The University of Hong Kong
Department of Computer Science, Faculty of Engineering

Blockchain Mining with Machine Learning

Final Report

Supervisor: Dr. Liu Qi

Member: Cheung Yau Shing Jonathan (3035783560)

Group: FYP23033
Date of submission: 26/04/2024

Abstract

Blockchain miners today heavily rely on the brute force method for mining. However, this approach consumes immense computational resources and has led to various environmental problems. Inspired by recent advancements in artificial intelligence, this project seeks to explore the application of machine learning to blockchain mining. More specifically, this work introduces the use of machine learning to 1) enhance nonce finding, 2) optimize transaction selection, and 3) discover optimal chain-level mining strategies. Historical mining data were first collected from blockchain explorers and APIs for model training and experiment evaluation. To enhance nonce finding, different iteration methods were compared and regression was applied to predict the starting nonce for iteration. A reinforcement learning algorithm was also developed to optimize the selection of block transactions while adhering to the weight limit. Finally, the blockchain mining process was modelled as a Markov Decision Process and solved using both undiscounted and discounted solvers to attain an optimal chain-level mining strategy. It is hoped that the insights gained from this project could encourage miners to adopt more environmentally conscious mining methods in the future.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Liu, Qi. for the guidance and support provided throughout the project. His expertise and mentorship have been instrumental in shaping the direction and success of this work. I would also like to extend my thanks to the University of Hong Kong for providing the necessary computational resources and budget crucial for my work.

Table of Contents

List of Figures

List of Tables

List of Equations

List of Abbreviations

Abbreviation	Definition
PoW	Proof of work
DNN	Deep neural network
RL	Reinforcement learning
ML	Machine learning
A2C	Advantage Actor-Critic
MDP	Markov Decision Process
RATAF	Running Average of Total Aggregated Fees
TWh	Terawatt-hours
ASICs	Application-Specific Integrated Circuits
API	Application Programming Interface
GPU	Graphics processing unit
HKU	The University of Hong Kong

1. Introduction

In this section, we provide an introduction of the project, including the background, motivation, project objectives, proposed deliverables, and an outline of this progress report.

1.1 Background

Blockchain technology is becoming increasingly popular. As a decentralized system, it could increase trust, security, transparency and traceability of data across a business network [1]. It is therefore applied in various industries. In finance, it is used for cross-border payments and smart contracts. In Supply Chain Management, blockchain allows businesses and consumers to track the origin, movement, and authenticity of products. In healthcare, Blockchain helps securely store and share patient medical records.

Mining is crucial in blockchain for block creation and transaction validation [2]. The block creation process differs across various consensus mechanisms employed in the blockchain. Proof of Work is the one of the most prominent consensus mechanisms. To create a block under PoW, miners have to 1) collect pending transactions, 2) verify their validity, and 3) construct the block header by solving the hash problem [3]. The hash problem involves miners searching for a nonce (numerical value) that generates a hash value complying to predefined criteria [3]. This process ensures the security and integrity of the blockchain by discouraging malicious actors from altering the blockchain's history [4].

1.2 Motivation

There has been a long-held belief that trial and error is the only feasible and profitable block-mining strategy for PoW [5]. Therefore, miners with greater computational resources have a higher capacity to explore a larger number of solutions, thus increasing their chances of winning. This resource-based competition has led to excessive energy consumption. Figure 1 displays the annual energy consumption of Bitcoin in 2021, measured in terawatt-hours (TWh). It can be seen that Bitcoin's energy consumption exceeded 100 terawatt-hours (TWh) annually, surpassing the total energy usage of nations such as Sweden, Ukraine, Norway, and Argentina — and nearly half that of the United Kingdom. The predominant use of fossil fuels in mining operations not only contributes to substantial greenhouse gas emissions but also intensifies global climate change concerns. Furthermore, the need for miners to continually upgrade equipment fosters a cycle of electronic waste, exacerbating the environmental impact of outdated mining rigs. Consequently, PoW blockchain mining emerges as one of the most environmentally damaging practices.

1.3 Project Objectives

Traditional Proof of Work (PoW) mining practices are not only energy-intensive but also increasingly non-viable due to escalating computational demands and environmental regulations. Therefore, the goal of this project is to apply machine learning to improve the efficiency and effectiveness of PoW mining methods. With this approach, we could not only enhance miners' mining outcomes but also move away from the traditional brute force methods, thus reducing the carbon footprint of blockchain mining.

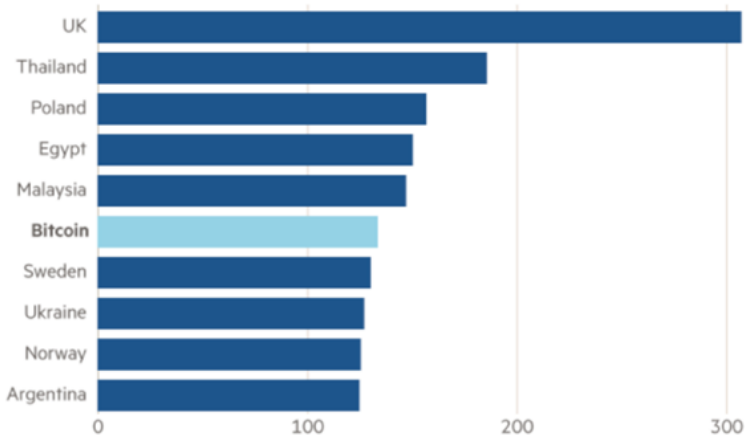


Figure 1. Energy consumption of bitcoin, compared with countries' reading (TWh) in 2021. Adopted from [6]

To achieve this, we first conducted a detailed literature review to explore the intersection between machine learning and blockchain technology. Afterwards, we collected historical Bitcoin block and transaction data from blockchain explorers and APIs. This data was then used to examine different nonce iteration methods and to train various regression models for predicting the starting nonce for iteration. We also used the Bitcoin transaction data to develop a Reinforcement Learning algorithm to optimize transaction selection. Lastly, we modeled the blockchain mining process as a Markov Decision Process (MDP) and solved the problem to attain optimal chain-level mining strategies. Experiments specific to each of the improvement areas were conducted, where existing mining methods were compared to the developed ML-based methods to reflect the improved efficiency and effectiveness of mining.

1.4 Project Deliverables

By the end of the project, the following 3 deliverables will be presented:

1) Research Report: A comprehensive research report will be produced, providing an in-depth analysis of the background, objectives, methodology, and findings of the project. The report will examine and compare existing approaches with the developed algorithm to reflect its effectiveness in mining.

2) Code: The project will provide the entire codebase, including relevant scripts, modules, and libraries utilized in developing the machine learning model and conducting the experiments. This code will enable others to replicate the work, extend it further, and validate the results. Proper documentation and comments within the code will ensure its comprehensibility.

3) Presentation and Demonstration: A comprehensive presentation summarizing the project contributions will be prepared, highlighting the key findings, methodology, and outcomes. Additionally, a demonstration on the machine learning model's functionality and performance will be provided to showcase its practical application in blockchain mining. The presentation and demonstration will be completed in a video format and be uploaded to the project website.

1.5 Paper Outline

This report will discuss the current completion status of the project in detail. Section 1 will provide an introduction to the project. Section 2 will focus on the detailed project background and literature review, highlighting the necessary background knowledge, as well as existing applications of machine learning in blockchain. Section 3 will outline the methodology employed in this project, providing insights into the research approach. Section 4 will cover experiments and results, while Section 5 will focus on the project schedule. Finally, Section 6 will cover limitations and future work, and a conclusion will be given in Section 7.

2. Project Background and Literature Review

In this section, we first provide a comprehensive overview of the foundational concepts involved in the project. Then, we present a detailed literature review on related methodologies for machine learning in blockchain technology.

2.1 Project background

This section provides a comprehensive overview of the background knowledge required for our project.

2.1.1 Blockchain Technology

Blockchain technology is a transformative approach to managing data and transactions in a decentralized manner. It operates as a distributed ledger that records all transactions across a network of computers, ensuring transparency, security, and integrity.

Blockchain technology has three main features. First, Decentralization [3]: Unlike traditional databases managed by central authorities, blockchain technology employs a distributed network of nodes, which means that no single entity has control over the entire chain. This reduces the risk of central points of failure and increases resistance to malicious attacks. Second, Immutability [7]: Once a transaction is confirmed and recorded in a block on the blockchain, it cannot be altered. This immutability is secured through cryptographic hash functions that link each block to its predecessor, creating a secure and unbreakable chain. Third, Transparency [3]: Every transaction on the blockchain is visible to all participants and is permanently recorded, making it nearly impossible to alter any aspect of the record without the network becoming aware. This level of transparency helps in building trust among users.

To ensure consistency and reliability, consensus algorithms are pivotal in maintaining the integrity and security of blockchains. They make decentralized decision-making possible, allowing networks to agree on the validity of transactions without needing a central authority. Various consensus mechanisms currently exist in the marketplace, including Proof of Work (PoW) [3], Proof of Stake [8], and Practical Byzantine Fault Tolerance [9]. In this project, we will focus on the Proof-of-Work mining strategy.

2.1.2 Proof of Work mining

The proof of work blockchain was proposed by Nakamoto in the paper 'Bitcoin: A Peer-to-Peer Electronic Cash System' [3]. From figure 2, we can see that a PoW block header contains the current and previous block hash, a Merkle root and a nonce value, while the block body holds a tree of transactions. Block hash in PoW is derived by applying a cryptographic hash function (SHA-256) to the block header. Including both the current and previous block hash in the header links the block to its predecessor in the blockchain, thereby creating an immutable chain. Altering any block retrospectively would require re-mining not only the altered block but all subsequent blocks due to the cascading change in hash values, thereby securing the blockchain against tampering. The Merkle root is a single hash that represents all of the transactions included in the block's body. More specifically, it is the root of a Merkle tree - a binary tree of hashes where the leaves are hashes of individual

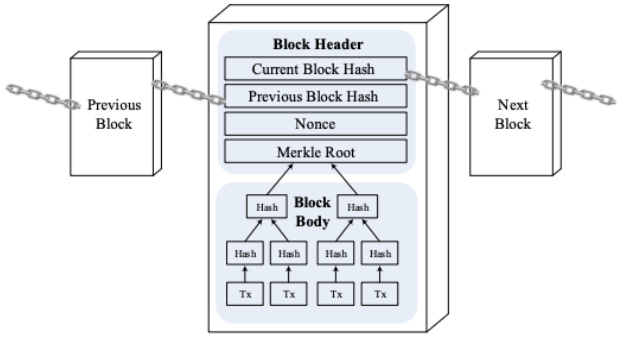


Figure 2. Data structure of a proof of work blockchain. Adopted from [10]

transactions and intermediate nodes are hashes of their respective child nodes. The Merkle root helps in efficiently and securely verifying the existence of transactions in a block.

To mine a block under PoW, miners must have to complete the following steps [3]:

1) Transaction Collection: Miners select transactions from a pool of pending transactions. Each transaction typically includes a fee which acts as an incentive for miners to include it in their block.
2) Transaction Verification: Miners check the validity of these transactions. This includes ensuring that the digital signatures are correct and that the transaction does not attempt to double-spend coins.
3) Creating a New Block: After selecting and verifying the transactions, miners begin the process of creating a new block. They compile the valid transactions into a block and calculate the Merkle Root, which is a single hash that represents all the transactions in the block.
4) Solving the Hash Problem: The core of PoW mining is solving the hash problem. This involves finding a nonce that, when combined with the hash of the previous block and the Merkle Root, produces a new hash that meets the network's difficulty target. More specifically, the hash value has to be below a certain target value and therefore start with a specific number of zeros. This difficulty target is dynamically adjusted by the blockchain network to ensure that the time to find a new block remains constant, despite fluctuations in the network's hashing power. This adjustment mechanism maintains the blockchain's security and efficiency, regardless of the total computational power dedicated to mining activities at any given time.

The first to solve the hash problem can then add the block to the blockchain and will be rewarded [3]. The reward typically includes two components: a) Block Reward: This is a predetermined amount of the cryptocurrency given to the miner. The size of the block reward is subject to a reduction mechanism known as "halving," which occurs at regular intervals (e.g., every four years in the case of Bitcoin) [3]. Halving systematically reduces the block reward by half, which controls the rate of new coin creation and contributes to the currency's scarcity and potential value appreciation over time. b) Transaction Fees: These are fees paid by users to have their transactions included in a block. Transaction fees are decided by the users themselves and can vary based on the network's congestion and the urgency of the transaction. As block rewards diminish over time due to halving, these transaction fees become increasingly significant as a source of income for miners. This reward system not only compensates miners for their computational power and energy costs but also secures the network by decentralizing the validation process.

2.1.3 Brute Force mining

According to the survey conducted by Wang et al. in 2019 [11], brute-force mining was recognized as one of the predominant mining approaches. Wang et al. described brute-force mining as miners iterating through an extensive range of nonce values in order to discover a valid solution to the hash puzzle. The study also revealed that brute-force mining demanded a significant amount of computational power. Early in the history of Bitcoin, miners could use ordinary personal computers for mining. But as the difficulty increased, more specialized hardware was developed. Today, ASIC miners are the standard for networks like Bitcoin because they offer the most efficient and powerful solution for brute force mining, providing a massive edge over older hardware like GPUs and CPUs [12].This intense computational requirement, however, has raised concerns about the environmental impact due to the significant energy consumption associated with continuous high-power operations.

2.1.4 Honest Mining

In honest mining, miners act in good faith by confirming and recording transactions correctly [13]. They contribute to the growth of the blockchain by adding new blocks to the network's longest-recognized chain. Miners solve complex cryptographic puzzles that secure the network and validate new transactions. Upon finding a new block, they immediately broadcast it to ensure it becomes part of the longest chain [13]. For their efforts, honest miners are rewarded with block rewards and transaction fees. These incentives not only compensate them for their computational work and energy expenses but also reinforce their commitment to maintaining the network's integrity and reliability.

2.1.5 Selfish Mining

Selfish mining is a strategic manipulation within the Bitcoin network where a miner or a group of miners discovers a new block but deliberately withholds this information from other participants in the network [2]. By not broadcasting the newly found block, selfish miners create a private fork of the blockchain on which they continue to mine secretly. According to Ittay. El This practice gives selfish miners several advantages [2]. Firstly, while other miners waste computational resources on an outdated chain, the selfish miners are already progressing on their private chain. If they can find subsequent blocks before the rest of the network catches up, they can create a longer chain in secret. According to Bitcoin's protocol [3], the longest chain is considered the legitimate one, so when selfish miners finally broadcast their version of the blockchain, it can potentially replace the previously accepted chain.

2.1.6 Machine Learning

Machine learning is a pivotal branch of artificial intelligence that empowers systems to learn from data and make decisions autonomously [14]. It encompasses various learning types, including supervised, unsupervised, and reinforcement learning, each suited to different applications. The ongoing advancements in machine learning are not only enhancing current technologies but are also paving the way for innovative solutions that can transform entire industries, such as finance, healthcare, retail, and more. Therefore, in this project, we would like to introduce machine learning to aid in blockchain mining. By optimizing and improving the efficiency of the mining process, we aim to decrease the overall power consumption and carbon footprint associated with PoW blockchain operations.

2.2 Literature Review

In this literature review section, we delve into the methodology and related works of machine learning in blockchain mining.

2.2.1 Regression

Regression analysis aims to understand how the typical value of a dependent (target) variable changes when any one of the independent variables is varied, while the other independent variables are held fixed [15]. It is mostly used for prediction and forecasting, where its use involves fitting a model to observed data in order to make informed predictions or decisions about new data from the same population.

The use of regression in blockchain has primarily centered on predicting cryptocurrency prices, and has achieved significant success [16, 17]. On the other hand, the application of regression in blockchain mining is relatively novel and has only been briefly explored. Simon et al [18] applied machine learning to analyze blockchain data on the Ethereum network and used linear and polynomial regression models to predict the next block rewards for miners. In 'Sustainable Optimizing Performance and Energy Efficiency in Proof of Work Blockchain: A Multilinear Regression Approach,' [19] the author employed regression to study the energy efficiency and computational expenditure of PoW blockchain mining. More specifically, multilinear regression analysis is utilized to explore the relationships between GPU performance indicators such as power consumption, thermal dynamics, core speed, and hash rate, revealing that strategic adjustments can significantly enhance energy efficiency and computational performance in PoW mining. However, these methodologies did not directly apply regression to aid the mining process, and as such, were unable to enhance mining efficiency.

2.2.1.1 Polynomial Regression

Polynomial regression is an extension of linear regression in which the relationship between the independent variable x and the dependent variable y is modeled as an n nth degree polynomial [20]. Unlike simple linear regression which models the response variable as a linear combination of the predictors, polynomial regression allows for a more flexible relationship involving higher powers of the input variables, thus enabling the model to capture the non-linear dependencies [21]. This flexibility makes polynomial regression particularly useful in cases where data exhibit curvature and more complex patterns that cannot be captured by a linear model alone. However, caution must be exercised to avoid overfitting, especially as the degree of the polynomial increases. The coefficients in polynomial regression are typically determined using the least squares method, aiming to minimize the sum of the squares differences between the observed and predicted values [22].

2.2.1.2 Random Forest Regression

Random Forest Regression is a powerful and versatile machine learning technique that uses ensemble learning methods for regression tasks. It constructs multiple decision trees during training and outputs the mean prediction of the individual trees to form a more accurate and stable prediction [23]. Unlike a single decision tree that might suffer from high variance or overfitting, a random forest mitigates these issues by averaging multiple trees that individually overfit to different aspects of the data. This approach not only improves predictive accuracy but also provides robustness against noise present in the training dataset.

A key advantage of random forest regression is its capability to handle large datasets with high dimensionality and multicollinearity among features without extensive pre-processing requirements. Additionally, random forests provide useful insights into feature importance, offering a natural form of feature selection [23].

2.2.2 Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning where an agent learns to make decisions by performing actions in an environment to maximize some notion of cumulative reward [24]. RL agents learn from the consequences of their actions through trial and error [24]. This paradigm is powerful in complex decision-making tasks where explicit programming of correct actions is impractical.

Reinforcement Learning has been applied in blockchain technology from various perspectives. A study by Mahatungade et al. [25] used RL to enhance the scalability of blockchain systems. It employed reinforcement learning to select miners for block mining, and their detailed analysis demonstrated this significantly improved the throughput of the blockchain network. 'Deep Bribe' [26] is another study that utilized deep reinforcement learning to analyse how a selfish miner might exploit petty-compliant miners to maximize revenue. They concluded that selfish miners will do so through bribery, by providing rational miners' rewards themselves, and this poses significant vulnerabilities in blockchain security. Related researches have also applied RL to examine miner behaviours and incentives [27, 28], and achieved significant discoveries and results. These works illustrate the diverse applications of RL in blockchain mining, thereby inspiring our use of it to enhance the profitability of blockchain mining.

2.2.2.1 Advantage Actor-Critic (A2C) algorithm

Advantage Actor-Critic (A2C) is a reinforcement learning technique that combines policy-based and value-based approaches through its actor-critic architecture [29]. The actor part of A2C determines actions based on the current policy, often represented by a neural network that outputs action probabilities. The critic evaluates these actions by computing a value function, another neural network that estimates the expected return from a given state. A2C utilizes the advantage function, calculated as $Q(s,a)-V(s)$, to measure how much better an action is compared to the average action at that state, which aids in reducing the variance of updates during training [20]. The critic updates its value predictions based on observed returns, while the actor adjusts its policy using gradients scaled by the advantage, promoting actions with above-average returns. A2C improves upon previous actor-critic methods by updating models synchronously, enhancing stability and reducing variance in updates, making it suitable for complex applications, such as blockchain mining.

2.2.3 Markov Decision Processes

Markov Decision Processes (MDPs) provide a mathematical framework for modelling decision-making situations where outcomes are partly random and partly under the control of a decision-maker [30]. MDPs are particularly useful in environments that require a series of decisions over time, making them suitable for applications in blockchain mining, where miners must make continuous decisions regarding which blocks to mine and when to publish them.

An MDP is characterized by four primary components [30]:

- States (S): These are the various conditions or configurations that the system or environment can assume. Each state encapsulates all the relevant information that the decision-maker needs to know in order to make decisions.
- Actions (A): For each state, there are typically several actions available to the decision-maker. Actions represent the different choices or decisions that can be made by the decision-maker from a particular state.
- Transition Probabilities (P): For each state and action, the transition probabilities determine the probability of moving to any other state. This is typically denoted as $P(s'|s,a)$, which is the probability of transitioning to state s' from state s after taking action a.
- Rewards (R): A reward function quantifies the immediate payoff resulting from taking a particular action in a specific state. Often denoted as $R(s,a,s')$. It represents the reward received after transitioning from state s to state s' via action a.

In the paper "Optimal Selfish Mining Strategies in Bitcoin," [31] the authors introduced the use of Markov decision process (MDP) to model the blockchain mining process from the miner/adversary perspective. They found that solving the MDP could lead to much better mining profits compared to traditional honest mining and selfish mining. This forms the basis of our methodology and has inspired us to explore various MDP solvers, as well as states and actions spaces to discover profitable mining strategies through MDP.

2.2.3.1 Undiscounted MDP: Relative Value Iteration

In undiscounted MDPs, the goal is to maximize the cumulative reward with no consideration for the time value of rewards. This model is particularly relevant in environments where the immediate reward is as significant as any future reward [32].

Relative Value Iteration is a method used in undiscounted MDPs to find an optimal policy that maximizes the long-term rewards. This algorithm iteratively updates the value of each state under the assumption that the value eventually converges to the optimal. The relative value iteration helps determine the best action to take in each state by comparing the values of possible future states, adjusted by their respective probabilities [33].

2.2.3.2 Discounted MDPs: Q-Learning, Value Iteration, and Policy Iteration

In many real-world situations, future rewards are considered less valuable than immediate rewards, modeled using discounted MDPs where rewards are discounted by a factor γ ($0 \leq \gamma < 1$) each step into the future [32]:

- Q-Learning: Q-Learning is a model-free reinforcement learning algorithm that does not require a model of the environment and learns the quality of actions directly from experiences of state-action pairs [34]. It updates the action-value function (Q-value) based on the reward received after an action is taken, making it ideal for environments like blockchain mining where the system dynamics may not be fully known.
- Value Iteration: This algorithm operates on the principle of iteratively refining the values assigned to each state in a system to converge on the optimal decision-making strategy [35]. In each iteration, Value Iteration computes the expected utility of undertaking each possible action from a given state and then updates the state's value to reflect the highest expected utility [35]. This method ensures that, over time, the

values of states accurately represent the best possible outcomes achievable from those states, guiding decision-makers toward the most beneficial actions. Thus, it is particularly applicable to blockchain mining, where strategic decision-making directly influences mining efficiency and resource management.

- Policy Iteration: Comprising two main steps—policy evaluation and policy improvement—Policy Iteration starts with an arbitrary policy and improves it iteratively [35]. During policy evaluation, the value of each state under the current policy is computed. In the policy improvement step, a new policy is formulated by choosing actions that lead to the highest-value states [35]. This iterative refining of policies makes Policy Iteration suitable for blockchain mining, where miners need to continually adapt their strategies to the changing dynamics of the network.

3. Methodology

In this section, we delve into the methodology of the project. This includes historical data collection, enhanced nonce finding, transaction selection optimization with reinforcement learning, and optimal chain-level mining strategy with a Markov Decision Process.

3.1 Historical Data Collection

The blockchain.com API was utilized to systematically collect data on Bitcoin transactions for our model training and evaluation. The data collection process involved retrieving detailed information from 1,000 sequential Bitcoin blocks. The fields included Hash, Version, Previous Block Hash, Merkle Root, Timestamp, and Nonce.

hash	version	previous_block_hash	merkle_root	timestamp	nonce
00000000000000000003cc7946007080c290b98c5cb856f7ab4749d13d4de040	571850752	00000000000000000002e7bd70441065668ad23bff065a76c6cd6ec1c6f93eff	6ff27b57400fdd0e0e595e2771e02fe58cbfb3b37615e2982c3515b6df22dc3c	1704738256	1972851776
00000000000000000002e7bd70441065668ad23bff065a76c6cd6ec1c6f93eff	857014272	00000000000000000003 1bce81404601803 7f1d6f1d4cb67bb1437aa31de31fb	1604ecc1fb48ef11acc21c7a37f2e3ba6e2a86aa13f2b8093473e9ef7a721ede	1704737187	1041145614
00000000000000000031bce81404601803 7f1d6f1d4cb67bb1437aa31de31fb	859897856	00000000000000000000382c51d6d6f14cc1291967c052200012871c096092a40	e18258dd8a7d936a7aef3613163a6f7ee5a731fe952a7f83823247875aab066a	1704736636	3012228387

Figure 3 Data fields of Bitcoin block data extracted

For each block, we further extracted data for 20 transactions, totalling 20,000 transactions. The fields for the transactions included Transaction ID, fee, and weight. This dataset provides a foundation for our experiments to derive insights into the effectiveness of applying machine learning in blockchain mining.

tx_id	fee	weight
17811f6fcd067b9d5b3ba523401f5568a81f9700c30ae88aa7b3e34a4509d9f3	14014	724
c9b6d39405311c5cbbd0cb1b12e8e2b7d9f99802f6e39d11fff355d3bf3137fe	20982	2152
b72088c627defcf5306cf9d1861ac97fa9b59b3141d402f8dc5b2f11f8911571	2010	670
531516ccbd627b848726674329321b734ed6df3ac7d3acca93965af2b8648ed	8060	617
5266a1902dda8b9922f57b17c7498f233b154b341ef991492c16d3d6a4789cf7	31920	892
f7bfeeaeae421336f98077892c3dbaa1bd758d4adba2b30f05b90d87b9f8a3f3	12757	712
edf26653263f42bc88706d2c1493460344efcea27ea08cda679feb4d3977b4d2	8052	730
3644d9d384c174eeb3855a3fa3da59c3b2417889c9b6bab76f4112fc505b873f	26514	1960
c0a0e8849e4048ae0213104e90ee6a45fbffaa2597e365285bb4eba9fe498ee4	88536	3916
d821778d01f328d31f312b772d49b899ecea4a2e0621e8854614bd65940a7af9	35938	603
e4ebe048d5983436bb847d9a8f172c8bd0cd4b5693b14eb1a18737b3811f299b	10745	983

Figure 4 Data fields of Bitcoin transactions attained

3.2 Enhanced Nonce Finding

To successfully mine blocks, miners must discover a nonce that produces a block hash meeting the difficulty requirement. Here, we apply various methods to find the correct nonce value with a fewer number of tries, thereby reducing the computational overhead of blockchain mining.

3.2.1 Iteration Methods

Existing miners have to iterate through an immense number of nonce values to find the correct solution. Here, different iteration methods were explored to find the most efficient way of traversing through nonce values. We compared the traditional approaches, which involve incrementing the nonce sequentially by one, with an alternative approach of

incrementing by 10, as well as applying the Collatz conjecture to iterate nonce values in a nonlinear fashion.

3.2.2 Regression to Predict Starting Nonce

To reduce the number of attempts needed to find the correct nonce value, we employed regression models to predict the starting point of nonce iteration. Our models were trained using historical data from previously mined Bitcoin blocks. Inputs to these models included the Merkle root (converted to integer format), the timestamp, and the previous block hash, all normalized before passing into the model. The output from the model is a suggested starting nonce value, ideally as close to the correct nonce value as possible. Then, we will apply the most effective iteration method found in Section 3.2.1 to iterate through different nonce value. We utilized advanced regression models such as Polynomial Regression and Random Forest Regression, alongside an ensemble method that averages the outputs from both models. These approaches were selected for their ability to capture complex patterns and relationships within the data, which is crucial for accurately predicting starting nonce values in block mining.

3.2.3 Custom Difficulty Formula

Mining under the official Bitcoin difficulty formula requires extensive computational resources and is not feasible for our experiments. Therefore, we adopted a custom difficulty formula that was more suitable for our low-scale experiments while ensuring the validity of our results. Figure 3 displays the code of the formula.

```
def produce_hash(block_body):
    h = hashlib.sha256()
    encoded = json.dumps(block_body, sort_keys=True).encode('UTF-8')
    h.update(encoded)
    return h.hexdigest()

def puzzle_solution_is_correct(solution, difficulty):
    integer_solution = int(solution, 16)
    diff = 2 ** (256 - difficulty)
    correct = integer_solution <= diff
    return correct
```

Figure 5 Custom difficulty formula

Similar to the official Bitcoin formula, once we have the block body, we pass it to the produce_hash function and apply SHA-256 to obtain the hash value. However, meeting the actual difficulty requirement is challenging due to our resource constraints. Therefore, we passed the hash value obtained to the puzzle_solution_is_correct function, which is a simplified difficulty setting to verify whether a given solution meets the determined difficulty level. Our method converts the solution, which is a hash string, to an integer and compares it against a threshold determined by the difficulty parameter. This threshold is calculated as 2^(256−difficulty), and the produced hash should have a value equal or below it. This makes the verification process simpler, while is still an efficient and effective method for validating the computational work done in a PoW blockchain.

3.3 Transaction Selection Optimization with Reinforcement Learning

In blockchain mining, miners are tasked with selecting the most profitable transactions from the transaction pool while adhering to the block's weight limit to maximize their gains. The transaction pool typically contains a large number of transactions, each varying in size, fee, and complexity. Therefore, traditional selection methods may not be able to efficiently handle the dynamic and multifaceted nature of transaction pools. In this section, we developed a reinforcement learning algorithm to optimize the selection of transactions, aiming to maximize not only gain but also efficiency in constructing the optimal block

3.3.1 Naive Methods

Two fundamental approaches to transaction selection for blockchain mining were first explored. They serve as a baseline for comparing more advanced transaction selection algorithms.

The "Random" method involves indiscriminately selecting transactions from the pool until the block reaches its weight capacity. This approach does not consider the fee or weight of transactions; instead, it relies purely on chance to populate the block. While this method is simple and requires minimal computation, it generally results in suboptimal block rewards since it does not strategically maximize the fees collected per block.

The 'Sorted' method enhances the selection process by strategically organizing transactions before inclusion. Transactions are first sorted in descending order by fee and then in ascending order by weight. This method aims to optimize the total reward garnered from the block before reaching the weight limit. It represents a basic yet more effective approach compared to random selection, as it considers both the economic and spatial efficiencies of transaction inclusion.

3.3.2 Action Space

In reinforcement learning, the action space defines the set of all possible actions that the agent can take at any given state. In our reinforcement algorithm, the action space is defined by the binary decision to either 'select' or 'not select' a transaction for inclusion in the upcoming block.

This binary action space, while seemingly simple, leads to a complex decision-making environment due to the combinatorial nature of block construction and the interdependencies between transactions. The reinforcement learning agent must learn to navigate this space effectively to optimize block composition for maximum profitability and adherence to blockchain protocols.

3.3.3 Reward Policy

The reward policy is a critical part of the reinforcement learning (RL) algorithm as it directly influences the agent's learning process by specifying the feedback it receives for its actions. In our RL agent, we experimented with two different reward policy criteria: Running Average of Total Aggregated Fees and the Fee-to-Weight Ratio.

3.3.1.1 Running Average of Total Aggregated Fees (RATAF)

Our initial reward policy was centered around the running average of the total aggregated fees. RATAF is defined as the total accumulated fee divided by the total number of transactions currently in the block.

$$RATAF = \frac{Total\ Fee\ (in\ Satoshis)}{Total\ Number\ of\ Transactions}$$

Equation 1. Formula for Running Average of Total Aggregated Fees (RATAF)

A positive reward is granted whenever the inclusion of a transaction increases the Running Average of Total Aggregated Fees (+1). Conversely, a negative reward is assigned (-1) if the inclusion of a transaction causes a decrease in this running average. This approach aims to help the RL agent include transactions that could maximize the profitability of each block.

Yet, we realised this reward policy did not consider the weight of each transaction, leading to inefficient block utilization. To address this, we refined our approach by implementing the Fee-to-Weight ratio reward policy.

3.3.1.2 Fee-to-Weight Ratio

Transactions not only carry fees but also have associated weights, which are indicative of their sizes or resource requirements. To better optimize the selection of transactions for inclusion in a block, we redefine the reward policy with reference to the fee-to-weight ratio. This Fee-to-Weight ratio is defined as the total transaction fee over the transaction weight of all transactions in a block.

$$\text{Fee-to-Weight ratio} = \frac{Transaction\ Fee\ (in\ Satoshis)}{Transaction\ Weight\ (in\ weight\ units)}$$

Equation 2. Formula for the Fee-to-Weight ratio

Furthermore, a dynamic reward system was implemented to scale reward according to changes in the Fee-to-Weight Ratio. More specifically, a positive reward is given when a transaction that increases the ratio is selected, or when a transaction that would decrease the ratio is excluded. Conversely, a negative reward is assigned when a transaction reduces the fee-to-weight ratio, or when a transaction that would improve the ratio is excluded. We have further doubled the reward for correctly including a beneficial transaction and for excluding a detrimental transaction. This adjustment aims to incentivize the RL agents to focus on ensuring that the block created maximizes the total possible rewards.

3.3.1.3 Mining Pool Exploration Penalty

Throughout model training, it was identified that the RL agents may continuously exclude transactions for long periods of time in the hopes that better options will appear later. However, this behavior often leads to extended periods of inactivity, which in turn reduces the overall efficiency of the mining operation. To counteract this issue, we implemented a penalty mechanism aimed at discouraging prolonged inactivity. The penalty is defined as the number of transactions explored over the total number of transactions in the transaction pool, multiplied by 1000. Multiplying the ratio by 1000 ensures the penalty is significant enough to

influence decision making. The penalty will finally be deducted from the total reward earned by the agent.

$$Mining\ pool\ exploration\ penalty = \frac{Number\ of\ transactions\ explored}{Total\ number\ of\ avaliable\ transactions} * 1000$$

Equation 3. Formula for Mining pool exploration penalty

By imposing this penalty, we aim to promote a more dynamic and continuous transaction processing environment, ensuring that the RL agent maintains efficiency and effectiveness in its operations.

3.3.3 State Space

In our RL algorithm, the state space is defined by three key components that help determine the most efficient transactions to include in the mining block: (Cur_ratio, Tx_ratio, Prop)

- Cur_ratio (Current Fee-to-Weight Ratio): This represents the fee-to-weight ratio of all transactions currently included in the block. It provides a baseline measure of the block's overall efficiency, allowing for reference with potential new transactions.
- Tx_ratio (Transaction Fee-to-Weight Ratio): This is the fee-to-weight ratio of the transaction that is currently being considered for inclusion in the block. This metric is crucial for decision-making, as it directly compares the potential value of a new transaction against those already included.
- Prop (Proportion of Transaction Pool Explored): This metric indicates the percentage of the total transaction pool that has been explored or processed so far. It is an important factor for assessing the thoroughness of the transaction selection process and helps in determining whether further exploration could yield more efficient transactions. It also ensures an efficient mining process.

By monitoring these three factors, the RL agent can make informed decisions that optimize the Fee-to-Weight ratio of their blocks, thereby maximizing profitability while ensuring efficient use of block space. This state space framework supports dynamic decision-making in real-time, allowing for adjustments based on ongoing assessments of transaction value and pool exploration progress.

3.3.4 Policy Learning

In our exploration of different policy learning methods for optimizing transaction selection in blockchain mining, we found that the Advantage Actor-Critic (A2C) method performed the best. The algorithm was able to effectively balance exploration of new transaction selection strategies with the exploitation of known profitable strategies, thereby performing superiorly in maximizing block rewards while adhering to block weight constraints.

3.4 Optimal Chain-level Mining Strategy with Markov Decision Process

Currently, there are various mining strategies [21] that achieve improved overall outcomes compared to traditional honest mining [19]. Therefore, we model blockchain mining as a Markov Decision Process (MDP) and apply different solvers to attain the optimal strategy. The block miner acts as an adversary and undertake different actions against the honest miner to maximize its own gain.

3.4.1 Blockchain Model Simulation

To effectively simulate the blockchain environment, we first establish the model parameters that define the dynamics and interactions within the network:

- α: This parameter represents the fraction of the network's computing power controlled by the adversary. Conversely, the honest miners control the remaining fraction, represented as $1-\alpha$.
- γ: This parameter indicates the proportion of miners in the network that would receive a block from the adversary first when both the adversary and an honest miner release their blocks simultaneously.
- $\gamma(1 - \alpha)$: This expression quantifies the computing power of the network that will mine on the adversary's blocks when both the adversary and honest miners release their blocks at the same time. This is a crucial metric as it reflects the influence of the adversary's blocks on the network under simultaneous release conditions.
- $\alpha / (1- \alpha)$: This represents the upper bound of the relative revenue that can be earned by the adversary, as defined in [31].

3.4.2 Action Space

In our blockchain model simulation, we define the following actions that an adversary/miner can take. Each action represents a strategic decision in response to the current state of the blockchain:

- Adopt: The miner adopts the current longest chain in the network, which is typically controlled by the honest miners. By mining on top of the latest block of this chain, the miner aligns with the network consensus and contributes to its growth.
- Override: The miner attempts to gain a strategic advantage by releasing a chain that is one block longer than the currently accepted honest chain. This effectively makes the miner's chain the new longest chain, compelling the network to accept this new chain.
- Match: This action creates a scenario of uncertainty and competition. The miner publishes a number of blocks that match the length of the current honest chain, resulting in a blockchain fork. This initiates a direct competition between two chains: one maintained by the honest network and the other by the adversarial miner.
- Wait: The miner chooses to delay any public action by not publishing any blocks. Instead, the miner continues to mine in private, extending a secret chain with the potential intent to later execute an override or match action. This strategy can build up a significant advantage that could be leveraged in future moves.

3.4.3 State Space

The state space, denoted S, is a crucial component of our Markov Decision Process model for blockchain mining. It is defined using three parameters: a, h, and $fork$, where:

- a: Represents the chain length of the adversary. This indicates the number of blocks in the chain that the adversary has been able to confirm.
- h: Represents the chain length of the honest miners. It indicates the number of blocks in the chain that the honest network has confirmed.
- fork: Describes the current status of the blockchain in terms of chain splits. It can take one of three values:
    - Irrelevant: This status indicates that the latest block was mined by the adversary, and the blocks published by the honest network have already been accepted by the majority of the network. In this state, the action "Match" is not permitted because there is no need for the network to choose between two competing chains.
    - Relevant: This status is assigned when the latest block is mined by the honest network. If *fork* is set to "Relevant" and l(a)≥l(h), the action "Match" is allowed. This enables the adversary to introduce a competing chain of equal length, potentially leading to a fork if the network is undecided on which chain to continue.
    - Active: This value indicates that the adversary has previously executed the "Match" action, resulting in a blockchain split into two competing branches. The network is currently has a fork and nodes have to decide on which chain to extend.

Each state in S provides a snapshot of the current blockchain conditions, informing the strategic decisions available to both honest and adversarial miners. This state representation is fundamental for simulating and analysing different strategies under various blockchain network conditions.

3.4.4 MDP solvers

Various discounted and undiscounted MDP solvers were applied to solve for the optimal strategy. Discounted MDP techniques, including Value Iteration, Policy Iteration, and Q-Learning, were applied with the discount factor, γ finetuned. We also leveraged an undiscounted MDP approach: Relative Value Iteration, which views immediate and future rewards as equally valuable. By leveraging these methodologies, we aim to comprehensively analyse and derive effective mining strategies under various network conditions and adversary capabilities.

4 Experiments and Results

This section covers the various experiments conducted to examine the effectiveness of the proposed methods. Section 4.1 covers experiments and results on enhancing nonce finding, Section 4.2 covers the use of reinforcement learning in optimizing transaction fees, and Section 4.3 discusses the effectiveness of deriving optimal chain-level mining strategies with Markov decision processes.

4.1 Enhance Nonce Finding

In this experiment, we compared the average number of attempts needed to find the correct nonce for different iteration methods and starting seed. 800 Bitcoin blocks were used for training the regression models and 200 Bitcoin blocks were used for evaluation. The difficulty value was set to 20 in the custom difficulty formula as shown in Figure 3.

4.1.1 Iteration Methods

Table 1 displays the average number of nonces tested for different iteration methods over the mining of 200 Bitcoin blocks. It is evident that the traditional method of sequential increment by 1 performs the best, requiring only 1,082,461 attempts to attain a correct nonce, followed by sequential increment by 10 with 1,122,047 attempts, and the Collatz conjecture with 1,409,675 attempts.

Method	**Average Number of Attempts**
Sequential (+1)	1,082,461
Sequential (+10)	1,122,047
Collatz conjecture	1,409,675

Table 1. Average number of nonces tested for different iteration methods

Figure 6 displays the number of nonces tested per block using different iteration methods across 10 random blocks. The results show that there are cases in which the Collatz conjecture performed significantly worse compared to linear methods. More specifically, for block 7, the Collatz conjecture required around 10 times more than the number of attempts needed by sequential methods to attain correct nonce values. This indicates that using a non-linear iteration method may not be ideal, as it could overlook many suitable nonce values. Furthermore, we can also see that iterating sequentially with too large an interval may lead to undesirable results, as observed in block 0 and block 8. When iterating sequentially by 1, all blocks required fewer than 2.0 x 1e6 attempts. Therefore, it is believed that the traditional way of iterating through nonce values is the most effective.

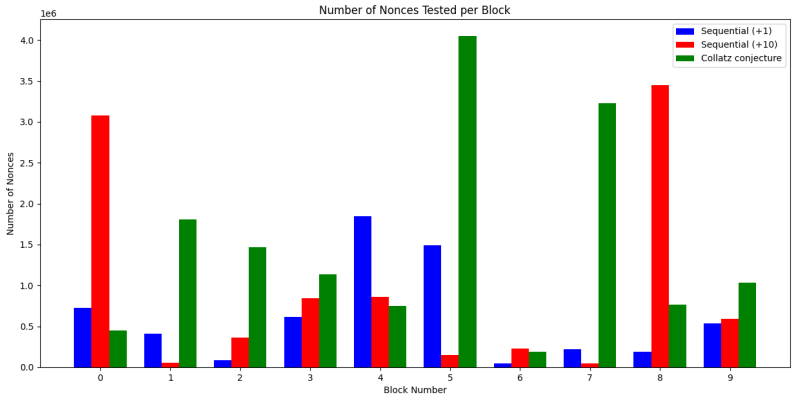


Figure 6. Number of nonces tested per block for different iteration methods

4.1.2 Starting Seed

In this part, we explore the use of machine learning to determine the starting seed for nonce searching. Section 4.1.1 demonstrates that iterating the nonce value by sequential increments of 1 is optimal; thus, upon establishing the starting seed, we continue with this approach to iterate through nonce values.

Table 2 displays the average number of nonces tested for different methods of finding the starting seeds over 200 blocks. The results show that, on average, all methods required a roughly similar number of tries to mine all blocks.

Method	**Average Number of Attempts**
Sequential (+1)	1,082,461
Polynomial Regression	1,163,762
Random Forest Regression	1,079,642
Ensemble Method	952,824

Table 2. Average number of nonces tested for different starting seed

Yet, a more detailed examination of the individual mining results offers further insights. In Figure 7, we visualize the number of attempts required for both sequential and regression methods to attain the correct nonce value across 10 randomly selected blocks. Notably, in 8 out of these 10 blocks, at least one of the regression methods outperformed the traditional sequential method, which starts mining from zero. Furthermore, for blocks 3 and 9, using random forest regression and polynomial regression respectively, were able to predict a starting seed that was very close to the correct nonce value. This suggests that regression methods could bring significant advantages in mining in various instances.

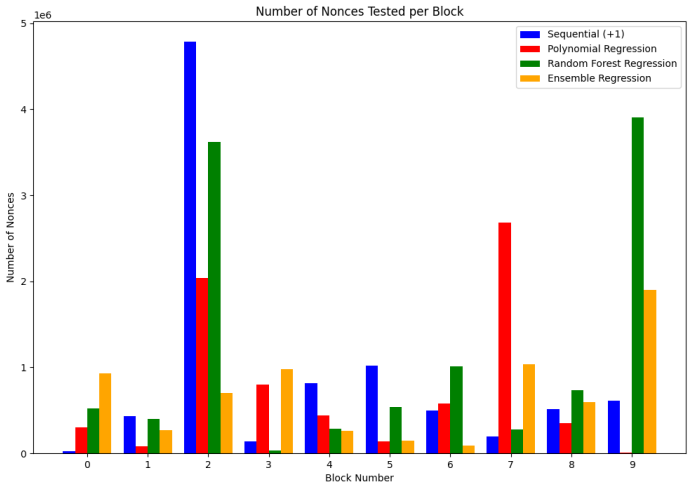


Figure 7. Number of nonces tested per block for different starting seed

In the actual mining competition, the winner is the one who mines the block first. In Table 3, we display the number of wins out of 200 blocks for the traditional method as well as for using regression to find the starting seed for nonce iteration.

Method	**Number of wins**
Sequential (+1)	50
Polynomial Regression	42
Random Forest Regression	52
Ensemble Method	56

Table 3. Number of wins for different methods in predicting the starting nonce

From the results, we can see that Ensemble regression produced the most wins, with a total of 56. Additionally, all regression methods were able to achieve more than 40 wins, which shows that there are various instances where finding the starting seed with regression is highly effective. This underscores the potential of regression techniques in enhancing the efficiency and success rate of mining operations.

4.2 Mining Fee Optimization

In this experiment, we compare our proposed reinforcement learning algorithm with the two naive methods 'Random' and 'Sorted', as defined in Section 3.3.1 in maximizing the total fee accrued over the course of mining 15 blocks. We consider a dataset comprising 20,000 transactions, each characterized by its unique Transaction ID, associated fee, and weight. We perform our experiments in two scenarios: one with a weight limit of 250,000 and the other with 500,000.

First, we analyze our transaction dataset. From Figure 8, we can see that the transaction pool distribution is complex. There are 'good' transactions, where the weight is low and the fee is high, and 'bad' transactions, where the weight is high, and the fee is low, as well as 'average' transactions. Therefore, these transactions should be carefully managed to maximize rewards.

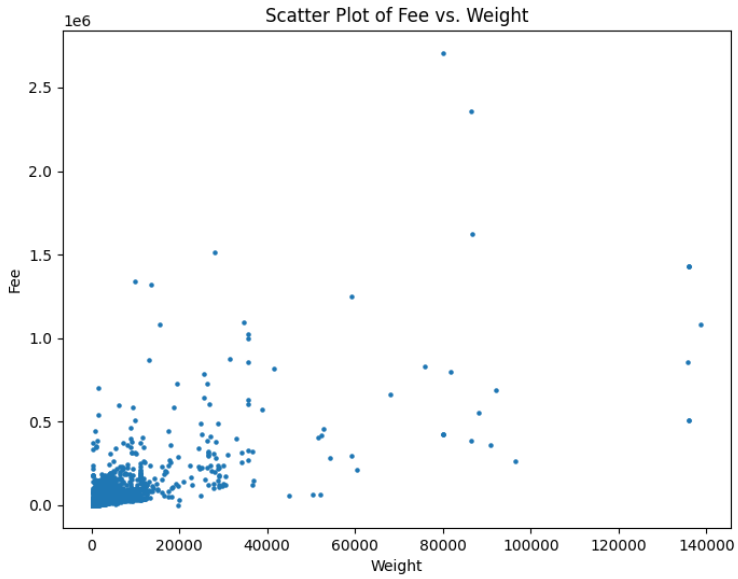


Figure 8. Relationship between fee and weight for the transaction dataset

Then, we display the total mining rewards of different transaction selection methods over 15 blocks in Table 4.

Random	Sorted	RL (Average Reward)	RL (Fee-to-Weight ratio)
53,790,148	62,399,034	72,593,261	86,625,949

Weight Limit: 250,000

Random	Sorted	RL (Average Reward)	RL (Fee-to-Weight ratio)
112,829,601	124,762,115	127,867,014	147,731,187

Weight Limit: 500,000

Table 4. Total mining reward for different transaction selection methods

From the results, it’s clear that the total mining rewards are lowest for 'Random', followed by 'Sorted', while the rewards produced by the RL algorithms outperform both naive methods. Furthermore, the new reward policy based on the Fee-to-Weight ratio outperformed the previous one based solely on average fees by near 20% and more than 15% in weight limit 250,000 and 500,000 respectively. Therefore, this reflects the Fee-to-Weight reward policy brought significant improvements in rewards earned by miners.

Figure 9 displays the mining reward per block for various transaction selection methods with block weight limit of 250,000 and 500,000.

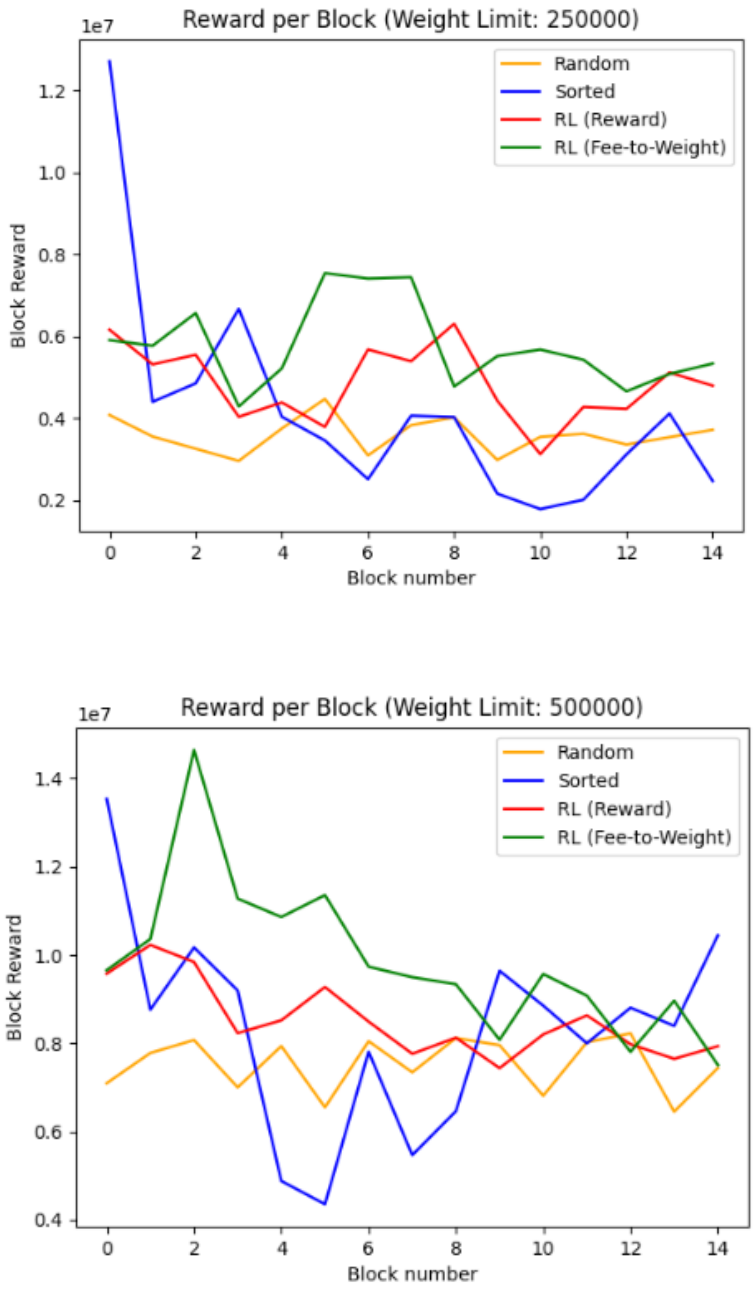


Figure 10. Mining reward per block for different transaction selection methods

From the results, it is evident that the 'Random' strategy, which selected transactions randomly, was not able to leverage the characteristics of different transactions in the pool, resulting in suboptimal outcomes. The 'Sorted' strategy selected the best transactions first, yet this led to a significant performance degradation over time. In contrast, both reinforcement learning algorithms are capable of attaining a more balanced reward over 15 blocks, leading to a higher cumulative mining reward at the end. Using the Fee-to-Weight ratio reward policy further improved block profits as it evaluates a transaction based on not only the fee but also on the weight. Therefore, applying reinforcement learning with the Fee-to-Weight ratio reward policy is effective in constructing optimal blocks and maximizing rewards over time.

4.3 Optimal Mining Strategy with Markov Decision Process

To validate the effectiveness of the MDP mining strategy, we conducted experiments on a blockchain simulation to compare the performance of the solved MDP mining strategy against honest mining and selfish mining strategies under various α and γ parameter values as defined in Section 3.4.1. The maximum fork length allowed is 8. In this simulation, the primary goal of the adversary/miner is to maximize relative revenue, which is defined as the revenue of the adversary divided by the total revenue earned by both the adversary and honest miners.

$$Relative\ Revenue = \frac{\sum_{i \in A} R_i}{\sum_{i \in A} R_i + \sum_{j \in H} R_j}$$

Equation 4. Formula for relative revenue

4.3.1 Selfish Mining vs Honest Mining

The performance of Selfish Mining was first compared with Honest Mining in the defined blockchain environment under different α and γ values.

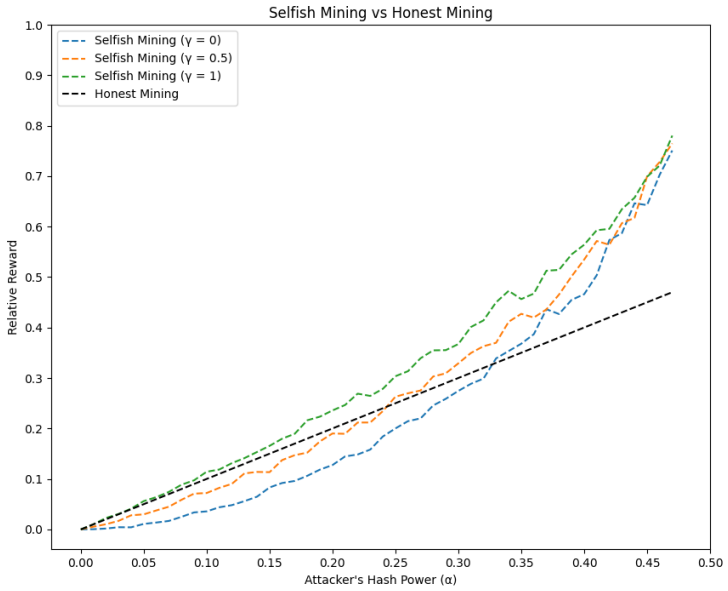


Figure 11. Performance comparison between selfish mining and honest mining

From the result, it is clear that the performance of selfish mining improves with increased adversary hash power. This is due to its enhanced control over the blockchain. More specifically, higher computational power enables a selfish miner to withhold blocks with greater efficiency and potentially create a longer private chain. This enhances the likelihood of invalidating honest miners' blocks and earning extensive rewards when the adversary eventually broadcasts the chain. Additionally, when α exceeded 0.35, selfish mining outperformed honest mining across all γ values (0, 0.5, and 1). As γ increased, the effectiveness of selfish mining also enhanced. This shows that when a selfish miner has favourable hash power and communication factors, he can heavily outperform traditional honest mining.

4.3.2 Comparing different MDP solvers

We formulated the blockchain mining process as an Markov Decision Process. Here, we solved the problem using different MDP solvers and compared their performances under varying α with γ fixed at 0.5.

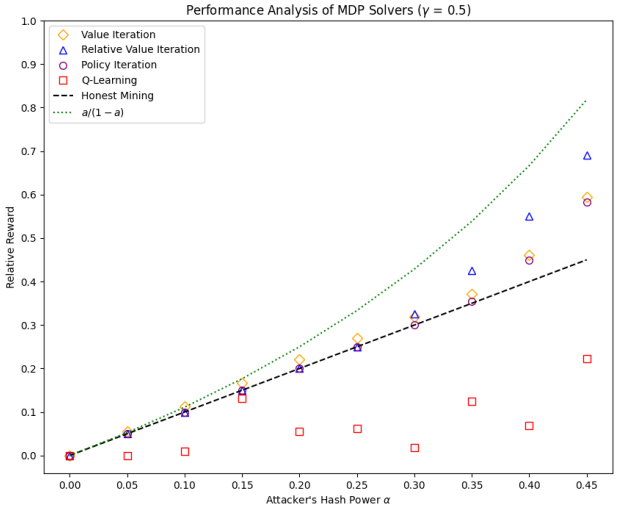
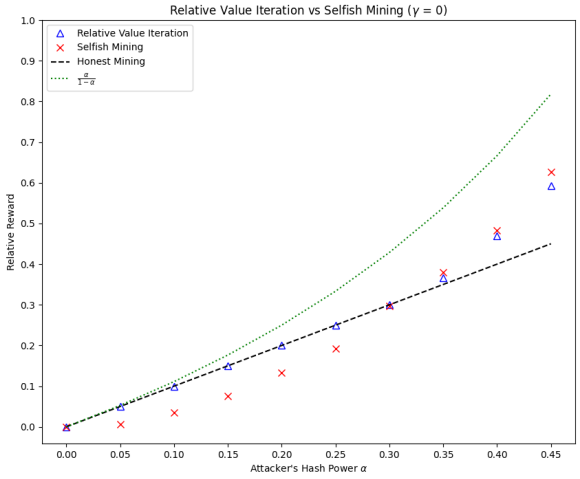


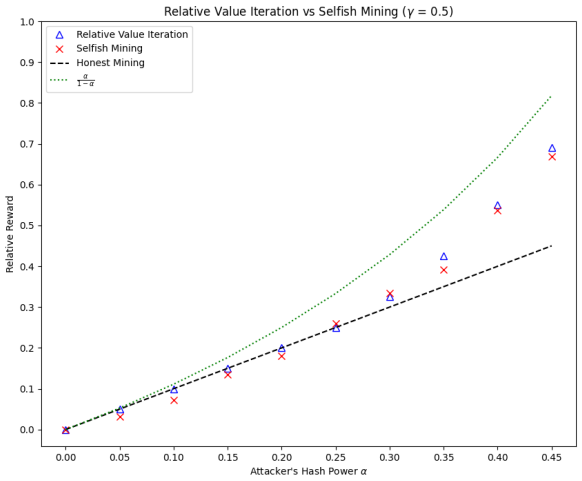
Figure 12 Performance comparison between mining polices solved with different MDP solvers

From the result, it is evident that Relative Value Iteration performed the best, followed by Value Iteration and Policy Iteration, and finally, Q-learning, which attained a suboptimal outcome. This reflects that Undiscounted MDP solvers (Relative Value Iteration) perform better than Discounted MDP solvers (Value Iteration, Policy Iteration, and Q-learning) in the task because they treat immediate and future rewards of the same importance. This therefore allows the policy to better prioritize short-term gains without devaluing long-term benefits, ensuring a more balanced approach to decision-making across all states.

4.3.3 Comparing MDP mining strategy with Selfish Mining

In Section 4.3.2, we determined that Relative Value Iteration solved for the best mining policy. Here, we compare its performance with traditional selfish and honest mining under different α and γ values.





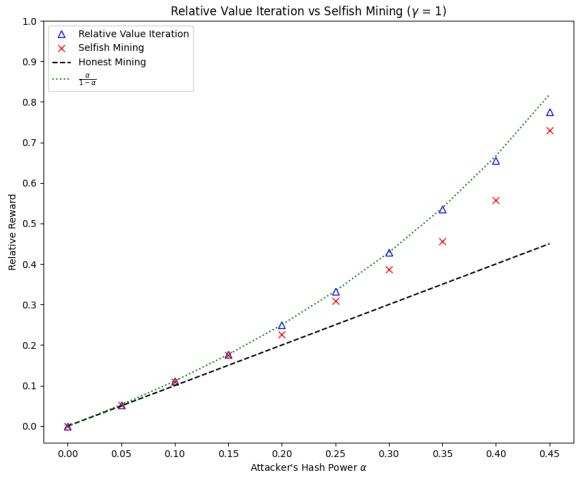


Figure 13. Comparing Optimal Mining Policy solved by Relative Value Iteration with Selfish Mining and Honest Mining

From the results, we observe that for γ = 0, Relative Value Iteration performed better than selfish mining when α ranges from 0 to 0.3, but its performance is worse when α is larger or equal to 0.35. When γ is set at 0.5 and 1, both the MDP mining strategy outperformed selfish mining under the majority of α values. Furthermore, the improvement gap of the MDP strategy compared to selfish mining is larger when γ is 1 than when γ is 0.5, indicating that the increase in γ significantly enhances the strategy's effectiveness. This reflects the importance of effective communication in the MDP-derived mining strategy to achieve optimal and best performance. Better network connectivity allows for quicker block propagation, therefore blocks solved by the MDP strategy are more likely to be accepted and built upon by other nodes in the network, and this significantly increases the miner's reward.

It is also noted in the figure that when γ = 1, the derived mining strategy nearly reaches the theoretical upper bound of α / (1 – α). This reflects that the performance of the solved mining strategy is highly optimized and effective.

4.3.4 Optimal Mining Strategy Policy

We further explore the policy derived from solving the blockchain MDP with Relative Value Iteration, using parameters α = 0.3, γ = 0.5, and a maximum fork length of 8, as demonstrated in the below figure. This visualization outlines the actions taken by the policy across different chain lengths of both the adversary/miner chain and the honest chain. The actions include: 0 for 'Adopt' (accepting the honest chain), 1 for 'Override' (publishing an additional block to the network), 2 for 'Match' (publishing a conflicting block to create a fork), 3 for 'Wait' (continuing to mine on one's own chain), and * for 'Unattainable State'. The figure also details the policy's responses under the three states: [Irrelevant, Relevant, Active].

output

l(a)/l(h)	0	1	2	3	4	5	6	7	8
0	[*, *, *]	[0, 0, *]	[*, *, *]	[*, *, *]	[*, *, *]	[*, *, *]	[*, *, *]	[*, *, *]	[*, *, *]
1	[3, *, *]	[*, 2, *]	[*, 0, *]	[*, *, *]	[*, *, *]	[*, *, *]	[*, *, *]	[*, *, *]	[*, *, *]
2	[3, *, *]	[1, 1, 1]	[*, *, *]	[*, 3, *]	[*, 0, *]	[*, *, *]	[*, *, *]	[*, *, *]	[*, *, *]
3	[3, *, *]	[*, 2, *]	[1, 1, 1]	[3, *, *]	[*, 3, *]	[*, 0, *]	[*, *, *]	[*, *, *]	[*, *, *]
4	[3, *, *]	[3, 2, 3]	[*, 2, *]	[1, 1, 1]	[3, *, *]	[*, 3, *]	[*, 0, *]	[*, *, *]	[*, *, *]
5	[3, *, *]	[3, 2, 3]	[3, 2, 3]	[*, 2, *]	[1, 1, 1]	[3, *, *]	[*, 3, *]	[*, 0, *]	[*, *, *]
6	[3, *, *]	[3, 2, 3]	[3, 2, 3]	[3, 2, 3]	[*, 2, *]	[1, 1, 1]	[3, *, *]	[*, 3, *]	[*, 0, *]
7	[3, *, *]	[3, 2, 3]	[3, 2, 3]	[3, 2, 3]	[3, 2, 3]	[*, 2, *]	[1, 1, 1]	[3, *, *]	[*, 0, *]
8	[1, *, *]	[1, *, 1]	[1, *, 1]	[1, *, 1]	[1, *, 1]	[1, *, 1]	[*, *, *]	[1, *, 1]	[*, *, *]

Figure 14. Visualization of the mining policy derived by Relative Value Iteration

From the results, we can see that when the adversary has a lead, $l(a)>l(h)$, under the irrelevant state, the policy tend to choose the 'wait' action. The policy suggests the adversary continue mining on its own chain, which allows him to maintain their advantage and extend their lead. When the state is relevant, the policy suggests the 'match' action, to publish a conflicting block to create a fork. This is beneficial to the adversary as their current chain is much longer than the honest chain, allowing it to potentially overtake the honest chain and receive significant rewards.

Secondly, when the honest chain is much longer than the adversary/miner chain, the policy tend to abandon their chain, as reflected by '*' which indicates unattainable states. This suggests that instead of continuing to mine on their shorter chain and hoping to catch up, the policy prefer to abandon this effort and not take any action.

Finally, when the adversary chain reaches the maximum fork length of 8, the policy adopts the 'override' action. It continues publishing one more block to the network to ensure its chain maintains the maximum fork length. This secures the adversary position in having the longest chain, and by doing so, they can maximize their rewards and influence within the network.

5 Project Schedule

The project has been finished, and all designated tasks have been accomplished. Table 5 displays the development schedule and records the tasks undertaken throughout the final year project.

Time	Objectives
Sep 2023 (60 learning hours)	Focus: Project Setup and Detailed Project Plan - Define project objectives, scope, and deliverable - Develop a detailed project plan, including timelines and resource allocation - Set up the WordPress website for progress updates
Oct 2023 (60 learning hours)	Focus: Literature Review and Data collection - Conduct research on existing and related works on blockchain with machine learning - Collect historical blockchain data from various sources
Nov 2023 – Jan 2024 (200 learning hours)	Focus: Environment Setup and Model Training - Develop regression models to aid in nonce finding. - Develop an RL algorithm based on the average reward policy for transaction selection. - Train and fine-tune model parameters. - Define metrics and baseline models for performance evaluation and comparison.
Feb – Mid April 2024 (200 learning hours)	Focus: Model Improvements and Experiments - Attain more training data to perform more rigorous and detailed experiments. - Improve the RL reward policy to a fee-to-weight ratio, garnering better results. - Develop a Markov Decision Process to derive the optimal mining strategy.
Late Apr 2024 (80 learning hours)	Focus: Documentation and Reporting: - Document the project findings, methodologies, and outcomes. - Prepare the final project report and presentation summarizing the research, analysis, and results.

Table 5. Project schedule

6 Limitations and Future works

In this section, the limitations of our project and identify key areas for further research and development are evaluated. While our study made promising strides in applying machine learning in blockchain mining, several challenges remain that need to be addressed to enhance the robustness, scalability, security and environmental sustainability of our work.

6.1 Scalability and Validation of Experiments

Our initial experiments were conducted on relatively small-scale simulations. This provided a solid starting point but did not fully encapsulate the complexities and variables of broader applications. To enhance the credibility and applicability of our findings, future work should scale these experiments to involve more extensive datasets and more complex simulation environments. This could examine the robustness and effectiveness of our algorithms under conditions that more closely mimic the actual blockchain systems.

By conducting large-scale experiments, findings can be further validated by demonstrating consistent results across a variety of scenarios and setups. This will involve rigorous statistical analysis to ensure that the improvements observed are not only statistically significant but also repeatable and reliable under different conditions. Extending the scale of testing also allows us to explore the limits and capabilities of our algorithms, affirming their potential for broader use.

6.2 Deployment in Real-World Applications

Although our theoretical findings are promising, translating these models into practical applications presents several challenges. The integration of our models with existing blockchain systems requires careful consideration of compatibility and efficiency across different technological infrastructures.

Furthermore, the models should be capable of handling real-time data and ensuring robustness and security in diverse environments. Effective real-time data processing is essential for the models to be relevant and useful in actual blockchain operations. Future research should also aim to maintain high levels of security and system integrity under varied operational conditions. With these improvements, theoretical models could be translated into practical applications that can perform reliably and effectively in the field.

6.3 Security of blockchain network

Introducing machine learning to Proof of Work (PoW) blockchain mining could significantly alter the difficulty of the mining process. This thereby may introduce new security vulnerabilities. Machine learning models might enable miners to forecast nonce values and other aspects of the mining puzzle more efficiently than traditional methods. This efficiency, while beneficial in reducing computational effort and energy consumption, could lead to disparities in mining power. A small number of miners might therefore leverage advanced machine learning techniques to dominate the mining process, disrupting the decentralized nature of blockchain.

Moreover, if machine learning models are trained to manipulate the difficulty adjustment algorithms — which are designed to ensure a consistent block time despite fluctuating network power — they could artificially lower the difficulty levels, making the blockchain

susceptible to attacks such as double-spending. By making it easier to mine blocks rapidly, an attacker could also alter previously confirmed blocks if they achieve substantial control over the network's hash rate. This fundamentally undermines the blockchain's integrity, where the security commonly relies on the impracticality of altering all subsequent blocks following a transaction. Therefore, as machine learning integrates into blockchain mining, it is essential to carefully consider and mitigate potential impacts on network security and the equitable distribution of mining power.

6.4 Promoting Sustainability in Blockchain Mining

Our project highlighted the potential for more energy-efficient practices in blockchain mining, a critical step toward reducing the technology's environmental footprint. Further research should focus on related areas and develop methods that not only lessen the environmental impact but also enhance the performance and security of blockchain systems. With these improvements, we could achieve a greener and more sustainable future in blockchain mining.

7 Conclusion

To conclude, blockchain miners today heavily rely on the brute force method for mining. This approach, however, consumes immense computational resources and has led to various environmental problems. Inspired by recent advancements in artificial intelligence, this project seeks to explore the application of machine learning to enhance the efficiency and sustainability of blockchain mining. More specifically, we addressed the problem from three perspectives. 1) Different nonce iteration methods were explored, and regression models were trained to predict the starting nonce for nonce iteration. 2) A reinforcement algorithm based on the Fee-to-Weight ratio reward policy was developed to optimize block transaction selection while adhering to the block weight limit. 3) The blockchain mining process was modeled as a Markov Decision Process and solved it to attain an optimal mining strategy. Our experiments and blockchain simulations demonstrated that the proposed methods were able to achieve significant improvements in mining rewards and efficiency compared to traditional methods. Finally, it is hoped that the insights gained from this project will encourage miners to adopt more environmentally conscious mining methods and contribute to a greener future in blockchain mining.

Reference List

[1] Joseph Bonneau; Andrew Miller; Jeremy Clark; Arvind Narayanan; Joshua A. Kroll; Edward W. Felten, "SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies," in *IEEE symposium on security and privacy*, 2015.

[2] Ittay Eyal, Emin Gun Sirer, "Majority is not Enough: Bitcoin Mining is Vulnerable," *Communications of the ACM,* vol. 61, no. 7, pp. 95-102, 2018.

[3] S. Nakamoto, A Peer-to-Peer Electronic Cash System, Decentralized business review, 2008.

[4] Juan A. Garay, Aggelos Kiayias, Nikos Leonardos, "The Bitcoin Backbone Protocol: Analysis and Applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2015.

[5] Raza, Ali, Kyunghyun Han, and Seong Oun Hwang, "A framework for privacy preserving, distributed search engine using topology of DLT and onion routing," *IEEE Access ,* vol. 8, pp. 43001-43012, 2020.

[6] Katie Martin, Billy Nauman, "Bitcoin's growing energy problem: 'It's a dirty currency'," Financial Times, 2021.

[7] Esteban Landerreche, and Marc Stevens, "On immutability of blockchains," in *European Society for Socially Embedded Technologies*, 2018.

[8] Sunny King, and Scott Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," in *self-published paper*, 2012.

[9] Miguel Castro, Barbara Liskov, "Practical byzantine fault tolerance," *OsDI,* vol. 99, no. 1999, 1999.

[10] Taotao Wang, Soung Chang Liew, and Shengli Zhang, "When Blockchain Meets AI: Optimal Mining Strategy Achieved By Machine Learning," in *International Journal of Intelligent Systems*, 2019.

[11] Wenbo Wang; Dinh Thai Hoang; Peizhao Hu; Zehui Xiong; Dusit Niyato; Ping Wang; Yonggang Wen; Dong In Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE journal,* vol. 7, pp. 22328-22370, 2019.

[12] M. B. Taylor, "The evolution of bitcoin hardware," *Computer,* vol. 50, no. 9, pp. 58-66, 2017.

[13] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, Srdjan Capkun, "On the Security and Performance of Proof of Work Blockchains," *2016 ACM SIGSAC Conference on Computer and Communications Security,* pp. 3-16, 2016.

[14] K. P. Murphy, Probabilistic machine learning: an introduction, MIT press, 2022.

[15] A. O. Sykes, An introduction to regression analysis, 1993.

[16] Uday Kiran CH, K. Deekshith, V. Alekhya, "Bitcoin price prediction based on linear regression and lstm," *South Asian Journal of Engineering and Technology,* vol. 12, no. 3, pp. 87-95, 2022.

[17] Kirill Smelyakov, Oleksandr Bizkrovnyi, Natalia Sharonova, Serhii Smelyakov, and Anastasiya Chupryna, "Building of Regression Models for Cryptocurrency Price Prediction," *COLINS,* pp. 1216-1232, 2022.

[18] Jeyasheela Rakkini Simon, and K. Geetha, “Block Mining reward prediction with Polynomial Regression, Long short-term memory, and Prophet API for Ethereum blockchain miners,” in *ITM Web of Conferences*, 2021.

[19] Meennapa Rukhiran, Songwut Boonsong, and Paniti Netinant, “Sustainable Optimizing Performance and Energy Efficiency in Proof of Work Blockchain: A Multilinear Regression Approach,” *Sustainability,* vol. 16, no. 4, p. 1519, 2024.

[20] E. Ostertagová, “Modelling using polynomial regression,” *Procedia Engineering,* vol. 48, pp. 500-506, 2012.

[21] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, An introduction to statistical learning, vol. 112, New York: springer, 2013.

[22] Douglas C. Montgomery, Elizabeth A. Peck, and G. Geoffrey Vining, “Introduction to linear regression analysis,” in *John Wiley & Sons*, 2021.

[23] L. Breiman, “Random forests,” *Machine learning,* vol. 45, pp. 5-32, 2001.

[24] Richard S. Sutton, and Andrew G. Barto, Reinforcement learning: An introduction, MIT press, 2018.

[25] Rohit R. Mahatungade, Prakashgoud Patil, and Manjula K. Pawar, “Performance analysis of Reinforcement Learning for Miner Selection in Blockchain,” in *2023 4th International Conference for Emerging Technology (INCET)*, 2023.

[26] Roi Bar-Zur, Danielle Dori, Sharon Vardi, Ittay Eyal, and Aviv Tamar, “Deep bribe: Predicting the rise of bribery in blockchain mining with deep RL,” in *2023 IEEE Security and Privacy Workshops (SPW)*, 2023.

[27] Roi Bar-Zur, Ameer Abu-Hanna, Ittay Eyal, Aviv Tamar, “WeRLman: to tackle whale (transactions), go deep (RL),” in *15th ACM International Conference on Systems and Storage*, 2022.

[28] Charlie Hou, Mingxun Zhou, Yan Ji, Phil Daian, Florian Tramer, Giulia Fanti, and Ari Juels, “SquirRL: Automating attack analysis on blockchain incentive mechanisms with deep reinforcement learning,” in *arXiv preprint arXiv:1912.01798*, 2019.

[29] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, Koray Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *International conference on machine learning*, 2016.

[30] M. L. Puterman, “Markov decision processes: discrete stochastic dynamic programming,” in *John Wiley & Sons*, 2014.

[31] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar, “Optimal selfish mining strategies in bitcoin,” in *Financial Cryptography and Data Security: 20th International Conference*, 2017.

[32] R. Bellman, “Dynamic programming,” *science,* vol. 153, no. 3731, pp. 34-37, 1966.

[33] A. Gosavi, “Relative value iteration for average reward semi-Markov control via simulation,” in *2013 Winter Simulations Conference (WSC)*, 2013.

[34] Christopher JCH Watkins, and Peter Dayan, “Q-learning,” *Machine learning,* vol. 8, pp. 279-292, 1922.

[35] Elena Pashenkova, Irina Rish, Rina Dechter, “Value iteration and policy iteration algorithms for Markov decision problem,” *AAAI’96: Workshop on Structural Issues in Planning and Temporal Reasoning,* vol. 39, 1996.

[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King,

Dharshan Ku, “Human-level control through deep reinforcement learning,” *nature,* vol. 518, no. 7540, pp. 529-533, 2015.

[37] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, Nando de Freitas, *Sample efficient actor-critic with experience replay,* arXiv preprint, 2016.

[38] Yonatan Sompolinsky and Aviv Zohar, “Secure high-rate transaction processing in Bitcoin”.

[39] Hamza Baniata, Radu Prodan, Attila Kertesz, *Machine Learning for Alternative Mining in PoW-based Blockchains: Theory, Implications and Applications,* Authorea Preprints, 2023.

[40] S. A. Alahmari, “Predicting the price of cryptocurrency using support vector regression methods,” *J. Mech. Continua Math. Sci,* vol. 15, no. 4, pp. 313-322, 2020.