or neuro-dynamic programming. The main difference between Reinforcement learning and supervised learning is that supervised learning needs to be fed with data and all those data have the correct action. On the contrary, the reinforcement agent decides what to do to perform the given task. Without a training dataset, it is bound to learn from experience. Much research has been conducted on making new strategies and techniques to find or build an Agent that can efficiently trade in the market. Beyond a shadow of a doubt, it is basic to think about creating a trade Market Agent. Nevertheless, creating an Agent that can survive the worst conditions is much more essential. There are many existing models, but since most are based on existing rule-based algorithms, they can be easily influenced. It is very normal for an Agent not to profit even though every condition suggests there will be profit. To overcome this significant issue, the thought of creating an Agent that will not be subject to these influences. It may be possible to create such an Agent with the help of Reinforcement learning. The use of different reinforcement learning Algorithms and comparing them based on the results can be fruitful. An understanding of Market indicators will help greatly since these indicators can detect the pace and nature of the market. The Stock Market is full of opportunities. But there have always been problems. All kinds of things can influence the market. Weather, political instability, political changes, threats, boycotts and strikes, economic trends even pandemics are factors of stock market problems. In the early days of trading, people depended on an inside man or position to understand the condition of the market. Since the invention of the internet, researching the market has become straightforward, and various news about

the companies and market are available. The market was always volatile, but this further contributed to the volatility. As a result of such volatility, trading becomes much riskier. [2]In the year 2008, the Dow Jones industrial average fell by 777.68 points. It was reported as the largest point drop in history until 2020. The Stock Market did recover from that recession. However, whether to invest in the market greatly remains a question mark. People or companies always want to play safe, so they are always searching for people or models that can give a sense of security. Some even prioritize a consistent low-profit model over a high-profit inconsistent model. This paper proposes a trading agent based on reinforcement learning that can learn and understand the market environment and trade efficiently even in the worst conditions.

A. RESEARCH MOTIVATION

Much research has been conducted on making new strategies and techniques to find or build an Agent, that can efficiently trade in the market. The stock market has always been the center of attention for the financial market since it was first introduced in the early 16th century. Beyond a shadow of a doubt, it is basic to think about creating a trade Market Agent. But it is much more essential to create an Agent that can survive the worst of conditions. The majority of the existing models are based on Supervised learning. The limitation of those models is they can only match the performance of the existing human model. This also means that the models can be influenced by human-made rules. It is already mentioned how volatile the stock market is. So, it is very normal for an Agent not to profit even though every condition suggests there will be profit. To overcome this major issue, the thought of creating an Agent that will not be subject to these sorts of influences. It may be possible to create such an Agent with the help of Reinforcement learning. The use of different reinforcement learning Algorithms and comparing them based on the results can be fruitful. An understanding of Market indicators will help greatly Since these indicators can detect the pace and nature of the Market and trade efficiently.

B. RESEARCH OBJECTIVE AND AIM

This thesis carried out a comparative study of different Reinforcement learning algorithms and studied different risk measurement techniques, various Python libraries, and toolkits that help develop an artificial environment. This paper reflects some research about different Neural Networks and the indicators of the Stock market. Data sets of different tech companies, such as Google, Microsoft, Ap- ple is used here. Those data are collected from the website yahoo. finance. At first, using the given data set it performs the training part of the model. Then the agent is tested on 90 90-day fragment of the data set. Different fragments of the data set were used in different iterations. There are many iterations of observation. In each observation, a different set of indicators was used. Either that or the same indicators is used on different data sets to figure out the efficiency of the proposed model. By comparing the result of each iteration it gives a pathway to compare the set of indicators. Comparing the algorithms based on the results is a good technique to get the expected outcome. Whether it is safe to invest or not, the stock market will always remain important in the world of finance. So, it is important to address the volatility of the stock market and come up with a consistent solution. Every time the market faces a recession all the companies across the globe suffer from it thus hampering the overall development of the world. Solving this problem of the Stock The market can be very beneficial for all companies. Thus, bringing a significant pace difference in development activities

C. RESEARCH CHALLENGES

The Stock Market is full of opportunities. But there have always been problems. The trade market can be influenced by all kinds of things. Weather, Political Instability, Political Changes, Threats, Boycott and strikes, economic trends even Pandemics are factors of Stock Market problems. In the early days of trading people dependent on an inside man or position to understand the condition on market. Since the invention of the internet researching the market becomes very easy and various news about the companies and market were available. The Market was always volatile but this contributes to the volatility even further. As a result of such volatility trading become much riskier. **RePEc:wsi:wschap:9781786348401_0010**In the year 2008, the Dow Jones industrial average fell by 777.68 points. It was reported as the largest point drop in history until 2020. The Stock Market did recover from that recession. However, whether to invest in the market greatly remains a question mark. People or Companies always want to play safe so they are always in search of people or Models that can give give them security. Some even prioritize a consistent low-profit model over a high-profit inconsistent model. This paper proposes a Trading Agent based on Reinforcement Learning. This agent uses the ability of Reinforcement learning to learn and understand the market environment thus trade efficiently even in the worst of the conditions.

D. RESEARCH CONTRIBUTIONS

this model follows an approach to building an Agent that will consistently trade with profits or at least will not face a loss within a given period. This may address the insecurities companies have about the trading market.

1) Comparative study of Reinforcement learning, Risk management technique and various python library.
2) Collecting real data of companies like Apple and Microsoft from yahoo.finance.
3) Training and testing of agent iteratively.
4) Make a profitable and yet consistent agent.

E. PAPER ORGANIZATION

1) Chapter 1 is an Introduction where motivation, problem statement, objectives, and contribution are discussed.

2) Chapter 2 is a literature review where background and related work are discussed. In the literature review part, it reviews the previous work on the Trade Market.
3) Chapter 3 is Proposed Method, here we discussed our approach to building the agent.
4) Chapter 4 is the Algorithm, here the description of the used algorithm is discussed.
5) Chapter 5 includes performance analysis and results.
6) Chapter 6 is Conclusion and Future works where the conclusion consists of all work till now and future works include the scope of improvement.

II. BACKGROUND OF BLOCKCHAIN

This section looks into various research conducted in the field of Blockchain and multiple performance aspects of Reinforcement Learning.

A. RELATED WORKS

Reinforcement learning in finance is an active research area as there are not many practical examples of reinforcement learning because it is mainly used in a simulated environment. It is already mentioned that stock market volatility is a major problem as machine learning is mainly used to learn the patterns from data. There are few works on machine learning in finance topics. A neural network is a powerful function approximator and if we can train it for a large amount of time it can reverse-engineer the algorithms and strategies that other traders are running and then learn to exploit them. TD (0) is a machine learning algorithm that can learn from past experiences. [3]There is an experimental result of the Korean stock market based on TD (0) algorithm. The RMS error between value grades and the return grades is 3.07.[4]This paper applies the Adaptive Network Fuzzy Inference System (ANFIS) supplemented by the use of reinforcement learning (RL) as a non-arbitrage algorithmic trading system. Here ANFIS-RL determines the delay in the predicted cycle. This helps to find the best point to long go or short.[5]This paper uses (SARIMA) and the discrete Markov process combined to generate price series with period. A constructed scenario tree helps to plan the model. [6]There is another approach instead of using the policy-based approach that paper has applied a value-based approach and using the popular Q-learning algorithm. The result is quite good but the main problem is it's taking a large amount of time to make a good profit. [7] Another paper has used an RNN (Recurrent neural network) and a task-aware backpropagation through time method to cope with the gradient vanishing issue in deep learning. That model is used to represent financial signals and trading.[8]Here this paper tried to figure out how hyper-parameter configuration affects the performance of DNNS and then compare the outcome in Naïve Bayes, k-nearest neighbor, random forest and support vector machines model.[9]This paper shows the novel hybrid view for a financial series and critic adaptation stock price forecasting architecture using direct reinforcement. It also refers policies-matching ratio utility function.[10]This paper proposes Multi-node Evolutionary Neural Networks for Deep Learning (MENNDL) for automated network selection on computational clusters through hyper-parameter optimization.

After studying these papers, it is understandable that RNN is the best approach to represent recurrent signals and through Q-learning, it can make a profit with a long-term investment mind. But none of the mentioned papers represents the power of an Artificial Neural Network. So, this model tries to explore the financial domain with ANN (Artificial Neural Network)
A paper named (Go-Explore: a New Approach for Hard-Exploration Problems) [11] uses a different type of algorithm on different games and measures the accuracy and mentions a new Reinforcement Learning approach called Go-Explore. The current model also tried to implement this algorithm as this is a new technique but there is not much on the internet about this topic cause the paper itself has been published on (30 Jan 2019).[9] Another paper suggests using a different technical indicator for the buy, sell, hold strategy in the stock market.

In **mechkaroska2018analysis**, authors mentioned the scalability issue of the Bitcoin network as Nakamoto introduced a block size limit of 1MB for every block in the public BlockChain **nakamoto2008bitcoin**. 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 authors also presented 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 **9369776** 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 several 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, 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 **simulator**, a comprehensive and exhaustive analysis of the efficacy of machine learning techniques for predicting the price of cryptocurrencies is conducted. 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, 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 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 the significance of future research and optimization efforts.

In **eyal2016bitcoin**, 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 several key components and scaling mechanisms. It employs a leader-based strategy in which a leader is designated for a predetermined period to process transactions and create micro-blocks. 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. 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 it is difficult to measure the performance of the model. In **8215367**, authors 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 **9369776**, authors 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 worse than supervised algorithms. Interestingly, Logistic regression performs the best, impacting the lowest mining rate.

In **wang2021blockchain**, 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. 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.

The research **yue2023glassdb** presents a novel architecture called GlassDB which is a verifiable ledger database system that aims to protect the integrity of data history and support transactions. This paper proposes a design space for verifiable ledger databases along three dimensions: abstraction, threat model, and performance. The paper compares GlassDB with existing systems and shows that GlassDB achieves higher performance and efficiency by using a novel authenticated data structure, deferred verification, and batching techniques. GlassDB inherits the verifiability of transparency logs but supports transactions and offers high performance. It extends a ledger-like key-value store with a data structure for efficient proofs and adds a concurrency control mechanism for transactions. GlassDB batches independent operations from concurrent transactions when updating the core data structures. The paper also introduces new benchmarks for evaluating verifiable ledger databases with verification workloads. Experimental results demonstrate that GlassDB is an efficient, transactional, and verifiable ledger database system.

In **10.14778/3415478.3415540**, the authors discuss a state-of-the-art blockchain technology called ledgerDB. The paper illustrates that the introduction of Blockchain has gained significant attention. However, it is seen that in fact, many apps on permissioned blockchains do not profit from the decentralized architecture. When decentralized architecture is adopted but not needed, system performance is often hindered, resulting in low throughput, high latency, and large storage costs. Hence, we propose LedgerDB on Alibaba Cloud, which is a centralized ledger database with tamper-evidence and non-repudiation capabilities comparable to blockchain, and provides robust auditability. LedgerDB provides substantially

higher throughput compared to blockchains. It enables enhanced auditability by implementing a TSA two-way peg protocol, which inhibits malicious activities from both consumers and service providers. LedgerDB offers verifiable data deletions requested by many real-world applications, which can remove obsolete records for storage saving and hide records for regulatory purposes, without compromising its verifiability. Experimental evaluation demonstrates that LedgerDB's throughput is 80X higher than state-of-the-art permissioned blockchain (i.e., Hyperledger Fabric). Many blockchain customers (e.g., IP protection and supply chain) on Alibaba Cloud have migrated to LedgerDB for its fast throughput, low latency, excellent auditability, and ease of use.

In **10.1145/3589774**, the literature provides a comprehensive exploration of VeDB, a trusted relational database system that integrates both software (Ve-S) and hardware (Ve-H) enabled solutions for data notarization and lineage. It discusses the improvement of latency through the use of Intel Converged Security and Management Engine (CSME) and the implementation of a Physical Secure Timer in ARM architecture and Risc-V enclave. The text delves into the verification process and fine-grained verification in VeDB, detailing data verification path selection, storage, ledger table design, and transaction types. It also outlines the algorithm for inserting data into a Verified Shrub Append-only (VSA) data structure and generating a receipt. The paper evaluates the performance of VeDB in different hardware configurations and workloads, comparing the performance of Ve-S and Ve-H applications in data notarization and data lineage. Overall, the paper presents VeDB as a practical and efficient solution for trusted relational databases, addressing limitations in conventional blockchains and centralized ledger databases.

The study **9928220** offers a novel consensus technique for blockchain termed platform-free proof of federated learning (PF-PoFL). It harnesses the processing capacity of blockchain nodes to train federated learning models for artificial intelligence tasks. The paper claims that PF-PoFL can achieve the following advantages: it removes the dependence on a central platform and enables decentralized task outsourcing and reward distribution; it preserves user-level differential privacy for miners to prevent data leakage during federated mining; it dynamically forms the optimal pool structure for diverse federated learning tasks using a game-theoretical mechanism. The research assesses the performance of PF-PoFL using simulations and shows that it can achieve stable blockchain throughput, desirable model performance, lower computing cost, and higher efficiency.

The research **9631953** reviews the integration of blockchain technologies for securing space-air-ground integrated networks (SAGINs), which are composed of satellites, aerial devices, and terrestrial networks. The study covers the characteristics, security concerns, and possible applications of SAGINs, and then reviews the blockchain ideas, features, kinds, and problems. The article also discusses alternative blockchain-based solutions for SAGIN security in multiple scenarios and domains, such as communication, networking, resource management, data sharing, and emergency response. The article also analyzes the prospects of blockchain in artificial intelligence and beyond 5G networks and presents open research directions for constructing future blockchain-empowered SAGINs.

The research **9835536** proposed LedgerDB, a centralized ledger database that achieves both high external auditability and quick verification. The study offers a concept of Dasein verification, which formalizes the what-when-who validation for ledger data. The research presents various unique mechanisms to facilitate Dasein verification, such as fractal accumulating model (fam) for existence verification, time notary protocol (T-Ledger) for time verification, and clue merged tree (CM-Tree) for lineage verification. The study also assesses the performance and completeness of LedgerDB and shows that it beats existing ledger systems in terms of verification efficiency and auditability.

The following sub-sections discuss the background of Blockchain.

B. TYPES OF BLOCKCHAIN

In this subsection, we discuss the two types of blockchain depending upon the accessibility of blockchain (Permissioned and Permission-less) **8513729**. In our research, the proposed architecture is for a permission-less blockchain as our focus is to establish a model that optimizes the transactions with Bitcoin which is implemented using a permission-less blockchain which is necessary as the transaction time is slower in permission-less blockchain.

1) **Permissioned Blockchain:** In a permissioned blockchain, access to the network and participation in the consensus process are restricted to a preset set of nodes or businesses. Participants are normally known and identifiable. Permissioned blockchains are typically utilized in corporate or enterprise contexts where a controlled and known group of participants cooperate. These blockchains are useful for applications like supply chain management, where various stakeholders need to collaborate. Permissioned blockchains often give higher privacy and confidentiality features. Transactions may be viewable only to authorized participants, which might be critical for sensitive company data. Since permissioned blockchains have a smaller number of participants and frequently use speedier consensus techniques, they can offer faster transaction processing times and reduced costs compared to permissionless blockchains.
2) **Permission-less Blockchain:** In a permissionless blockchain, also known as a public blockchain, anybody can join the network, participate in the con-

sensus process, and validate transactions without requiring permission. Examples of permissionless blockchains include Bitcoin and Ethereum. Permissionless blockchains are often more decentralized, as they allow unrestricted participation from anybody around the world. This decentralization is a critical element of cryptocurrencies like Bitcoin, where no single institution or group has control. Participants in permissionless blockchains can maintain a measure of anonymity, as network access is not tied to real-world identities. Due to their open and decentralized nature, permissionless blockchains may have slower transaction processing times and higher fees compared to permissioned blockchains.

C. BLOCK

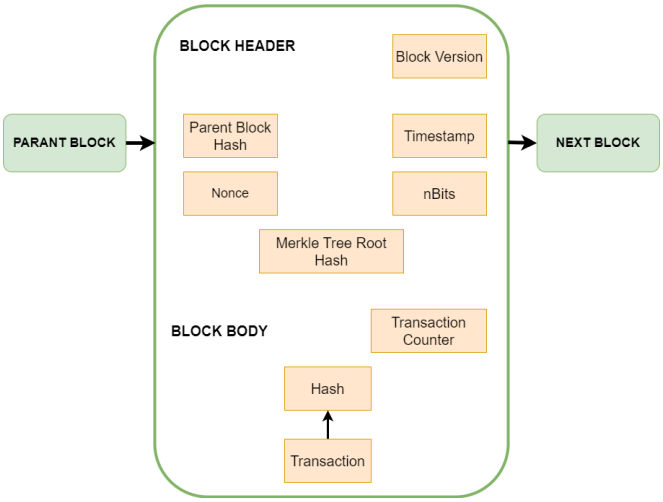

FIGURE 1. Structure of a block in Blockchain

A block is a fundamental blockchain component that contains a set of transactions **di2017blockchain**. It is a data structure that organizes and stores information within the blockchain. Here are the key elements and concepts associated with a block:

1) 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.

D. 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:

1) Cryptographic Hash Function

A cryptographic hash function is a mathematical algorithm that takes an input (data) and produces a fixed-size output (hash value)**liu2022building**. 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.

2) 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.

3) 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.

4) 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.

5) 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.

6) Security

Hashes contribute to the security of a blockchain through their cryptographic properties **li2020survey**. 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.

7) 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.

E. 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.

F. TRANSACTION

In blockchain technology, a transaction represents an action or exchange of value recorded on the blockchain**gupta2021blockchain**. 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 incentivize 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 it makes 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.

G. CONSENSUS MECHANISM

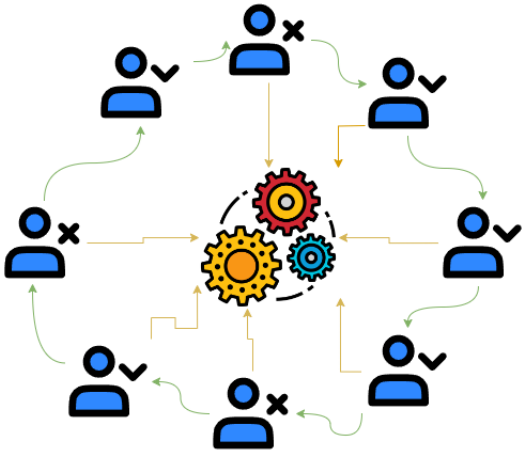

FIGURE 2. Consensus Mechanism

An essential element of blockchain technology that ensures agreement and validation of transactions across a decentralized network is the consensus mechanism**lashkari2021comprehensive**. In Fig 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:
   - **Proof of Work (PoW)**: PoW **nair2021evaluation** 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 **saleh2021blockchain**, 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 **saad2021comparative** 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**yang2022proof**, 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 **gao2019t** 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.

H. PEER-TO-PEER NETWORK

In the context of blockchain, a peer-to-peer (P2P) network refers to the decentralized network architecture used by blockchain systems**wang2021ethna**. 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 decentralized, where multiple nodes validate and store trans-

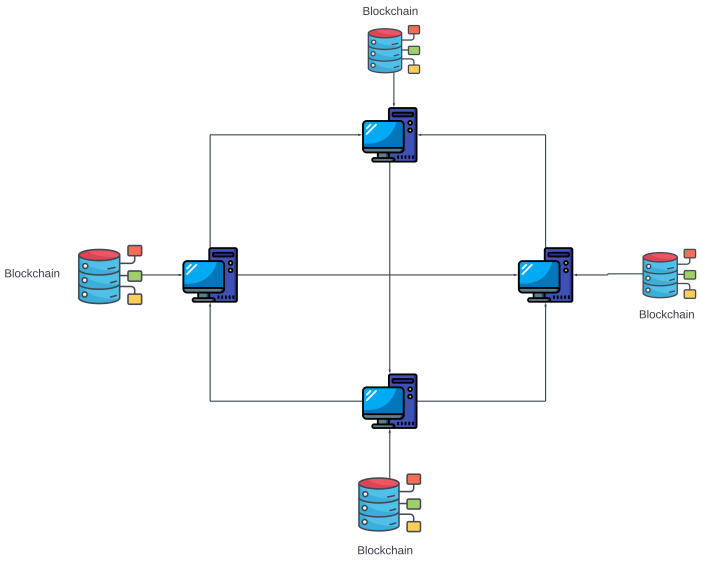

FIGURE 3. P2P network diagram

actions. 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**yang2020review**. 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.

I. DISTRIBUTED LEDGER

In the context of blockchain, a distributed ledger**hughes2019beyond** 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 decentralized, 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**asante2021distributed** 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.

J. TIMESTAMP AUTHORITY (TSA)

A Timestamp Authority (TSA) **HeppSchoenhalsGondekGipp+20** is a trustworthy third-party institution or service that provides timestamping services in the context of data and document verification. TSA plays a crucial role in preserving the integrity and validity of digital information by giving a trustworthy timestamp for when a specific piece of data or document was created or modified. While TSAs are not limited to blockchain technology, they are typically used in conjunction with blockchain to strengthen the trustworthiness and tamper-evidence of timestamped data. In this section, we discuss how timestamps from the Timestamp Authority (TSA) can make blockchain more secure in storing data and for transactions.

1) **Timestamping Data**: When a person or organization wishes to timestamp a piece of data or a document, they send it to the TSA. The TSA stores a cryptographic hash of the data combined with a timestamp identifying the precise day and time when the data was received. This timestamp is generally obtained using a very accurate time source, making it credible for legal and evidentiary purposes.
2) **Cryptographic Hash**: The cryptographic hash **HYLA20201065** of the data is a unique and fixed-size string of characters generated by applying a hash function to the original data. Even a small change in the data will result in a significantly different hash value. This property ensures that the timestamp is bound to the specific data.
3) **Tamper-Evidence**: Once the data's timestamp and hash are recorded on the blockchain, they become nearly tamper-evident. Any effort to change the original data will result in a different hash value. Since the blockchain is immutable, the original timestamped data and hash remain intact and verifiable by anybody.
4) **Verification**: To validate the validity and integrity of timestamped data, users can resort to the blockchain. By comparing the stored hash on the blockchain with the hash of the data they hold, they may validate that the data has not been altered since the timestamp was established.
5) **Legal and Evidentary values**: Timestamped data from a trustworthy TSA, especially when housed on a blockchain, can have legal and evidentiary importance. It can act as evidence in judicial processes to prove the existence of certain information at a specific point in time.
6) **Distributed Trust**: The usage of blockchain for timestamping promotes trust and reliability. Since blockchain is decentralized and maintained by a network of nodes, it eliminates the need to rely on a single central authority for timestamp validation.

K. CURRENT BLOCKCHAIN ARCHITECTURE

In this subsection, we discuss the current blockchain architecture.

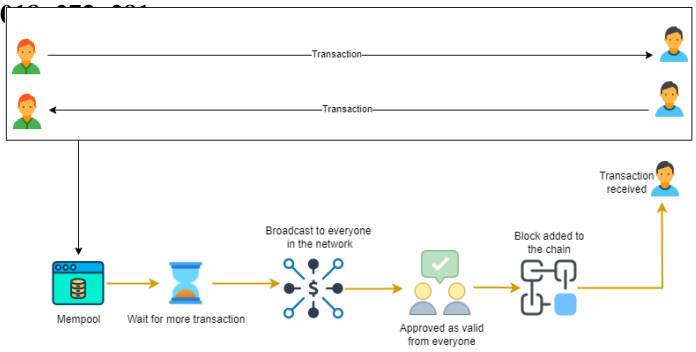

FIGURE 4. Current Bitcoin blockchain Architecture

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**poon2015bitcoin** 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. Afterward, 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 can 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 labor 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.

III. METHODOLOGIES

For our proposed model, we have used a reinforcement learning package called *OpenAI Gym***brockman2016openai** which 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 it's 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**brockman2016openai** 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 maximizing 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**brockman2016openai** 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**brockman2016openai** 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.

Fig. 5 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

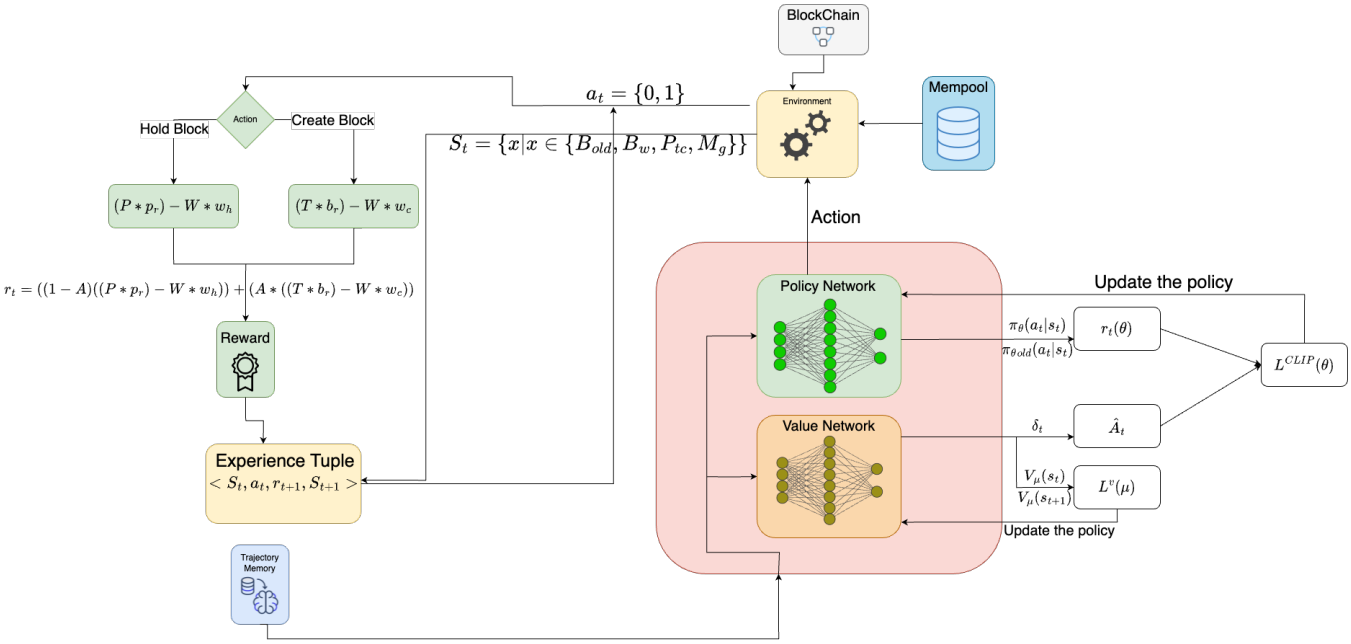

FIGURE 5. ROBB system workflow

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 favorable 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 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 parameters 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 9 and Equation 10.

A. 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.

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** $k = 0, 1, 2, ...$ **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 $\widehat{RR}_t$.
10: Compute advantage estimates, $\widehat{AA}_t$ (using any method of advantage estimation) based on the current value function $V_{\phi k}$.
11: Update the policy by maximizing the RPPO-clip objective:

$$\theta_{k+1} = \underset{\theta}{argmax} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta k}(a_t|s_t)} A^{\pi_{\theta k}}(s_t, a_t), g(\in A^{\pi_{\theta k}})(s_t, a_t) \right) \quad (1)$$

Use Adam optimizer to update the gradient.
12: Fit value function by regression on mean-squared error:
$\phi_{k+1} = \underset{\phi}{argmin} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} (V_\phi(s_t) - \widehat{RR}_t)^2$
use gradient descent algorithm to update the weights
13: **end for**

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.

$$L^{PG}(\theta) = E_t \left[log \pi_\theta(a_t|s_t) * A_t \right] \quad (2)$$

The idea was that by taking a gradient ascent step on this function (equivalent to taking a 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 weight updates.

$$\mathcal{L}^{CLIP}_{\theta_k}(\theta) = \underset{\tau \sim \pi_k}{\mathrm{E}} \left[\sum_{t=0}^{T} \left[\min \left(r_t(\theta) \hat{A}^{\pi_k}_t, \mathrm{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right) \hat{A}^{\pi_k}_t \right) \right] \right] \quad (3)$$

Another important thing we need to know is the Ratio Function. The ratio function measures the difference between the new and old policies and controls the number 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.

$$r_t = \frac{\pi_\theta(a_t|s_t))}{\pi_{\theta old}(a_t|s_t))} \quad (4)$$

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.

$$L = \frac{1}{|T|} \sum_t (V(s_t) - V_\theta(s_t))^2 \quad (5)$$

We can use this 6 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}.

$$V(s_t) = r_t + \gamma \cdot V(s_{t+1}) \quad (6)$$

B. RPPO BASED REINFORCEMENT LEARNING AGENT DESIGN

There are existing deep reinforcement learning methods like DQN, PPO, A2C, and DDPG, which are commy used in the fields of reinforcement learning. 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 promptly. 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 with the full picture we have

provided the 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 7.

$$S_t = \{x | x \in \{B_{old}, B_w, P_{tc}, M_g\}\} \tag{7}$$

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 behaviors 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 utilization. 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 8

$$a_t = \{0, 1\} \tag{8}$$

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 utilize available space by creating dynamic-sized blocks (action 1).

This design choice reflects the real-world behavior 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 behavior. 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, the 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:

$$r_{th} = (P * p_r) - W * w_h \tag{9}$$

Reward for action 1:

$$r_{tc} = (T * b_r) - W * w_c \tag{10}$$

Combine reward function will be:

$$r_t = ((1 - A)((P * p_r) - W * w_h)) + (A * ((T * b_r) - W * w_c)) \tag{11}$$

Table 1 shows the various constants for the values of the reward function.

TABLE 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

IV. PROTOTYPE DESIGN

A. BLOCKCHAIN SIMULATOR FOR ROBB

In this Section, first, we will explain how we create a blockchain simulator followed by a thorough discussion on the testing environment.

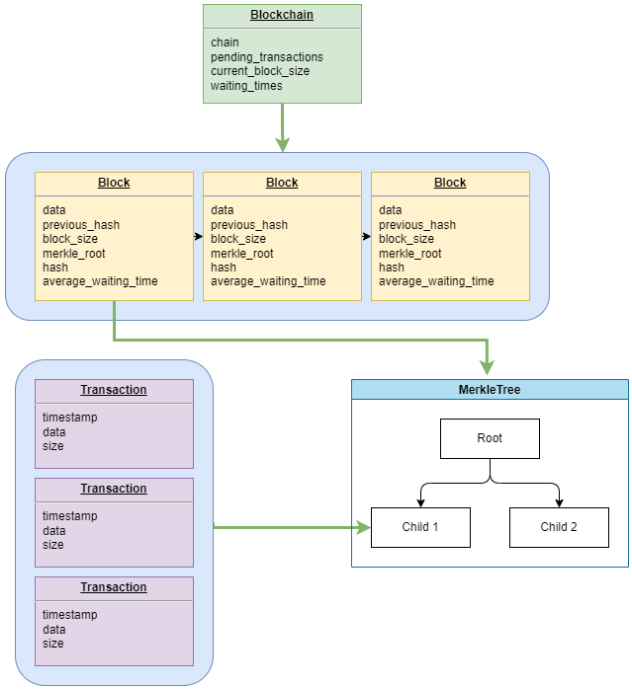

FIGURE 6. Blockchain Simulator

B. 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.

C. 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.

D. 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 authorizing 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 of their qualities like date, data, size, sender, and recipient.

E. 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.

F. BLOCKCHAIN ENVIRONMENT

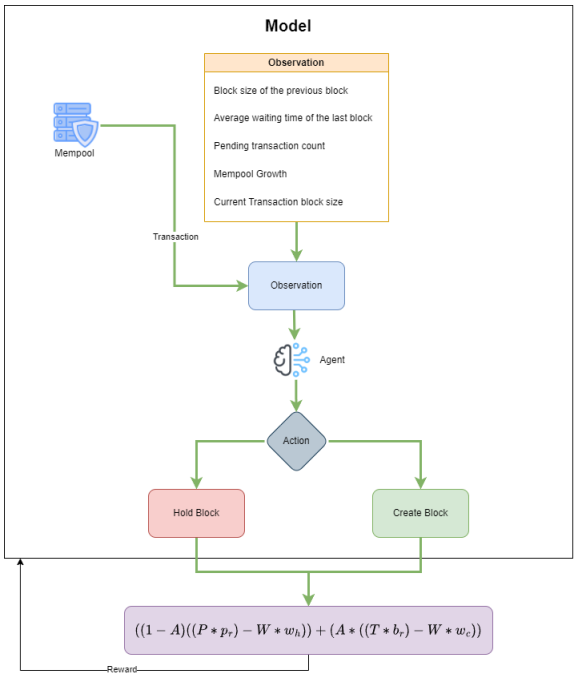

FIGURE 7. ROBB Blockchain Environment

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.

G. 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 random transactions with varying sizes. Initially, the model's performance

was subpar, 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 behavior 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.

H. 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 research **MONEM202412** wanted to predict the block size based on the network and create it every timestep so it's not feasible, instead we chose 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.

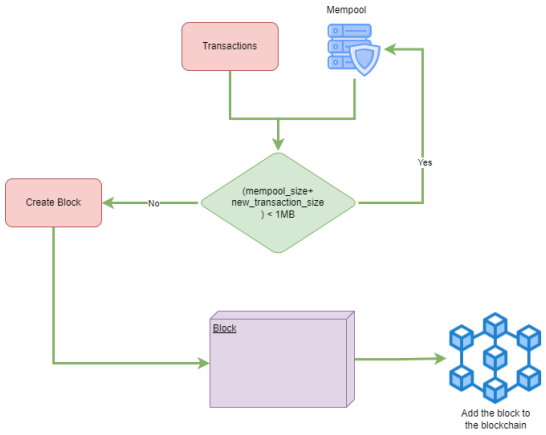

FIGURE 8. Testing environment Fixed block size

The workflow depicted in Figure 8 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.

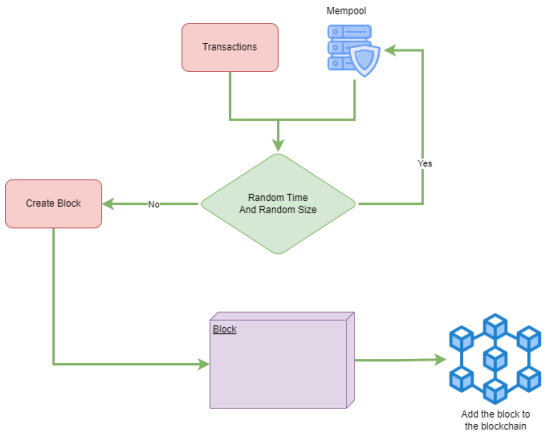

FIGURE 9. Testing environment random block size

Figure 9 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.

I. 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 10 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 color denotes the original value, and the deep color determines the smoothed value. So here, the x-axis denotes the timestep, and the y-axis denotes the loss value.

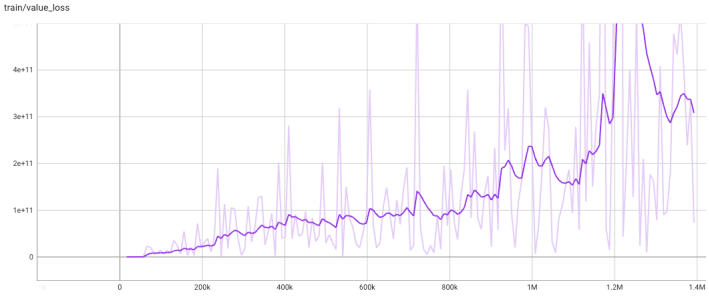

FIGURE 10. Value loss over time

As previously mentioned, we have a policy network as well, so let's discuss the policy gradient loss. Fig 11 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.

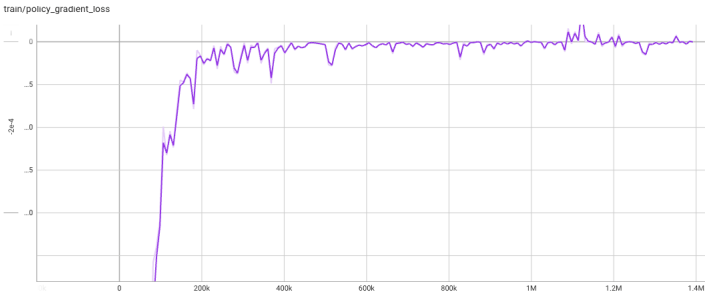

FIGURE 11. Policy loss over time

J. 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. The reward function tuning results are illustrated in Table 2.

TABLE 2. Reward function tuning result

p_r	w_h	b_r	w_c	Waiting Time
0.9	0.15	0.7	0.2	1.5s
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

V. PERFORMANCE EVALUATION

As mentioned in the previous section, we have created a simulation environment to measure the performance of the proposed model. In this section, we have discussed the performance metrics and performance of the proposed model and presented a comparative study with different existing approaches.

A. PERFORMANCE MATRICES

To understand the metrics of our result analysis we have to go through the metrics at first. In this subsection, we have discussed the performance matrices that we have used to evaluate our models. A short description of each matrix is listed below.

- **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 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.

B. RESULT ANALYSIS

We have analyzed the performance of our proposed method and also studied the performance of different baseline strategies followed by different blockchain networks for block size determination.

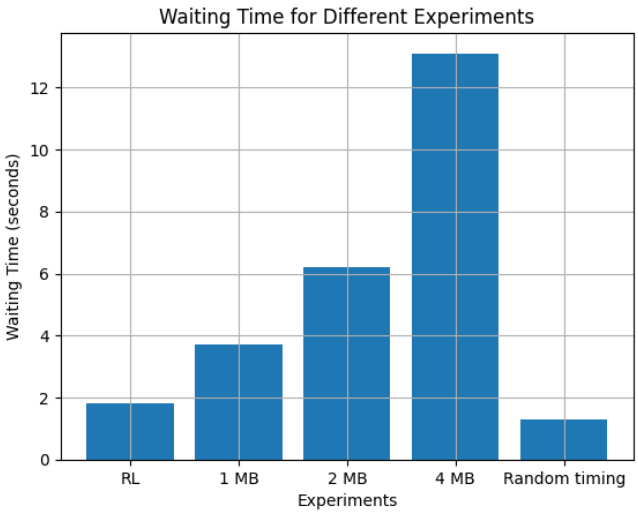

FIGURE 12. Comparison between different models and their waiting time

In Fig 12, we can see that for the random block size model, the waiting time is the lowest. It indicates that for random block size mode, the 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 block size 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 block size model we got a waiting time of 1.3 seconds. The results show we should go with the random block size model but we cannot rely on the random block size model it could be possible that the simulated environment was suitable for the random block size model but in the real world, it will not work.

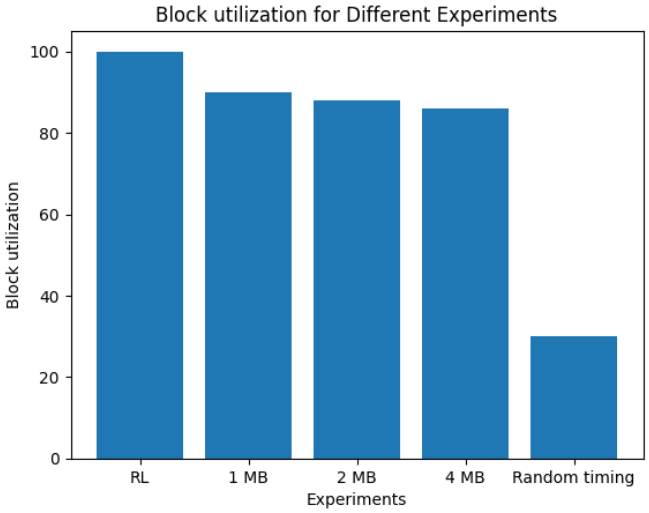

FIGURE 13. Comparison between different models and their block utilization

In fig 13, 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 block size model we got a 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 have a poor block utilization rate.

In Fig 14, 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 block size 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.

Table 3 illustrates the waiting time and block utilization of different strategies that benchmark our proposed model in various block sizes. In the table, we have presented

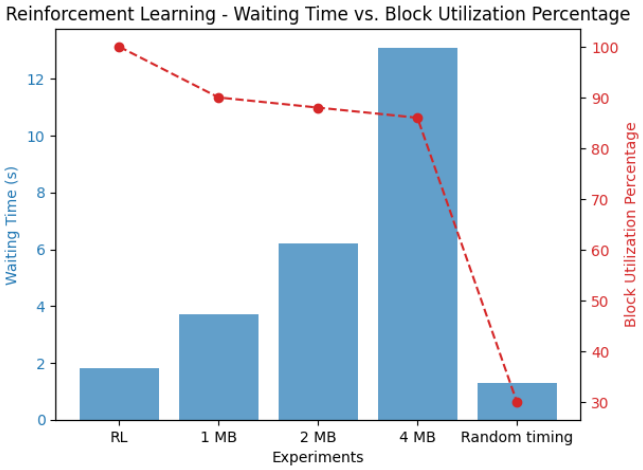

FIGURE 14. Comparison between different models and their waiting time and block wastage

the simulation results considering the different block sizes of Bitcoin and Ethereum blockchain networks. Bitcoin is one of the cryptocurrencies that utilizes fixed block size (generally 1 MB) to keep waiting time below 7 seconds **stonehem2016satoshi**. But recently, Bitcoin block size has increased to 2 MB and can have a theoretical maximum value of 4 MB **magazine_what**. Also, for benchmarking, we have simulated the Ethereum blockchain along with Bitcoin, which does not have a fixed block size but has an average block size of about 4 MB and can have a max value of up to 12 MB. From the table, we can see that our proposed model that utilizes a dynamic block size performs better in benchmark compared to Bitcoin and Ethereum.

TABLE 3. Comparative study among the baseline strategies vs the proposed model of block size determination

Strategies	Block size	Waiting time
Fixed block size **stonehem2016satoshi**	1 MB	3.7s
Fixed block size **magazine_what**	2 MB	6.2s
Fixed block size **magazine_what**	4 MB	13.1s
Random block size **gencer2018decentralization**	Dynamic	1.3s
Proposed ROBB	**Dynamic**	**1.8 s**

Moving on to the discussion of the behaviors of different models, we have used multiple algorithms in our environment for comparison, hoping to get the best result possible. We have used PPO, DQN, A2C, and RPPO. After experimenting with these algorithms, we found that the RPPO model works best in our scenario. Now let us see the model performance of different models. From Table 4, we can see that RPPO (Recurrent proximal policy optimization) based model works best in blockchain scenarios where the models because the model is not taking small transactions and creating blocks with it; instead, it takes a reasonable amount of blocks and creating the block with it.

TABLE 4. Performance of different RL Algorithms (DQN, PPO, A2C, RPPO)

RL Algorithms	Waiting time	Total block	BU
PPO	0.3s	4602 block	100%
DQN	0.9s	4357 block	100%
A2C	1.1s	3976 block	100%
RPPO (Proposed)	**1.8 s**	**155 block**	**100%**

C. DISCUSSION

Utilizing 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 utilization, 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 can design a system that optimizes resource utilization, enhances transaction processing speed, and ensures scalability in real-life scenarios by utilizing 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.

VI. CONCLUSION

In conclusion, this research 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 proposed approach may optimize block generation choices and boost blockchain performance.

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.

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. 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.

ACKNOWLEDGMENT

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) and Brac University, Bangladesh.

References

[1] J. Pound, *Global stock markets gained $17 trillion in value in 2019*, Dec. 2019. [Online]. Available: https://www.cnbc.com/2019/12/24/global-stock-markets-gained-17-trillion-in-value-in-2019.html.

[2] K. Amadeo, *When and why did the stock market crash in 2008?* Apr. 2020. [Online]. Available: https://www.thebalance.com/stock-market-crash-of-2008-3305535#:~:text=The%20stock%20market%20crash%20of%202008%20occurred%20on%20Sept.,largest%20point%20drop%20in%20history..

[3] J. W. Lee, "Stock price prediction using reinforcement learning," in *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570)*, IEEE, vol. 1, 2001, pp. 690–695.

[4] Z. Tan, C. Quek, and P. Y. Cheng, "Stock trading with cycles: A financial application of anfis and reinforcement learning," *Expert Systems with Applications*, vol. 38, no. 5, pp. 4741–4755, 2011.

[5] M. Olsson and L. Soder, "Modeling real-time balancing power market prices using combined sarima and markov processes," *IEEE Transactions on Power Systems*, vol. 23, no. 2, pp. 443–450, 2008.

[6] J. E. Moody and M. Saffell, "Reinforcement learning for trading," in *Advances in Neural Information Processing Systems*, 1999, pp. 917–923.

[7] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 653–664, 2016.

[8] A. Koutsoukas, K. J. Monaghan, X. Li, and J. Huan, "Deep-learning: Investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data," *Journal of cheminformatics*, vol. 9, no. 1, p. 42, 2017.

[9] E. Hurwitz and T. Marwala, "Suitability of using technical indicator-based strategies as potential strategies within intelligent trading systems," in *2011 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, 2011, pp. 80–84.

[10] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," in *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, 2015, pp. 1–5.

[11] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune, "Go-explore: A new approach for hard-exploration problems," *arXiv preprint arXiv:1901.10995*, 2019.

AMIT DUTTA received the B.S. degree in computer science and engineering from BRAC University in 2020 and the M.S. degree in computer science and engineering from BRAC University, in 2023. His research includes the improvement of machine learning, especially in the field of reinforcement learning. Apart from his research, he works as a senior software engineer.

NAFIZ IMTIAZ RAFIN received the B.S. degree in computer science and engineering from BRAC University in 2023 and currently pursuing the M.Sc. degree in Computer Science and Engineering from BRAC University. His research interest includes the improvement of machine learning, especially in the fields of reinforcement learning, Deep Learning (Neural Networks), Image Processing, Natural Language Processing, Transfer Learning, Attention Mechanism, etc.

M. ALI AKBER DEWAN (Member, IEEE) received the B.Sc. degree in computer science and engineering from Khulna University, Bangladesh, in 2003, and the Ph.D. degree in computer engineering from Kyung Hee University, South Korea, in 2009. From 2003 to 2008, he was a Lecturer at the Department of Computer Science and Engineering, Chittagong University of Engineering and Technology, Bangladesh, where he was an Assistant Professor, in 2009. From 2009 to 2012, he was a Postdoctoral Researcher at Concordia University, Montreal, QC, Canada. From 2012 to 2014, he was a Research Associate at the École de Technologie Supérieure, Montreal. He is currently an Associate Professor with the School of Computing and Information Systems, Athabasca University, Athabasca, AB, Canada. His research interests include artificial intelligence in education, affective computing, computer vision, machine learning, biometric recognition, and medical image analysis. He has served as an editorial board member, a chair/co-chair, and a TPC member for several prestigious journals and conferences. He received the Dean's Award and the Excellent Research Achievement Award for his excellent academic performance and research achievements during his Ph.D. studies in South Korea.

MD. GOLAM RABIUL ALAM received B.S. and M.S. degrees in Computer Science and Engineering, and Information Technology respectively. He received a Ph.D. in Computer Engineering from Kyung Hee University, South Korea in 2017.

He served as a Post-doctoral researcher in the Computer Science and Engineering Department, Kyung Hee University, Korea from March 2017 to February 2018. He is currently serving as a Professor in the Computer Science and Engineering Department at BRAC University, Bangladesh. His research interests include healthcare informatics, mobile cloud and Edge computing, ambient intelligence, and persuasive technology. He is a member of IEEE IES, CES, CS, SPS, CIS, and ComSoc. He received several best paper awards from prestigious conferences.